Trends in Contemporary Computer Science

Podlasie 2014

Edited by
Anna Gomolińska, Adam Grabowski,
Małgorzata Hryniewicka, Magdalena Kacprzak,
and Ewa Schmeidel

Białystok 2014

Typesetting: Adam Grabowski
Cover design: Anna Poskrobko

Supported by
the Polish Ministry of Science and Higher Education

Bialystok University of Technology Publishing Office
Białystok 2014

ISBN 978-83-62582-58-7

Table of Contents

III Optimization Techniques and Decision-making Processes

IV Formal Methods and Data Mining

Preface

This monograph presents the recent results obtained by the Podlasie computer scientists and their colleagues working among the wide world. The book consists of sixteen chapters and is divided into four parts.

In Part I, comprising four chapters, some aspects of formalization of mathematics are presented. In Chapter 1, almost distributive lattices (ADL) being structures defined by Swamy and Rao in 1981 as the generalization of ordinary distributive lattices by removing certain conditions from the well-known axiomatization of these are considered. As distributive lattices are pretty well present in the Mizar Mathematical Library (MML), also formal characterization of ADL in terms of Mizar attributes and clusters are given. Many of the lemmas and counterexamples can be obtained semiautomatically with the help of external provers (e.g. Prover9 and MACE), and also internal Mizar utilities can improve human work in this area. A new formalization of crucial parts of the chosen results obtained by Rao is presented. The characterization of the class of generalized almost distributive lattices is obtained as well.

The fundamental theorem of finite abelian groups, which describes the structure of these groups, is formalized in Chapter 2. Since the theorem underlies the study of finite abelian groups, it is expected to become widely applied in formalizing fields such as number theory and cryptology as future Mizar results.

In Chapter 3, an original theorem in auction theory is provided. It specifies general conditions under which the sum of the payments of all bidders is granted not to be identical to zero, and more generally – not to be constant. Moreover, it explicitly supplies a construction for a finite minimal set of possible bids on which such sum is not constant. In particular, the theorem presented applies to the important case of the second-price Vickrey auction, where it reduces itself to a basic result of which a novel, alternative proof is given. In order to enhance the confidence in this new theorem, it has been formalized in Isabelle/HOL: the main results and definitions of the formal proof are reproduced here in the common mathematical language, and accompanied with an informal discussion about the underlying ideas.

Chapter 4 is devoted to the analysis of algorithms with Mizar. The approach proposed consists in the Mizar formalization of abstract concepts like algebra of instructions, execution function, termination, and their substantiation in a model with integers as the only data type and in models with abstract data types. The proof of correctness of the algorithm Exponentiation by Squaring is described.

Part II is devoted to the memory of Andrzej Trybulec, a pioneer of computer–assisted formalization of mathematical problems. Over the last decades we witnessed a number of successful instances of such formalization. Research in this field has been boosted by the development of systems for practical formalization of mathematics (proof assistants), creation of large repositories of computer–verified formal mathematics, and integration of interactive and automated meth-

ods of theorem proving. Proof assistants provide a very useful teaching tool suitable for undergraduate instruction, in particular for training beginning students in writing rigorous proofs. Thus, interactive theorem proving is the main subject of this part of the monograph. The part comprises five chapters.

The usage of extensionality of sets, possibly satisfying some additional properties, by proof checkers is presented in Chapter 5. In particular, it is shown how extensionality influences proof tactics and equational calculus. Short descriptions of Mizar constructions, having an impact on two basic Mizar modules: Reasoner for supporting definitional expansions and Equalizer for computing of the congruence closure of a given set of equalities, are collected.

Chapter 6 is devoted to the formal proof checking systems such as Mizar and Isabelle/Isar. These systems can verify the correctness of proof scripts, both easily readable and obscure. However, for humans like those analysing of the main idea of a formal proof or redeveloping of fragments of reasoning to make them stronger, the legibility has a substantial significance. Furthermore, proof writers create still more and more complex deductions that cannot be shortened to several steps by any tools currently available. Therefore, it is important to understand better how we can facilitate the work of script readers modifying the order of independent deduction steps or how we can reorganize the proof structure by extracting lemmas that are obtained automatically. Experimental results are presented, obtained with a method that improves proof legibility and is based on human short-term memory.

Chapter 7 is devoted to a brief description of the logical and semantic approaches being developed in the Kyiv school of automated reasoning. This approach can be traced back to 1970, when academician V. Glushkov initiated research on automated theorem proving in mathematics, which is known as the Evidence Algorithm programme (EA) having many common theses with the Polish Mizar project. A significant attention has been paid to the study and the development of logic and semantics as well as a technique for logical inference search. Carried out first at the Institute of Cybernetics of NASU, these investigations moved in 1987 to the Faculty of Cybernetics of the National University of Kyiv, where now they are developed in two directions. The first direction is the traditional one, centered mainly on the construction of proof methods in various first-order logics. The second direction aims at developing of logics oriented on semantic models of programs. It is expected that the results obtained will give us a possibility to extend SAD with more expressive logical languages and more powerful reasoning tools.

Though more and more advanced theorems have been formalized in proof systems, their presentation still lacks the elegance of the mathematical writing. The reason is that proof systems have to state much more details – a large number of which is usually omitted by mathematicians. In Chapter 8, the authors argue that proof languages should be improved into this direction to make proof systems more attractive and usable – the ultimate goal of course being a like-on-paper presentation. It has been shown that using advanced Mizar typing techniques the results of formalization look pretty close to the mathematical

paper style. Consequently, users of proof systems should be supplied with environments providing and automating these techniques, so that they can easily benefit from these.

In Chapter 9, the author surveys and describes the implementation of parallelization of the Mizar proof checking and of related Mizar utilities. The implementation makes use of Mizar's compiler-like division into several relatively independent passes, with typically quite different processing speeds. The information produced in earlier (typically much faster) passes can be used to parallelize the later (typically much slower) passes. The parallelization now works by splitting the formalization into a suitable number of pieces that are processed in parallel, assembling from them together the required results. The implementation is evaluated on examples from the Mizar library, and future extensions are discussed.

Parts III and IV show a recent study in theoretical and applied computer science. Part III contains four chapters devoted to optimization techniques and decision-making processes.

Chapter 10 describes the project of innovative software component – LOGTRAVEL which supports planning and organization of touristic travels. The implementation of LOGTRAVEL requires the solution of many optimization problems on graphs. These problems are variations of a computationally difficult task called Orienteering Problem, also known under the Selective Travelling Salesman Problem with Profits.

The new method of solving the Orienteering Problem with Time Windows is provided in Chapter 11. The authors propose a hybrid heuristic which combines genetic algorithms and the path relinking strategy. Computer experiments has shown that the proposed method gives better solutions in comparisons to the previous described in the literature algorithms. The proposed method can be applied to solve realistic problem of planning the most valuable routes for tourism.

Different types of graph weights representations used to solve the Time-Dependent Orienteering Problem was shown in Chapter 12. The author has applied three variants: real time-dependent weights, mean weights and the hybrid of previous two. As optimization algorithm the randomized, local search was used. Tests were conducted on real public transport network of Białystok.

Chapter 13 is devoted to applications of metasets in making decisions. Metasets are designed to represent and process vague, imprecise data, similarly to fuzzy sets or rough sets. The definitions of metaset and related notions are directed towards efficient computer implementations and applications. The proposed approach can be used in automated personalized tour-planning devices. In particular, it can be a starting point in the Orienteering Problem. The authors demonstrate an example of the application of first-order metasets to solving the problem of finding the most appropriate holiday destination for a tourist on real data from the city of Białystok.

Part IV refers to formal methods and data mining, in particular to the following three topics: (a) analysis of formal models of computer systems which are

represented by graphs, (b) extracting knowledge from data structures, and (c) structural analysis of musical pieces.

Chapter 14 presents the progress in the development of techniques for automatically verifying correctness properties of finite-state software and hardware systems. The authors deal with the problem of verification of properties expressible in the language of Metric Temporal Logic by means of Bounded Model Checking. Model checking refers to the problem in which given a model of the system, it is automatically checked whether this model meets a given specification. In order to solve such a problem, both the model of the system and the specification are formulated in some precise mathematical language. To this end, it is formulated as a task in logic, namely to check whether a given structure satisfies a given logical formula.

Chapter 15 is devoted to the problem of computer-aided diagnostics. Its aim is to present an algorithm for finding action rules from medical databases. The classical knowledge discovery algorithms have the potential to identify enormous number of significant patterns from data. But at the same time people are overwhelmed by a large number of uninteresting patterns. Therefore, a need for new methods with the ability to assist users in analyzing a large number of rules for a useful knowledge is seeking. An action rule is a rule extracted from a decision system that describes a possible transition of objects from one state to another with respect to a distinguished attribute called a decision attribute.

Chapter 16 focus on construction of specialized grammars covering music information. The authors present a grammar-oriented searching operations for analyzing a musical composition. They propose three operators (rhythm ratio, scalar difference and half difference operator) that describe relations between neighboring notes, regarding melody and rhythm. The operators can be used for searching of transformed and non-transformed motives, analysis of melodic and rhythmical sequences, structure discovery and comparative analysis of musical pieces.

The main goal of this monograph is to report on some actual results and directions of the research conducted both by computer scientists in Podlasie and their colleagues in Poland and abroad. The exchange of ideas and, more generally, the collaboration among various groups of researchers are clearly of a paramount importance for the further development of computer science in our region. The edition of the monograph was supported by the Polish Ministry of Science and Higher Education.

Editors of the volume:
Anna Gomolińska
Adam Grabowski
Małgorzata Hryniewicka
Magdalena Kacprzak
and Ewa Schmeidel

Białystok, June 2014

Part I

Computer-Assisted Formalization of Mathematics

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Formal Characterization of Almost Distributive Lattices

Adam Grabowski

Institute of Informatics
University of Białystok
ul. Akademicka 2, 15-267 Białystok, Poland
adam@math.uwb.edu.pl

Abstract. Almost distributive lattices (ADL) are structures defined by Swamy and Rao in 1981 as the generalization of ordinary distributive lattices by removing certain conditions from well-known axiomatization of these. As distributive lattices are pretty well present in the Mizar Mathematical Library (MML), we decided to give also formal characterization of ADL in terms of Mizar attributes and clusters. Many of the lemmas and counterexamples can be obtained semiautomatically with the help of external provers (e.g. Prover9 and MACE), also internal Mizar utilities can improve human work in this area. We formalized crucial parts of the chosen papers of Rao (some of them are pretty recent), obtaining also the characterization of the class of generalized almost distributive lattices.

1 Introduction

The problem of the formalization of mathematics with the help of computer is rather old. Finding equivalent sets of axioms for various classes of structures was also a challenge for machine proof-assistants for years. Boolean lattices are special field here, if we recall the solution of Robbins problem obtained with the help of EQP/Otter prover by William McCune back in 1996. Also other varieties of lattices are important for representation reasons, to enumerate here modular or distributive ones, at least.

We decided to formalize this concrete topic for some reasons: one of them was to eventually give the formal description of almost distributive lattices – the topic which was suggested many years ago by Andrzej Trybulec. The other one was to give some answers for open questions formulated by G.C. Rao (including associativity of ADLs). Finally, we hoped to obtain uniform characterization of these structures (including some generalizations) as in many papers their axiomatization was not clear and unique (as e.g. zero was sometimes included among the axioms, in other cases the structures were explicitly given as ADLs with zero).

If we treat this just as giving another equationally-defined class of structures (as modular, distributive, implicative, pseudo-complemented, etc.), the task is not very hard; it can be also of rather minor interest as the impact of this

specific class of almost distributive lattices for the mathematics as a whole is rather small. On the other hand, recently we have proven a generalized version of Nachbin theorem for spectra of distributive lattices, and we would try to apply similar techniques in the case of ADLs.

The paper is organized as follows: at the beginning we draw the state of the current formalization of the lattice theory within the Mizar Mathematical Library (MML); in Section 3 we show how much the problem of finding equivalent (alternative) characterization of lattices was influential for the world of automatic proof-assistants. The fourth section will explain the notion of an almost distributive lattice and later we will show some details of the formalization of ADLs within Mizar proof checker. Finally we draw some conclusions and plans for future work.

2 Lattices in Mizar

Lattices can be defined by the well-known equational characterization (the structure $\langle L, \sqcup, \sqcap \rangle$, where $\sqcup$ and $\sqcap$ are binary operations defined on the non-empty set L; both operations are commutative, associative and satisfy the absorption laws).

The other definition, which is based on the ordering relation $\langle L, \leqslant \rangle$, assumes that the binary relation $\leqslant$ is reflexive, transitive, and antisymmetric, that is, $\leqslant$ partially orders L, and all binary suprema and infima exist. Both approaches are equivalent; the common structure in which we have the ordering relation as a field in the structure and both lattice operations is relatively recent; we eventually merged both into the single structure called `LattRelStr` (the details of the approach can be found in [4]).

Relational-structure-based approach was heavily used when formalizing *Compendium of Continuous Lattices* in Mizar [1] between 1996 and 2003 (a long-term project, resulting in `YELLOW` and `WAYBEL` series, 58 articles in total). `LATTICE` series however was significantly older back to 1989 and Żukowski submission [14] (this was the second article ever in MML using the notion of a structure).

Simple grepping reports 188 articles using `LATTICES`, but the real test is the scanning of `dno` files (the files using not anything connected with `LATTICES` file – including reserved symbols, e.g. `"\/"`, but only the notation, which corresponds with the actual use of the notion of a lattice). This searching reports 94 Mizar articles; some of them are only construction of certain lattices (of subspaces of a vector space, of subalgebras of an algebra, etc.), but still there's a significant number (as there's about 1200 articles in the whole MML).

For sure, the author of original submission [14] had not a chance to foresee the development of adjectives and clusters mechanism; it is not very strange that, e.g. the existence of the zero element in a lattice was guaranteed by the conjunction of four equalities (some of them are of course superfluous in the general lattice). We still have some hard decisions to make: either to exchange our approach with the original one or just to let things be developed in parallel and to let them meet eventually in the area they have to meet. In our approach

we wrote the next article in LATTICES series, so we are closer to equationally defining class of structures rather that starting from the orderings – it's not very strange as the ordering occuring in Rao's papers is not even partial order.

3 The Problem of Equivalent Characterizations

Giving equivalent axiomatizations, although not looking as a challenging point for a human at a first glance (because it's usually very labour-consuming task and the results are not really innovative as they don't push the theory too much forward), can serve as a testbed for computer systems. Alternative characterization of Boolean algebras effected in probably one of the most famous results obtained with the help of automatic theorem provers – the solution of Robbins problem [7].

Various classes which arose in this direction and already present in MML are:

Sheffer stroke-based lattices – the characterization of Boolean lattices in terms of the Sheffer stroke ($x|y = x' \sqcup y'$), with the characterization of Boolean algebras in terms of single short axiom (SHEFFER2);

Robbins algebras – the Mizar formalization of the solution of the Robbins problem that satisfying the equation

$$\big((a+b)' + (a+b')'\big)' = a$$

(additionally $+$ is associative and commutative, and the dual operation is given by the ordinary de Morgan laws) makes the lattice Boolean (ROBBINS1, both in the original version given by EQP/Otter, and in the significantly shortened version);

Huntington algebras – this was the first step in the proof of the solution of Robbins problem, contained also in ROBBINS1; Huntington axiom is $(a' + b')' + (a' + b)' = a$;

Short axiom systems – of course, the abovementioned Sheffer stroke is included here; useful collection of various equational systems for lattices is given in [9].

Here we can point out that relatively powerful mechanism of structures can make a problem here; once the structure is under consideration, the operations are determined, so if we want to give e.g. the characterization of Boolean algebras in terms of the Sheffer stroke, so-called merging mechanism for structures should have been applied.

4 Tempting Generalizations

Almost distributive lattices according to [12] are structures

$$\langle L, \sqcup, \sqcap, 0 \rangle$$

with the following axioms:

$$(x \sqcup y) \sqcap z = (x \sqcap z) \sqcup (y \sqcap z) \qquad (x \sqcup y) \sqcap y = y$$
$$x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z) \qquad (x \sqcup y) \sqcap x = x$$
$$0 \sqcap x = 0 \qquad x \sqcup (x \sqcap y) = x$$

for arbitrary $x, y, z \in L$.

First surprise was the naming scheme: almost distributive lattices are not lattices at all! Essentially, either commutativity of $\sqcup$ or $\sqcap$, or remaining distributivities were dropped. Also the role of zero is uncertain.

As always, one can pose a question of the direction of generalization: how far we can go? One can imagine the series of papers which stems from excluding some among seven axioms for distributive lattices and discuss what we can expect (it is not really question of the number seven; e.g. there are two absorption laws, but if we have no commutativity, the number extends – similarly in the case of the distributivity, observe that we have no general distributivity in GADLs). ADLs have some mathematical origins (Baer semirings), but further generalization work can not be sufficiently justified.

Generalized almost distributive lattices have not less axioms, but just different ones, and the implication that all ADLs are also GADLs is not very clear at the very first sight (Def. 3.1 in [11]):

$$(x \sqcap y) \sqcap z = x \sqcap (y \sqcap z) \qquad x \sqcap (x \sqcup y) = x$$
$$x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z) \qquad (x \sqcup y) \sqcap x = x$$
$$x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z) \qquad (x \sqcap y) \sqcup y = y$$

But the axiom set for GADLs is just nicer to understand and to remember; which is also important, the lower bound is not included which gives more general approach. But the form of these equations forces to prove some basic properties of ADLs before to show that the latter are consequences of the earlier.

The ordering on such defined structures is given by

$$x \leqslant y \;\Leftrightarrow\; x \sqcup y = y$$

or equivalently

$$x \leqslant y \;\Leftrightarrow\; x \sqcap y = x$$

Interesting to see, although authors of [12] stated that "$\leqslant$ is partial ordering", it's not antisymmetric, hence without any additional assumptions it's not the partial order for arbitrary generalized almost distributive lattice L.

The reader should be really aware of the ordering, it's important to have the proper one – of course if we have commutativity in mind, taking either $x \sqcap y = x$ or $y \sqcap x = x$ as a definition of $x \leqslant y$ doesn't really matter, but in fact we have two various relations defined; first – classical one (already mentioned above), but the second one new (`ThetaOrder` in Mizar formalism): $x \sqcap y = y$, which is not equivalent to $y \leqslant x$. Of course the relation is reflexive and transitive; its antisymmetry makes general almost distributive lattice also commutative (so essentially calling `ThetaOrder L` the partial ordering is a strong assumption about L).

We were suggested by rough set community and Zhu's approach to get a kind of reverse mathematics in obtaining core rough approximations properties

by distinct properties of binary relations [2]; essentially the formalization in Mizar was done from scratch at the satisfying level of generality.

5 The Use of Proof-Assistants

Paradoxically, although our primary motivation was to formalize ADLs in the Mizar language to extend the Mizar Mathematical Library, our first turn was to use Prover9/MACE to obtain the answer for the question of the associativity of ADLs. As $\sqcap$ is associative as it was proven by Rao, we attacked the question formulated in [12] and repeated in 2009 in [11] about the associativity of $\sqcup$.

We started Prover9 with no positive results; we ask MACE (Models and Counterexamples) and almost immediately got the answer: there exists an ADL which is not $\sqcup$-associative.

For example, formulating an input for Prover9/MACE as follows:

```
formulas(assumptions).

%% Axioms for ADL
(x v y) ^ z = (x ^ z) v (y ^ z).
x ^ (y v z) = (x ^ y) v (x ^ z).
(x v y) ^ y = y.
(x v y) ^ x = x.
x v (x ^ y) = x.

end_of_list.

formulas(goals).

%% Associativity condition
(x v y) v z = x v (y v z).

end_of_list.
```

we obtained after a 0.03 seconds the model:

```
interpretation( 5, [number = 1,seconds = 0], [
    function(^(_,_), [
        0,1,0,1,0,
        0,1,0,1,0,
        0,1,2,3,4,
        0,1,2,3,4,
        0,1,2,3,4]),
    function(v(_,_), [
        0,0,2,4,4,
        1,1,3,3,3,
        2,2,2,2,2,
```

```
      3,3,3,3,3,
      4,4,4,4,4]),
  function(c1, [0]),
  function(c2, [1]),
  function(c3, [2])]).
```

which in more human-readable form is

c1: 0 c2: 1 c3: 2

$\sqcap$:	0	1	2	3	4
0	0	1	0	1	0
1	0	1	0	1	0
2	0	1	2	3	4
3	0	1	2	3	4
4	0	1	2	3	4

$\sqcup$:	0	1	2	3	4
0	0	0	2	4	4
1	1	1	3	3	3
2	2	2	2	2	2
3	3	3	3	3	3
4	4	4	4	4	4

In fact, in the above ADL, we have $0 \sqcup (1 \sqcup 2) = 4$, but $(0 \sqcup 1) \sqcup 2 = 2$.

Josef Urban's work on the automatic translation of Otter proof objects into Mizar is available and potentially useful, but we should also take into account model building which seems to be not as straightforward in the current state of the Mizar system.

6 The Details of the Formalization

Our first doubts before we even started the process of the translation of [12] were if the expressive power of the existing formal apparatus present in [14] is enough to give the proper axiomatization of structures which are not even lattices. We observed just the opposite situation in the series started with `ALGSTR_0` – very general setting when structures are equipped by proper attributes one by one.

Example handling is formally very complex in Mizar: of course one can just give the set of all lattice operations' results, but this can be long for larger universes; in the series of Rao's papers there are some examples of ADLs defined; these are just the tables of both lattice operations, usually on the set $\{a, b, c\}$, where a, b, c are (distinct, which is not explicitly stated) elements, we use rather natural numbers in such case. In the case of lattices, where we could use the partial ordering to characterize lattice operations, we use set-theoretical inclusions between natural numbers as examples.

Definitely much more automation could be useful in this specific case, perhaps the set of potentially useful structures with their properties shown automatically. Here we can point out the role of Sledgehammer in Isabelle which automatically tries to fill the gaps in the demonstrations; similar functionality was recently enabled by Josef Urban in Emacs, but we didn't benefit from it in its full extent.

```
definition let L be non empty LattStr;
  attr L is left-Distributive means
```

```
    for x,y,z being Element of L holds
      x "\/" (y "/\" z) = (x "\/" y) "/\" (x "\/" z);
  attr L is ADL-absorbing means
    for x,y being Element of L holds
      x "/\" (y "\/" x) = x;
end;
```

As usual, all single axioms were formulated in terms of Mizar adjectives, so equationally defined class in Mizar should be called rather defining by attributes of specific type.

```
definition
  mode GAD_Lattice is meet-associative distributive left-Distributive
    join-absorbing Meet-Absorbing meet-absorbing non empty LattStr;
end;
```

Of course, even if not really useful, we had to introduce the type of being a general almost distributive lattice, just for a record. For such structures, all further properties, as e.g. $\sqcup$-idempotence, can be proven without any major problem:

```
 reserve L for GAD_Lattice;
 reserve x,y,z for Element of L;

theorem ISum:   :: Lemma 3.4. (I \/)
  x "\/" x = x
  proof
    thus x "\/" x = ((x "\/" x) "/\" x) "\/" x by DefA2
      .= x by LATTICES:def 8;
  end;
```

If we compare the above with the original proof from [14] (as idempotence is the consequence of classical lattice axioms), we can note that as a result of revision introducing reduction registrations it was formulated as a reduction:

```
registration
  let L be meet-absorbing join-absorbing meet-commutative
    non empty LattStr,
      a be Element of L;
  reduce a "\/" a to a;
  reducibility
  proof
    thus a "\/" a = (a "/\" (a "\/" a)) "\/" a by Def9
                .= a by Def8;
  end;
end;
```

After such registration terms $a \sqcup a$ and a are automatically unified. In our Mizar article we didn't use reductions, because we were focused on clear proofs, understandable for human, by adding some more reductions our proofs will be probably much shorter.

```
theorem Th31143:  :: Theorem 3.11. (4) <=> (3)
  L is join-commutative iff L is ADL-absorbing
  proof
    thus L is join-commutative implies L is ADL-absorbing
    proof
      assume
A1:   L is join-commutative;
      let a,b be Element of L;
      a "\/" b = b "\/" a by A1;
      hence thesis by LATTICES:def 9;
    end;
    assume
B1: L is ADL-absorbing;
    let a,b be Element of L;
    a "/\" (b "\/" a) = a by B1;
    hence thesis by Th3726;
  end;

registration
  cluster join-commutative -> ADL-absorbing for GAD_Lattice;
  coherence by Th31143;
  cluster ADL-absorbing -> join-commutative for GAD_Lattice;
  coherence by Th31143;
end;
```

Both registrations of clusters together state that the associativity of the join operation and the absorption expressed in the original set of axioms for generalized almost distributive lattices are equivalent. Furthermore, the reference for Theorem 3.11 is not needed anymore.

Although we formalized quite a significant piece of basic theory of almost distributive lattices, which is a good starting point for future developments, some issues are still to be considered:

- we are not sure about the real role of zero in Rao's proofs. In earlier papers it was used really much, but in later generalization it is often missing;
- the handling of "TFAE" (the following are equivalent) – already mentioned during formalization of CCL; there are three important theorems formulated in such a mood in [11], and there is no satisfactory solution of how to formulate it in a feasible way in Mizar (of course, proofs can be in arbitrary ways, but the formulation is usually in a star-like mood);
- implementing associativity as another property; tricks with brackets are usually simple, but sometimes can be really annoying; they also can blow the real idea of the proof; commutativity is obvious and it's really nice not to refer to it;
- the programs which enhance reasoning by eliminating irrelevant proofs steps (e.g. `relinfer` and `reliters`) can be better used as after introducing some additional reduction registrations can suggest further abbreviations and improvements; it's fine as long as in the original work there is often no proof at all, but for a human it's sometimes unacceptable without longer thinking;
- paradoxically to show that almost distributive lattices are GADLs we have to prove some properties of ADLs before, but we don't see any method of avoiding it.

6.1 The issue of knowledge reuse

Essentially, from the machine viewpoint, it's not very important which collection of attributes is really needed. But human mathematician usually uses just the same set of axioms through all the paper and the reflection which generalization ways can be chosen comes usually later on. Some individual adjectives are unexpected, as the below:

```
definition
  let L be non empty /\-SemiLattStr;
  attr L is meet-Associative means
:: ROBBINS3:def 2
  for x, y, z being Element of L holds
    x "/\" (y "/\" z) = y "/\" (x "/\" z);
end;
```

which is a mix of associativity and commutativity.

We were stuck that after finishing our formalization (and during writing of this paper) we discover that we already introduced the attribute of the dual distributivity (called `left-Distributivity` in current approach and present also in the formalization of Sheffer stroke-based lattices). Of course, the one which introduced later will be removed.

```
definition
  let IT be non empty LattStr;
  attr IT is distributive' means
:: SHEFFER1:def 5
  for a, b, c being Element of IT holds
    a "\/" (b "/\" c) = (a "\/" b) "/\" (a "\/" c);
end;
```

6.2 The question of the quantifier

We were surprised in the proof of one of the lemmas by the phrase "Exchanging variables in the formula above we obtain...". In Mizar, the naming scheme of variables is really unimportant. To be honest, in the place mentioned above, we were suggested that the proof failed, and we formulated it in the generalized (i.e. with general quantifier added) form. This caused however to formulate all equivalences in such form.

But the motivation is much deeper. Rao et al. wrote Theorem 3.8 for individual variables and then restated this for lattices which satisfy commutativity for all pairs of such elements x, y:

```
theorem  :: Theorem 3.8. 1) <=> 5)
  x "\/" y = y "\/" x iff x [= y "\/" x by LemX3,Th3851;
```

6.3 What's done already

At the very beginning of our work we wanted to give the formal characterization of almost distributive lattices as defined in Rao in 1981 [12] (1980 in his PhD thesis, but this work was unavailable for us). We were surprised that the existence of zero element was so important in these proofs, and we eventually discovered that the class

of generalized almost distributive lattices (GADL) doesn't use zero at all, and those structures in which the lower bound exists, are treated especially. Then we changed our focus on the latter class.

What's not formalized was two examples – of GADLs which are not ADLs and also a problem of defining interval in GADL. Theorem 3.8 from [11] states that if $\exists_z x \leqslant z \wedge y \leqslant z$, then $x \sqcup y = y \sqcup x$. As it is commutative, it makes every interval of GADL distributive lattice; then a theory of certain substructures behaves more regular than the original.

The Mizar article which is a complete source for the formalization (all theorems are proven and the Mizar checker doesn't report any errors) was send for inclusion into Mizar Mathematical Library under MML identifier `LATTAD_1`.

7 Conclusions

We described our experiment with the help of automated proof-assistants – the formalization of almost distributive lattices, or, to be more precise, generalized almost distributive lattices. Even if the content itself is not really impressing (about 1800 lines of the Mizar source code, currently submitted for inclusion in the Mizar Mathematical Library), we are sure our approach, even if less readable for those who do not know the Mizar syntax, has some obvious advantages. Some of the lemmas from [12] were left without proof, which is quite natural, other lemmas were automatically recognized by the system as straightforward corollaries. Essentially if the reader is not interested in proofs, but only in pure mathematical content, we hope that expressing certain identities (axioms) by Mizar adjectives offers some new capabilities and enables new insight for what is really written.

Acknowledgments

The author wishes to express his gratitude to Professor Andrzej Trybulec who passed away in September 11, 2013. Originally he suggested to formalize almost distributive lattices as described in [12] but we didn't go further than just a loose reading of this paper (altough we didn't expect any major difficulties with its translation into Mizar). But some four months after Andrzej's death one of the papers on ADLs was unexpectedly sent to me for a review, so I eventually treated this with the care it deserved (and as it was expected by Prof. Trybulec for so many years; I want to believe it was just a coincidence).

References

1. Grzegorz Bancerek, Development of the theory of continuous lattices in Mizar, M. Kerber, M. Kohlhase [eds.], *Symbolic Computation and Automatic Reasoning. The Calculemus-2000 Symposium,* A K Peters, 2000, pp. 65–80.
2. Adam Grabowski, Automated discovery of properties of rough sets, *Fundamenta Informaticae,* 128(1–2), 2013, pp. 65–79.
3. Adam Grabowski, Artur Korniłowicz, and Adam Naumowicz, Mizar in a nutshell. *Journal of Formalized Reasoning, Special Issue: User Tutorials I,* **3**(2), 2010, pp. 153–245.

4. Adam Grabowski and Markus Moschner, Managing heterogeneous theories within a mathematical knowledge repository. In: A. Asperti, G. Bancerek, A. Trybulec (eds.) Proc. of Third International Conference on Mathematical Knowledge Management, *Lecture Notes in Computer Science*, 3119, 2004, pp. 116–129.
5. Adam Grabowski and Christoph Schwarzweller, Translating mathematical vernacular into knowledge repositories, in: Proc. of the 4th international conference on Mathematical Knowledge Management, MKM 2005, Bremen, Germany, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 49–64.
6. Artur Korniłowicz, On rewriting rules in Mizar. *Journal of Automated Reasoning,* 50(2), 2013, pp. 203–210.
7. William McCune, Solution of the Robbins problem, *Journal of Automated Reasoning,* 19, 1997, pp. 263–276.
8. Adam Naumowicz and Artur Korniłowicz, A brief overview of Mizar. In: S. Berghofer, T. Nipkow, C. Urban, M. Wenzel (eds.) Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics, *TPHOLs'09, Lecture Notes in Computer Science*, vol. 5674, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 67–72.
9. R. Padmanabhan and S. Rudeanu, *Axioms for lattices and Boolean algebras,* World Scientific, 2008.
10. Karol Pąk, Methods of lemma extraction in natural deduction proofs, *Journal of Automated Reasoning,* 50(2), 2013, pp. 217–228.
11. G.C. Rao, R.K. Bandaru, and N. Rafi, Generalized almost distributive lattices – I, *Southeast Asian Bulletin of Mathematics,* 33, 2009, pp. 1175–1188.
12. U.M. Swamy and G.C. Rao, Almost distributive lattices, *Journal of Australian Mathematical Society (Series A),* 31, 1981, pp. 77–91.
13. Josef Urban, Piotr Rudnicki, and Geoff Sutcliffe, ATP and presentation service for Mizar formalizations, *Journal of Automated Reasoning,* 50(2), 2013, pp. 229–241.
14. Stanisław Żukowski, Introduction to lattice theory, *Formalized Mathematics*, 1(1), 1990, pp. 215–222.

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Formalization of Fundamental Theorem of Finite Abelian Groups in Mizar*

Kazuhisa Nakasho, Hiroyuki Okazaki
, Hiroshi Yamazaki, and Yasunari Shidama

Shinshu University
Nagano, Japan
13st205f@shinshu-u.ac.jp, okazaki@shinshu-u.ac.jp,
yamazaki@shinshu-u.ac.jp, shidama@shinshu-u.ac.jp

Abstract. In this paper, we describe the formalization of the fundamental theorem of finite abelian groups in Mizar language. This theorem describes not only the structures of finite abelian groups, but also a complete classification of them. Since the theorem underlies the study of finite abelian groups, it is expected to become widely applied in formalizing fields such as number theory and cryptology as future Mizar papers.

1 Introduction

Mizar [1, 11, 2, 3], initiated and directed by Andrzej Trybulec since 1973, is one of the oldest and largest computer-aided mathematical knowledge management systems. The Mizar project attempts to build a formalized library covering basic facts of all mathematics. The high readability of the Mizar language enables any mathematician to read its papers without deep knowledge of the Mizar grammar. In 2013, the Mizar Mathematical Library (MML) contains more than 11,600 definitions of mathematical concepts and more than 53,700 theorems. All MML papers are available on the Internet under the GPLv3 and CC BY-SA 3.0 license. The Mizar community remains actively engaged in expanding MML contents.

In this paper, we describe the formalization of the fundamental theorem of finite abelian groups [4, 5] in Mizar language. This theorem states that a finite abelian group can be decomposed into a direct sum of finite cyclic subgroups. The theorem has two different forms. The first form is the decomposition into a direct sum of cyclic groups of prime power order. The second form is called invariant factor decomposition, which gives a classification of finite abelian groups.

The formalization task in Mizar is divided into three parts [9, 10]. In the first part, we formalize the decomposition of a finite abelian group into a direct sum of finite abelian subgroups of prime power order. In the second part, we discuss the decomposition of a finite abelian group of prime power order into a direct sum of cyclic subgroups of prime power order. In the final part, we prove the targeted theorem. Here, we formalize that a two-level direct sum decomposition can be expressed as a one-level direct sum decomposition and apply this result.

* This work was partly supported by JSPS KAKENHI 22300285

2 Group and Some Basic Concepts

In this section, we give an introduction to the formalization of group and a few of its basic concepts in Mizar.

In Mizar, the definition of group is formalized as below:

Definition 1 (Group in Mizar).

```
definition
  let IT be multMagma;
  attr IT is unital means
:: GROUP_1:def 1
  ex e being Element of IT st for h being
  Element of IT holds h * e = h & e * h = h;
  attr IT is Group-like means
:: GROUP_1:def 2
  ex e being Element of IT st for h being Element of IT holds
  h * e = h & e * h = h & ex g being Element of
  IT st h * g = e & g * h = e;
  attr IT is associative means
:: GROUP_1:def 3
  for x,y,z being Element of IT holds (x*y)*z = x*(y*z);
end;

definition
  mode Group is Group-like associative non empty multMagma;
end;
```

Here, **multMagma** is magma (or groupoid), which consists of a set of elements and a single binary operator on it. So Definition 1 states the following:

Definition 2 (Group). *A group $(G, \cdot)$ is a set of elements with a binary operation $\cdot : G \times G \to G$ that satisfies the following conditions:*

1. **Associativity** *For all $a, b, c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.*
2. **Identity element** *There exists an element e such that $e \cdot a = a \cdot e = a$ for all $a \in G$.*
3. **Inverse element** *For all $a \in G$, there exists an element $b \in G$ with $a \cdot b = b \cdot a = e$.*

In Mizar, subgroup is formalized as **mode**. (**mode** is one of the type declaration keywords in Mizar language.)

Definition 3 (Subgroup in Mizar).

```
definition
  let G be Group-like non empty multMagma;
  mode Subgroup of G -> Group-like non empty multMagma means
:: GROUP_2:def 5
    the carrier of it c= the carrier of G
  & the multF of it = (the multF of G) || the carrier of it;
end;
```

the carrier of G is a set of all elements in G, and **the multF of G** is a binary operator of G. So the definition is equivalent to the following:

Definition 4 (Subgroup). *Let G is a group. Then H is a subgroup of G means*

1. *H is a group.*
2. *A set of elements of H is a subset of that of G.*
3. *A binary operator of H is the restriction of that of G on a set of elements of H.*

In Mizar, the definition of abelian group is formalized as an attribute of **multMagma** as following:

Definition 5 (Abelian group in Mizar).

```
definition
  let IT be multMagma;
  attr IT is commutative means
:: GROUP_1:def 12
  for x, y being Element of IT holds x*y = y*x;
end;
```

It is interpreted into natural language as follows:

Definition 6 (Abelian group). *A magma M is called abelian if M satisfies the following condition:*

- **Commutativity** *For all $a, b \in M$, $a \cdot b = b \cdot a$.*

It is helpful to take a look at the definition of the following terms: **cyclic** and **prime_power_order**.

Definition 7 (Cyclic group in Mizar).

```
definition
  let IT be Group;
  attr IT is cyclic means
:: GR_CY_1:def 7
  ex a being Element of IT st the multMagma of IT = gr {a};
end;
```

Definition 8 (Cyclic group). *A group G is called cyclic if there is $a \in G$ with $G = \langle a \rangle$, where $\langle a \rangle = \{a^n | n \in \mathbb{Z}\}$.*

Definition 9 (Group with prime power order in Mizar).

```
definition
  let G be Group;
  attr G is prime_power_order means
  ex p be Prime, m be Nat st card G = p |^ m;
end;
```

Definition 10 (Group with prime power order). *A group G is called prime power order if there is a prime number p and the order of G is a power of p.*

3 Direct Sum Decomposition

In this section, we describe the formalization of the direct sum decomposition of finite abelian groups in Mizar. For convenience, we consider only decomposition into an internal direct sum, but it can be extended to an external direct sum case easily.

In Mizar, the direct sum decomposition is defined as follows:

Definition 11 (Internal direct sum decomposition in Mizar).

```
definition
  let G be finite commutative Group;
  let I be non empty finite set;
  let F be commutative Group-Family of I;
  pred G is_direct_sum_of_subgroup_family F means
    (for i be Element of I holds F.i is Subgroup of G)
    & (for i,j be Element of I st i <> j holds
     (the carrier of (F.i)) /\ (the carrier of (F.j)) = {1_G})
    & (for y be Element of G
      ex x be (the carrier of G)-valued total I-defined Function
      st (for i be Element of I holds x.i in F.i) & y = Product x)
    & (for x1,x2 be (the carrier of G)-valued total I-defined Function
      st (for i be Element of I holds x1.i in F.i)
        & (for i be Element of I holds x2.i in F.i)
        & (Product x1 = Product x2)
      holds x1 = x2);
end;
```

We now explain Definition 11 in detail. **the carrier of G** denotes 'the set of all elements in group G', so **(the carrier of G)-valued total I-defined Function** implies 'Function from I to G'. The predicate in the definition satisfies the following conditions:

1. $\forall i \in I, F_i$ is subgroup of G.
2. $\forall i, j \in I, i \neq j \Rightarrow F_i \cap F_j = \{1_G\}$.
3. $\forall y \in G, \exists x \in \bigoplus_{i \in I} F_i$ such that $\prod_{i \in I} x_i = y$ as element of G.
4. $\forall x_1, x_2 \in \bigoplus_{i \in I} F_i, \prod_{i \in I} x_{1i} = \prod_{i \in I} x_{2i} \Rightarrow x_1 = x_2$.

Conditions 3 and 4 imply the existence and uniqueness of the internal direct sum representation. Finally, it is interpreted as follows:

Definition 12 (Internal direct sum decomposition of finite abelian groups). *Let G be a finite abelian group having subgroups $F_1, \cdots, F_n$. If the following condition is satisfied, $\{H_i\}$ is called an internal direct sum decomposition of G, and the conditions are denoted by $G = F_1 \oplus \cdots \oplus F_n$.*

1. *$i \neq j \Rightarrow F_i \cap F_j = \{1_G\}$.*
2. *Each $y \in G$ has a unique expression of the form $y = x_1 \cdots x_n$ where $x_i \in F_i$.*

4 Decomposition into Subgroups of Prime Power Order

In this section, we formalize the direct sum decomposition of a finite abelian group into finite abelian subgroups of prime power order in Mizar.

We first formalize a simple case; the decomposition of a finite abelian group into the direct sum of two groups.

Theorem 1 (Decomposition into two groups in Mizar).

```
theorem :: GROUP_17:16
  for G being finite commutative Group,
    h,k be Nat
  st card G = h*k
    & h,k are_coprime holds
  ex H,K being strict finite Subgroup of G
  st the carrier of H =
       {x where x is Element of G: x|^h = 1_G}
    & the carrier of K =
       {x where x is Element of G: x|^k = 1_G}
    & H is normal & K is normal
    & (for x be Element of G holds
       ex a,b be Element of G
       st a in H & b in K & x = a*b)
    & (the carrier of H) /\ (the carrier of K) = {1_G};
```

From Theorem 1, we notice that the relationship among G, H, and K satisfies the conditions of direct sum decomposition formalized in Definition 11. Thus Theorem 1 shows that a direct sum is decomposable into two nontrivial groups if the original abelian group order is expressed as a product of two co-prime integers. The essence of the theorem is described as follows:

Theorem 2 (Decomposition into two groups). *Let G is a finite abelian group with $|G| = h \cdot k$ where h is coprime to k. Then there exist H and K that are subgroups of G where $|H| = h, |K| = k$ and satisfy $G = H \oplus K$.*

Recursive application of Theorem 1 leads to the following theorem.

Theorem 3 (Decomposition into abelian groups of prime power order in Mizar).

```
theorem :: GROUP_17:35
  for G being strict finite commutative Group
    st card G > 1 holds
  ex I be non empty finite set,
    F be commutative Group-Family of I
  st I = support (prime_factorization card G)
    & (for p be Element of I holds
       F.p is strict Subgroup of G
       & card (F.p) = (prime_factorization card G).p)
    & (G is_direct_sum_of_subgroup_family F);
```

We now explain the terms in Theorem 3.

The function **prime_factorization** satisfies

$$\text{prime\_factorization}(m)(p_i) = {p_i}^{r_i}$$

where $m = {p_1}^{r_1}{p_2}^{r_2} \cdots {p_n}^{r_n}$, p_i is prime, and r_i is a positive integer.

support is an attribute that attaches to a function, where a function is a subset of the function domain containing non-zero function values. Therefore, in Theorem 3.2, **I** denotes a set of prime divisors of $|G|$, where $|G|$ is the number of elements in group G.

It should be noted the reason why theorem excludes a trivial case by **card G > 1**. Supposing **card G = 1**, then **I = support(prime_factorization card G)** becomes empty, because 1 is not included in prime numbers. It is not suited for our **Group-Family** definition.

In conclusion, Theorem 3 encodes the following theorem:

Theorem 4. *Let G be a finite abelian group where $|G| = {p_1}^{r_1}{p_2}^{r_2}\cdots{p_n}^{r_n}$, $\forall i$, p_i is prime and r_i is a positive integer. Then there exists a family of subgroups $\{F_{p_i}\}_{p_i \in I}$ in G that satisfies $G = F_{p_1} \oplus F_{p_2} \oplus \cdots \oplus F_{p_n}$ and $\forall i$, $|F_{p_i}| = {p_i}^{r_i}$.*

5 Decomposition into Cyclic Subgroups of Prime Power Order

In this section, we formalize the direct sum decomposition of a finite abelian group of prime power order into cyclic subgroups of prime power order in Mizar. Similarly, the theorem is formalized by mathematical induction.

First, we define a function **Ordset** for a finite group.

Definition 13 (Ordset in Mizar).

```
definition :: GROUP_18:def 1
  let G be finite Group;
  func Ordset G -> Subset of NAT equals
    the set of all ord a where a is Element of G;
end;
```

That definition is translated into natural language as follows:

Definition 14 (Ordset). *Let G is a finite group. then* **Ordset G** *is a set of the orders of all elements in G.*

Since **Ordset G** is a finite set of natural numbers, it contains a unique upper bound. We define this upper bound as **upper_bound Ordset G**.

Then we formalize the theorem by which a cyclic group factor is extracted from a finite abelian group of prime power order in Mizar.

Theorem 5 (Extraction of a cyclic group factor in Mizar).

```
theorem :: GROUP_18:19
  for G being strict finite commutative Group,
    p being Prime, m be Nat st card(G) = p|^m
  ex K be normal strict Subgroup of G,
    n, k be Nat,
    g be Element of G
  st
    ord g = upper_bound Ordset G
    & K is finite commutative
    & (the carrier of K) /\ (the carrier of gr{g}) = {1_G}
```

```
  & (for x be Element of G holds
     ex b1, a1 be Element of G
     st b1 in K & a1 in gr{g} & x = b1*a1)
  & ord g = p|^n
  & k = m - n & n <= m
  & card K = p|^k
  & ex F being Homomorphism of product <*K,gr{g}*>, G
    st F is bijective
  & for a,b be Element of G
    st a in K & b in gr{g}
    holds F.(<*a,b*>) = a*b;
```

The essential aspects of Theorem 5 are listed below.

1. One of the largest order elements in G is selected and labeled **g**.
2. A cyclic subgroup, **gr{g}** is generated by the group element **g**.
3. A subgroup of G, labeled K, satisfies $G = K \oplus \langle g \rangle$.

So the essence of the theorem is described as follows:

Theorem 6 (Extraction of a cyclic group factor). *Let G is a finite abelian group.*

1. *There exists an element $g \in G$ with the largest order.*
2. *And there exists K being a subgroup of G that satisfies $G = \langle g \rangle \oplus K$.*

Finally, Theorem 7 is formalized by recursive application of Theorem 5.

Theorem 7 (Decomposition into cyclic subgroups of prime power order in Mizar).

```
theorem :: GROUP_18:21
  for G being strict finite commutative Group,
    p being Prime, m be Nat
  st card(G) = p|^m
  holds
  ex k be non zero Nat,
    a be k-element FinSequence of G,
    Inda be k-element FinSequence of NAT,
    F be commutative Group-Family of Seg k
  st (for i be Nat st i in Seg k holds
      ex ai be Element of G
      st ai = a.i & F.i = gr{ai}
        & ord(ai) = p|^(Inda.i))
    & (for i be Nat st 1 <= i & i <= k-1
       holds Inda.i <= Inda.(i+1))
    & G is direct_sum_ob_subgroup_family F;
```

Theorem 7 can be translated into the following.

Theorem 8 (Decomposition into cyclic subgroups of prime power order). *Let G be a finite abelian group of prime power order where $|G| = p^m$, p is prime and m is a positive integer. Then there exist k-element sequences $\{a_i\}$ and $\{r_i\}$ that satisfy the following conditions:*

1. *$a_i \in G$ and $r_i \in \mathbb{N}$.*
2. *$|\langle a_i \rangle| = p^{r_i}$*
3. *$\{r_i\}$ is a monotonically increasing sequence.*

4. $G = \langle a_1 \rangle \oplus \cdots \oplus \langle a_k \rangle$

In other words, we can regard $\{a_i\}$ as a basis of G.

We should mention that $\{r_i\}$ need not be a monotonically increasing sequence, but the sequence has been ordered for the next step.

6 Fundamental Theorem of Finite Abelian Groups

In this section, we finally formalize the fundamental theorem of finite abelian groups in Mizar. A theorem that describes two-level direct summation can be expressed as a one-level direct sum.

Theorem 9 (Two-level direct sum flattening in Mizar).

```
theorem
  for G be finite commutative Group,
    I be non empty finite set,
    F be commutative Group-Family of I,
    II be ManySortedSet of I,
    FF be ManySortedSet of I
  st  (G is_direct_sum_of_subgroup_family F)
    & (for i be Element of I
       ex Fi be finite commutative Group,
         IIi be non empty finite set,
         FFi be commutative Group-Family of IIi
       st Fi = F.i & IIi = II.i & FFi = FF.i
        & Fi is_direct_sum_of_subgroup_family FFi)
    & (for i, j be Element of I st i <> j holds
       II.i misses II.j)
  holds
    ex Ix be non empty finite set,
       Fx be commutative Group-Family of Ix
    st Ix = Union II
      & Fx = Union FF
      & G is_direct_sum_of_subgroup_family Fx;
```

Essentially, Theorem 9 means an extension of associativity of direct sum $H_1 \oplus (H_2 \oplus H_3) = (H_1 \oplus H_2) \oplus H_3$. It states that

Theorem 10 (Two-level direct sum flattening). *Assume*

1. *G is a finite abelian group.*
2. *$\{F_i\}_{i \in I}$ is a family of subgroups of G such that $G = \bigoplus_{i \in I} F_i$.*
3. *$\{H_{i,j}\}_{(i,j) \in K}$ is a family of subgroups of G where $K = \bigcup_{i \in I} \bigcup_{j \in J_i} (i,j)$ such that $F_i = \bigoplus_{j \in J_i} H_{i,j}$.*

Then

$$G = \bigoplus_{(i,j) \in K} H_{i,j}.$$

To formalize the fundamental theorem, we apply Theorem 9 to Theorem 3 and Theorem 7.

Theorem 11 (Primary decomposition form in Mizar).

```
theorem
  for G be strict finite commutative Group
    st card G > 1 holds
  ex I be non empty finite set,
    F be commutative Group-Family of I
  st (G is_direct_sum_of_subgroup_family F)
    & (for i be Element of I holds
       F.i is cyclic & F.i is prime_power_order);
```

This theorem can be interpreted as follows:

Theorem 12 (Primary decomposition form). *Every finite abelian group G is a direct sum of finite cyclic groups of prime power order. That means G can be written as follow:*

$$G = \mathbb{Z}_{q_1} \oplus \mathbb{Z}_{q_2} \oplus \cdots \oplus \mathbb{Z}_{q_t},$$

where the numbers $q_1, \cdots, q_t$ are powers of prime numbers.

We also formalize the second expression of the fundamental theorem of finite abelian groups.

Theorem 13 (Existence of invariant factor decomposition in Mizar).

```
theorem
  for G being finite commutative Group st card G > 1
  holds
  ex n be non zero Nat,
    b be Function of Seg n, the carrier of G,
    F be commutative Group-Family of Seg n
  st
    (G is_direct_product_of_subgroup_family F)
    & (for i be Element of Seg n holds
       F.i = gr {b.i} & ord (b.i) > 1)
    & (for i,j be Element of (Seg n) st j=i+1 holds
       ord (b.i) divides ord (b.j));
```

Theorem 14 (Uniqueness of invariant factor decomposition in Mizar).

```
theorem
  for G being finite commutative Group,
    n0, n1 be non zero Nat,
    b0 be Function of Seg n0, the carrier of G,
    b1 be Function of Seg n1, the carrier of G,
    F0 be commutative Group-Family of (Seg n0),
    F1 be commutative Group-Family of (Seg n1)
  st (G is_direct_product_of_subgroup_family F0)
    & (G is_direct_product_of_subgroup_family F1)
    & (for i be Element of (Seg n0) holds
       F0.i = gr {b0.i} & ord (b0.i) > 1)
    & (for i be Element of (Seg n1) holds
       F1.i = gr {b1.i} & ord (b1.i) > 1)
```

```
    & (for i,j be Element of (Seg n0) st j=i+1 holds
       ord (b0.i) divides ord (b0.j))
    & (for i,j be Element of (Seg n1) st j=i+1 holds
       ord (b1.i) divides ord (b1.j))
  holds
    n0 = n1
  & (for i0 be Element of Seg n0,
        i1 be Element of Seg n1
     st i0 = i1 holds
     ord (b0.i0) = ord (b1.i1));
```

From Theorem 13 and Theorem 14, invariant factor decomposition form is obtained. Before that, we give a definition of invariant factors.

Definition 15 (Invariant factors). *If a finite abelian group G has a decomposition as a direct sum*

$$G = \mathbb{Z}_{m_1} \oplus \mathbb{Z}_{m_2} \oplus \cdots \oplus \mathbb{Z}_{m_t},$$

where $m_1|m_2|\cdots|m_t$ is satisfied. Then one says that G has invariant factors $(m_1, \cdots, m_t)$.

Invariant factor decomposition form can be expressed as follows:

Theorem 15 (Invariant factor decomposition form). *Every finite abelian group G has a invariant factor decomposition and its invariant factors are uniquely determined. That means,*

$$G = \mathbb{Z}_{m_1} \oplus \mathbb{Z}_{m_2} \oplus \cdots \oplus \mathbb{Z}_{m_t},$$

where $m_1|m_2|\cdots|m_t$ and the invariant factors $(m_1, \cdots, m_t)$ are uniquely determined by G.

The theorem gives us a complete classification for finite abelian groups. Once an order is determined, we can easily count the number of finite abelian groups by using the property of invariant factors. For example, let us think of the classification of abelian groups with order $72 = 3^2 \cdot 2^3$. Then the combination of invariant factors can be listed up as bellow:

1. $\mathbb{Z}_{72}$ $(2^3 \cdot 3^2)$
2. $\mathbb{Z}_{36} \oplus \mathbb{Z}_2$ $(2^2 \cdot 3^2, 2^1)$
3. $\mathbb{Z}_{18} \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$ $(2^1 \cdot 3^2, 2^1, 2^1)$
4. $\mathbb{Z}_{24} \oplus \mathbb{Z}_3$ $(2^3 \cdot 3^1, 3^1)$
5. $\mathbb{Z}_{12} \oplus \mathbb{Z}_6$ $(2^2 \cdot 3^1, 2^1 \cdot 3^1)$
6. $\mathbb{Z}_6 \oplus \mathbb{Z}_6 \oplus \mathbb{Z}_2$ $(2^1 \cdot 3^1, 2^1 \cdot 3^1, 2^1)$

Let us briefly outline the proof of Theorem 15. From Theorem 11, the following lemma is obtained by adding trivial groups to the direct sum factors as required.

Lemma 1. *Let G be a finite abelian group. Then there exists a double-indexed family of cyclic groups $\{H_{i,j}\}_{1\leq i\leq r, 1\leq j\leq s}$ and a sequence of primes $\{p_i\}_{1\leq i\leq r}$ that satisfies the following conditions:*

1. *$H_{i,j}$ is a subgroup of G.*
2. *$H_{i,j}$ is p_i-power order or trivial.*
3. *$1 \leq \forall j \leq s-1$, $|H_{i,j}| \leq |H_{i,j+1}|$.*
4. *$G = \bigoplus_{1\leq i\leq r, 1\leq j\leq s} H_{i,j}$*

Defining $F_i = \bigoplus_{1\leq j\leq r} H_{i,j}$, we obtain $G = F_1 \oplus \cdots \oplus F_r$ and can conclude Theorem 15.

7 Conclusions and Future Work

In this paper, we formalized the fundamental theorem of finite abelian groups in Mizar. We also defined the direct sum of finite abelian groups and several patterns of direct sum decomposition by which we proved the targeted theorem. Since we have formalized an essential algebraic principle, the result is applicable to a variety of future formalizations.

As the next step, we aim to formalize the extended case for finitely generated abelian groups in Mizar. We also have to note that the current definition of internal direct sum decomposition can only be applied to finite abelian groups. We are now trying to formalize a new definition of both internal and external direct sum decomposition of groups in Mizar. The new definition will be more natural and comprehensive for group theory.

References

1. Grabowski, A., Kornilowicz, A., Naumowicz, A.: Mizar in a Nutshell. Journal of Formalized Reasoning 3(2), 153–245 (2010)
2. Bandini, S., Federici, M. L., Vizzari, G.: An Overview of the Mizar Project. Cybernetics and Systems: An International Journal, 38(7), 729–753 (2007)
3. Muzalewski, M.: An Outline of PC Mizar. Fondation Philippe le Hodey, Brussels. (1993)
4. Robinson, D.: A Course in the Theory of Groups, Vol. 80. Springer. 90–116 (1996)
5. Rotman, J. J.: An introduction to the theory of groups, Vol. 148. Springer. 125–133 (1995)
6. Trybulec, W. A.: Groups. Formalized Mathematics, 1(5), 821–827 (1990)
7. Trybulec, W. A.: Subgroup and Cosets of Subgroups. Formalized Mathematics, 1(5), 855–864 (1990)
8. Surowik, D.: Cyclic Groups and Some of Their Properties – Part I. Formalized Mathematics, 2(5), 623–627 (1991)
9. Okazaki, H., Yamazaki, H. and Shidama, Y.: Isomorphisms of Direct Products of Finite Commutative Groups. Formalized Mathematics. 21(1), 65–74 (2013)
10. Okazaki, H., Yamazaki, H., Nakasho, K and Shidama, Y.: Isomorphisms of Direct Products of Cyclic Groups of Prime Power Order. Formalized Mathematics. 21(3), 207–211 (2013)
11. Wiedijk, F.: Writing a Mizar article in nine easy steps. `http://mizar.org/project/mizman.pdf`.

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Budget Imbalance Criteria for Auctions: A Formalized Theorem*

Marco B. Caminati[1], Manfred Kerber[1], and Colin Rowat[2]

[1] School of Computer Science, University of Birmingham, UK
[2] Economics, University of Birmingham, UK
Project homepage: `http://cs.bham.ac.uk/research/projects/formare/`

Abstract. We present an original theorem in auction theory: it specifies general conditions under which the sum of the payments of all bidders is necessarily not identically zero, and more generally not constant. Moreover, it explicitly supplies a construction for a finite minimal set of possible bids on which such a sum is not constant. In particular, this theorem applies to the important case of a second-price Vickrey auction, where it reduces to a basic result of which a novel proof is given. To enhance the confidence in this new theorem, it has been formalized in Isabelle/HOL: the main results and definitions of the formal proof are reproduced here in common mathematical language, and are accompanied by an informal discussion about the underlying ideas.

1 Introduction

The ForMaRE project [4] employs formal methods to provide a unified approach to both the generation of executable code for running auctions and the proving of theorems about them. In this paper, we will describe the formalization of a classical result about the second price Vickrey auction (which will be introduced in Section 2), stating that the sum of the payments for all participants cannot be zero for all possible bids. We will indicate this result as NB (for *n*o *b*alance).

The proof mechanism we present for NB is, to the best of our knowledge, new. Although it is also applicable to the specific case of the Vickrey auction, our proof works for a wide class of possible auction mechanisms: indeed, it gives a characterization of the kinds of auctions for which it holds. By contrast, all the existing proofs we know of vitally rely on the particular algebraic form that the mechanism assumes in the particular case of the Vickrey auction. Furthermore, our proof explicitly constructs a minimal, finite set of possible bids on which the sum of all the payments is not a constant function.

All the results have been formalized and checked in the Isabelle/HOL theorem prover [7]. Because the results are new, they are stated here in common mathematical language, which should be friendlier for the reader. The lemmas, definitions, and theorems in this paper correspond as far as possible their formalized counterparts and for each statement we indicate the corresponding Isabelle name in typewriter font. The relevant Isabelle theory file is `Maskin3.thy`, available at `https://github.com/formare/auctions/`.

* This work has been supported by EPSRC grant EP/J007498/1 and an LMS Computer Science Small Grant.

1.1 Structure of the paper

In Section 2, some context is given: we will see the basic mathematical definitions for auctions, which will be needed to state the main theorem NB, the object of the present formalization.

Section 3 informally illustrates the idea behind our original proof of NB, and Section 4 presents the formal result implementing this idea, which is Lemma 1. This lemma is the cornerstone of the whole formalization effort presented here: all the other results depend on it.

This fact is illustrated in Section 5, where it is informally explained how Lemma 1 can be applied to the particular case of the Vickrey auction.

This informal explanation is then made formal and systematic in Section 6, where ancillary lemmas and definitions are given in order to formally derive from Lemma 1 the main result, Theorem 1, which is the formal statement of our generalized version of NB.

1.2 Notations

- We will represent any function (e.g., the one associating to each participant her bid) in a set-theoretical fashion; that is, as the set $\{(x, f(x)) \,|x \in \operatorname{dom} f\}$, commonly called the graph of f. Hence, for example, the cartesian product $X \times \{y\}$ is the constant function defined on X and yielding y.
- Similarly, any relation will be represented as the set of the pairs of elements related through it; formally, this means that any relation is a subset of some cartesian product $X \times Y$.
- Given a relation R, $R(X)$ will be the image of the set X through R: $R(X) := \{y| (x, y) \in R \text{ and } x \in X\}$. For example, given a function f, $f^{-1}(\{y\})$ will then be the fiber of the point y under f.
- The restriction of a function f to the set-theoretical difference $\operatorname{dom} f \backslash X$ will be written $f - X$; moreover, in the special case of $X = \{x\}$, we will often abuse the notation, writing $f - x$ instead of $f - \{x\}$.
- A multiset (which is also called a bag) will be extensionally denoted writing, e.g., $\{|0, 0, 1, 1, 1, 2|\}$: we recall that a multiset is similar to a set, but each member has a multiplicity describing how many times it appears in the multiset. We will use $+$ as the infix symbol for pairwise multiset union; we will write $A \leqslant B$ to express the fact that A is a sub-multiset of B: for instance, $\{|2, 1, 1|\} \leqslant \{|0, 0, 1, 1, 1, 2|\}$ is true.
- Finally, $x \rightharpoonup X$ denotes the operation of union of x with each set in the family of sets X:

$$x \rightharpoonup X := \left\{x \cup x' : x' \in X\right\}.$$

  We will need this operation to state Lemma 1.

2 Statement of the main result

An auction mechanism is mathematically represented through a pair of functions a, p: the first describes how the goods at stake are allocated among the bidders (also called participants or agents), while the second specifies how much each bidder pays following

this allocation. Each possible output of this pair of functions is referred to as an *outcome* of the auction. Both functions take the same argument, which is another function, commonly called a *bid vector*; it describes how much each bidder prefers each possible outcome of the auction. This preference relation is usually expressed through money, hence the bid vector associates some outcome to how much the participant values that outcome. To stick with traditional notation, we will use bold face for bid vectors, as in $a(\boldsymbol{b})$.

In the case of a single good being auctioned, the bid vector simply associates to each bidder the amount she bids for the good. Given a fixed bidder i, moreover, a_i is $\{0,1\}$-valued, corresponding to the fact that i either wins the item or not. For a single good auction, the Vickrey mechanism has a special relevance because of the formal properties it enjoys [5], [3]. It works by assigning the good to the highest bidder. Each agent then pays a 'fee' term $p'_i(\boldsymbol{b}-i)$ irrespective of the outcome; this fee does not depend on how much i herself bids, but only on the other participants' bids: hence the argument of p'_i is $\boldsymbol{b}-i$ rather than the full $\boldsymbol{b}$. Additionally, the winner pays a proper price term given by the highest of the bids computed on the set of participants excluding the winner herself (second price); given a fixed bid vector $\boldsymbol{b}$, we will denote this amount as $f_2(\boldsymbol{b})$.

An often desirable property of an auction mechanism is that it achieves *budget balancing* [6, Section 2.3]. This means that the sum of the money given or received by each participant *always* totals to zero:

$$\forall \boldsymbol{b} \qquad \sum_i p_i(\boldsymbol{b}) = 0. \tag{1}$$

Such a property becomes attractive when

> it is preferable to maintain as much wealth as possible within the group of agents, and the transfer payment can be viewed as an undesirable "cost of implementation". [1]

There are important cases in which (1) assumes a more specific form:[3]

$$\forall \boldsymbol{b} \qquad \overline{f}(\boldsymbol{b}) + \sum_i p'_i(\boldsymbol{b}-i) = 0, \tag{2}$$

where $\overline{f}$ typically extracts some kind of maximum: e.g., for the single-good Vickrey mechanism, $\overline{f}(\boldsymbol{b})$ is the second price $f_2(\boldsymbol{b})$. The function p'_i is related to p_i through a simple construction we are not interested in here. Here, the important fact is the formal difference between p_i and p'_i: the former takes as an argument the full bid vector, while the latter takes a *reduced* bid vector, in which the bid pertaining participant i has been removed.

A standard result [6, Theorem 2.2], [5, Proposition 3], [2, Theorem 4.1] is that for such cases, budget balancing cannot be satisfied: (1) is false. Its known proofs, however, all exploit the particular form of the map $\overline{f}$ appearing in (2) in the considered case, namely that of $\overline{f}$ being f_2. Vice versa, we will see a proof starting from (2) where the latter map is considered as an unknown; the proof will work out the conditions it needs to impose on that map: only after that we will ascertain they are fulfilled for the given cases we are interested in (e.g., the mentioned single-good Vickrey auction). To be even

[3] We recall that bid vectors are modeled as functions, hence we can write $\boldsymbol{b}-i$, using the notation introduced in Section 1.2.

more informative, the proof will show that to falsify equation (2), it is not needed to quantify it over all the possible $\boldsymbol{b}$ admissible as an argument to $\overline{f}$: a smaller, finite set X will be suggested by the proof itself.

Hence, we will consider the logical formula

$$\forall \boldsymbol{b} \in X \qquad \sum_i p'_i(\boldsymbol{b} - i) = f(\boldsymbol{b}) \tag{3}$$

and study for what X and f it leads to a contradiction (which will include, of course, our starting case (2), where $f = -\overline{f}$). Since we are going to set up a proof mechanism minimizing the requirements on the generic f (e.g., we are not going to expect that f is the maximum or the second maximum of the bids), we must impose some different premises to carry through a proof. The main assumption needed is one of symmetry: while the p_is in (1) (and hence the p'_is in (2)) are completely arbitrary, we need to require that they do not depend on re-labeling the participants:

$$\exists P \qquad \forall i\ \boldsymbol{b} \qquad p'_i(\boldsymbol{b}) = P(\|\boldsymbol{b}\|), \tag{4}$$

where $\|R\|$ is the *multiset* (or *bag*) obtained from the relation R: that is, the range of R, but taking into account the multiplicities of possibly repeated values in it.[4] This means that the price paid by any participant i is given by a rule depending only on the amount of each bid other than i's, and not on *who* placed those. Moreover, such a rule itself must not depend on i.

With this assumption in place, (3) becomes

$$\forall \boldsymbol{b} \in X \qquad \sum_i P(\|\boldsymbol{b} - i\|) = f(\boldsymbol{b}). \tag{5}$$

3 Proof idea

Let N be the number of bidders. The proof starts by considering the case that they all submit the same (fixed but arbitrary) bid b_0, whence (5) yields:

$$P\left(\left\{\left|\underbrace{b_0, \ldots, b_0}_{N-1}\right|\right\}\right) = k_0 f\left(\underbrace{b_0, \ldots, b_0}_{N}\right), \tag{6}$$

with $k_0 := \frac{1}{N}$ not depending on b_0. Then we continue with a $\boldsymbol{b}$ in which only one component (let us say the first, for example) assumes an arbitrary value b_1 different than b_0; thus, (5) gives

$$(N-1)P\left(\left\{\left|b_1, \underbrace{b_0, \ldots, b_0}_{N-2}\right|\right\}\right) = -P\left(\left\{\left|\underbrace{b_0, \ldots, b_0}_{N-1}\right|\right\}\right) - f(b_1, b_0, \ldots, b_0). \tag{7}$$

At this point, we would like to trigger an iterative mechanism by exploiting (6) inside (7). A natural imposition to make this possible is to ask that

$$f(b_1, b_0, \ldots, b_0) = q_1 f(b_0, \ldots, b_0) \tag{8}$$

[4] For example, given the map associating to participant 1 the bid 10, to participant 2 the bid 20, and to participant 3 the bid 10, the obtained multiset is $\{|10, 10, 20|\}$.

for some arbitrary constant q_1. Then we can substitute (6) inside (7), obtaining a rational number k_1 not depending on b_0, b_1 such that

$$P\left(\left\{\left|b_1, \underbrace{b_0, \ldots, b_0}_{N-2}\right|\right\}\right) = k_1 f\left(b_0, \ldots, b_0\right). \tag{9}$$

Proceding the same way, we can build a rational constant k_2 such that

$$P\left(\left\{\left|b_1, b_2, \underbrace{b_0, \ldots, b_0}_{N-3}\right|\right\}\right) = k_2 f\left(b_0, \ldots, b_0\right) \tag{10}$$

for arbitrary b_1, b_2.

So that by iterating this mechanism we can construct a relation binding the generic $P\left(\{|b_1, \ldots, b_{N-2}, b_0|\}\right)$ to $f\left(b_0, \ldots, b_0\right)$:

$$P\left(\{|b_1, \ldots, b_{N-2}, b_0|\}\right) = k_{N-2} f\left(b_0, \ldots, b_0\right). \tag{11}$$

Moreover, at each step of this iteration, the requirement (8) gives indications about how $q_1, q_2, \ldots$, X and f must be related if we want the proof mechanism to work. It is important to note that, in doing so, such mutual relations should be weakened as much as possible, with the only rationale that they should grant that the proof mechanism just outlined actually works. For example, we imposed one equality of the kind (8) at each inductive step, but these equalities actually need to hold only for the bid vectors inside some minimal X; otherwise, we would restrict ourselves to quite trivial instances of f. In this section, we wanted to give a general idea of the proof, and we did not explicitly state the exact mutual relations between b_0, X and f. Indeed, such relations are not immediate, at first: they actually became clearer when formalizing the proof itself; a process that we feel would have been harder to manage with a standard paper-based proof. These relations will be given in full detail in Section 4, in Definition 1.

The iteration explained above implies that we assign some value to the numbers $q_1, q_2, \ldots$. We deemed them arbitrary because the proof mechanism works whichever values we assign them. For simplicity, however, we restricted our work to the case

$$1 = q_1 = q_2 = \ldots, \tag{12}$$

which will be general enough for any application to auction theory.

The idea is now to take equation (11), which was obtained using equation (5), and to derive a contradiction between it and (5) itself. Hence, the formalization can be seen as split in two stages: there is Lemma 1, presented in Section 4, which formalizes equation (11) and takes care of spelling out all the exact requirements in order to derive it exploiting the idea we informally presented above. Then there are other auxiliary definitions and lemmas, presented in Section 6, which employ Lemma 1 to obtain the wanted contradiction (given in the thesis of Theorem 1).

4 From the idea to the formal statement

Given a multiset m, an m-restriction of $\boldsymbol{b}$ is any $\boldsymbol{b}' \subseteq \boldsymbol{b}$ such that $\|\boldsymbol{b}'\| = m$.

An m-completion of $\boldsymbol{b}$ to b_0 is a $\boldsymbol{b}'$ writable as the disjoint union of an m-restriction of $\boldsymbol{b}$ with a function constantly valued b_0, and such that $\operatorname{dom} \boldsymbol{b} = \operatorname{dom} \boldsymbol{b}'$. In other

words, an m-completion of $\boldsymbol{b}$ to b_0 is obtained from $\boldsymbol{b}$ by changing its value to b_0 outside some m-restriction of it.

A full family of b_0-completions of $\boldsymbol{b}$ is a set containing one m-completion of $\boldsymbol{b}$ to b_0 for each possible $m \leqslant \|\boldsymbol{b}\|$.

Lemma 1 (`lll57`). *Consider a full family Y of b_0-completions of $\boldsymbol{b}$, and set $X := (\{i_1, i_2\} \times \{b_0\}) \rightharpoonup Y$ for some $i_1 \neq i_2$ outside $\operatorname{dom} \boldsymbol{b}$. Assume that, $\forall \boldsymbol{b}' \in X$:*

$$f\left(\boldsymbol{b}'\right) = f\left(\left(\{i_1, i_2\} \cup \operatorname{dom} \boldsymbol{b}\right) \times \{b_0\}\right) \tag{13}$$

$$f\left(\boldsymbol{b}'\right) = \sum_{i \in \operatorname{dom} \boldsymbol{b}'} P\left(\boldsymbol{b}' - i\right). \tag{14}$$

Then

$$P\left(\|\boldsymbol{b}\| + \{|b_0|\}\right) = \frac{1}{2 + |\operatorname{dom} \boldsymbol{b}|} f\left(\left(\{i_1, i_2\} \cup \operatorname{dom} \boldsymbol{b}\right) \times \{b_0\}\right).$$

For later discussion, it will be useful to express requirements in Lemma 1 up to equality (13) in a predicate form:

Definition 1. *[`pred2, pred3`] The set X is* adequate *for the quintuple $(\boldsymbol{b}, b_0, f, i_1, i_2)$ if*

- *$X = \{i_1, i_2\} \rightharpoonup Y$ for some Y being a full family of b_0-completions of $\boldsymbol{b}$;*
- *$\forall \boldsymbol{b}' \in X \quad f\left(\boldsymbol{b}'\right) = f\left(\left(\{i_1, i_2\} \cup \operatorname{dom} \boldsymbol{b}\right) \times \{b_0\}\right)$.*

This allows us to restate Lemma 1 as

Lemma 2 (`lll57`). *Assume that X is adequate for the quintuple $(\boldsymbol{b}, b_0, f, i_1, i_2)$, and that*

$$\forall \boldsymbol{b}' \in X \qquad \sum_{i \in \operatorname{dom} \boldsymbol{b}'} P\left(\boldsymbol{b}' - i\right) = f\left(\boldsymbol{b}'\right).$$

Then

$$P\left(\|\boldsymbol{b}\| + \{|b_0|\}\right) = \frac{1}{2 + |\operatorname{dom} \boldsymbol{b}|} f\left(\left(\{i_1, i_2\} \cup \operatorname{dom} \boldsymbol{b}\right) \times \{b_0\}\right).$$

5 Example application to the Vickrey auction

Let us consider the specific case of $f = -f_2$: we recall that $f_2\left(\boldsymbol{b}\right)$ is the maximum of the bids $\boldsymbol{b}$, once the bid of the winner has been removed. In this case, choosing any $b_0 \geqslant \max(\operatorname{rng} \boldsymbol{b})$ satisfies hypothesis (13) of Lemma 1, permitting

$$P\left(\|\boldsymbol{b}\| + \{|b_0|\}\right) = \frac{-b_0}{2 + |\operatorname{dom} \boldsymbol{b}|}. \tag{15}$$

Let us apply this to two particular bid vectors:

$$\underline{\boldsymbol{b}} := (1, 2, \ldots, n, n+1, n+3) \qquad \overline{\boldsymbol{b}} := (1, 2, \ldots, n, n+2, n+3).$$

We get

$$f_2\left(\underline{\boldsymbol{b}}\right)+\sum_i P\left(\left\|\underline{\boldsymbol{b}}-i\right\|\right) \overset{(15)}{=} (n+1)-\left[\frac{(n+3)}{n+2}(n+1)\right]-\frac{n+1}{n+2}$$

$$\neq n+2-\left[\frac{(n+3)}{n+2}(n+1)\right]-\frac{n+2}{n+2} \overset{(15)}{=} f_2\left(\overline{\boldsymbol{b}}\right)+\sum_i P\left(\left\|\overline{\boldsymbol{b}}-i\right\|\right). \quad (16)$$

Thus, we have falsified (2) as an application of Lemma 1. To do that, we had to apply Lemma 1 $n+2$ times for the first equality of the chain above, and further $n+2$ times for its last equality. This corresponds to having imposed (5) on the sets

$$\left\{\{i,j\}\times\{n+3\}\rightharpoonup\mathcal{C}^{n+3}_{\underline{\boldsymbol{b}}-i-j}\right\}_{\substack{\underline{\boldsymbol{b}}(j)=n+3\\ i\in\operatorname{dom}\underline{\boldsymbol{b}}-\{j\}}} \quad (17)$$

$$\left(\underline{\boldsymbol{b}}^{-1}\{n+1,n+3\}\right)\times\{n+1\}\rightharpoonup\mathcal{C}^{n+1}_{\underline{\boldsymbol{b}}-\underline{\boldsymbol{b}}^{-1}\{n+1,n+3\}} \quad (18)$$

and on the sets

$$\left\{\{i,j\}\times\{n+3\}\rightharpoonup\mathcal{C}^{n+3}_{\overline{\boldsymbol{b}}-i-j}\right\}_{\substack{\overline{\boldsymbol{b}}(j)=n+3\\ i\in\operatorname{dom}\overline{\boldsymbol{b}}-\{j\}}} \quad (19)$$

$$\left(\overline{\boldsymbol{b}}^{-1}\{n+2,n+3\}\right)\times\{n+2\}\rightharpoonup\mathcal{C}^{n+2}_{\overline{\boldsymbol{b}}-\overline{\boldsymbol{b}}^{-1}\{n+2,n+3\}} \quad (20)$$

respectively. Above, we have indicated with $\mathcal{C}^b_{\boldsymbol{b}}$ a fixed, arbitrary full family of b-completions of $\boldsymbol{b}$. Hence, the union of the family of sets in (17), the union of that in (19), the sets in (18), (20), together with $\left\{\underline{\boldsymbol{b}},\overline{\boldsymbol{b}}\right\}$, form the wanted set X: we have a contradiction when imposing (5) on it.

6 Application to the general case

Formally, as a first step of what we did in Section 5, we apply Lemma 1 to each possible $\boldsymbol{b}-i$ appearing in (5), obtaining an equality for the sum featured there. The following result does exactly that and is an immediate corollary of Lemma 2:

Corollary 1 (`llI68`). *Assume that* $\forall i\in\operatorname{dom}\boldsymbol{b}$ *there are* j_i *and* X_i *such that*

$$j_i\in\operatorname{dom}\boldsymbol{b}\backslash\{i\}$$
$$X_i \text{ is adequate for}\left(\boldsymbol{b}-\{i,j_i\},\boldsymbol{b}\left(j_i\right),f,i,j_i\right).$$

Also assume that

$$\forall\boldsymbol{b}'\in\bigcup X_i \qquad \sum_{i\in\operatorname{dom}\boldsymbol{b}'}P\left(\left\|\boldsymbol{b}'-i\right\|\right)=f\left(\boldsymbol{b}'\right).$$

Then

$$\sum_{i\in\operatorname{dom}\boldsymbol{b}}P\left(\{|\boldsymbol{b}-i|\}\right)=\frac{1}{|\operatorname{dom}\boldsymbol{b}|}\sum_{i\in\operatorname{dom}\boldsymbol{b}}f\left(\operatorname{dom}\boldsymbol{b}\times\{\boldsymbol{b}\left(j_i\right)\}\right). \quad (21)$$

What we informally did in Section 5 was to find $\underline{\boldsymbol{b}}$, $\overline{\boldsymbol{b}}$ to each of which corollary 1 is applicable, but such that the maps $\underline{\eta}:i\mapsto f\left(\operatorname{dom}\underline{\boldsymbol{b}}\times\left\{\underline{\boldsymbol{b}}\left(\underline{j}_i\right)\right\}\right)$ and $\overline{\eta}:i\mapsto f\left(\operatorname{dom}\overline{\boldsymbol{b}}\times\left\{\overline{\boldsymbol{b}}\left(\overline{j}_i\right)\right\}\right)$ enjoy the following properties:

1. each assumes exactly two values, call them $\underline{v_1}\neq\underline{v_2}$ and $\overline{v_1}\neq\overline{v_2}$, respectively;

2. of these four values, exactly two are equal, let us say $\underline{v_1} = \overline{v_1}$, while $\underline{v_2} \neq \overline{v_2}$;
3. the sets $\underline{\eta}^{-1}\left\{\underline{v_1}\right\}$ and $\overline{\eta}^{-1}\left\{\underline{v_1}\right\}$ coincide: that is, the points on which $\underline{\eta}$ and $\overline{\eta}$ yield the same value are exactly the same.

These facts cause the occurrence of the two identical terms which can be cancelled in expression (16): they are put in square brackets there for clarity. This cancellation is fundamental, because it immediately allows to establish the inequality appearing there. Finding such $\underline{\boldsymbol{b}}$ and $\overline{\boldsymbol{b}}$ is particularly easy in the case of $f = f_2$, but the same mechanism works for a generic f, leading to a similar cancellation between the two sums

$$\sum_{i \in \operatorname{dom} \underline{\boldsymbol{b}}} f\left(\operatorname{dom} \underline{\boldsymbol{b}} \times \left\{\underline{\boldsymbol{b}}\left(\underline{j}_i\right)\right\}\right)$$

and

$$\sum_{i \in \operatorname{dom} \overline{\boldsymbol{b}}} f\left(\operatorname{dom} \overline{\boldsymbol{b}} \times \left\{\overline{\boldsymbol{b}}\left(\overline{j}_i\right)\right\}\right);$$

each of those sums is yielded by a separate application of corollary 1, in the right hand side of equation (21).

To formalize the requirements (1), (2), (3) appearing in the list above, we introduce the following definition:

Definition 2 (`counterexample`)**.** *The triple* $\left(\underline{\boldsymbol{b}}, \overline{\boldsymbol{b}}, h\right)$ *is a* counterexample *for the map* f *if* $\operatorname{dom} \underline{\boldsymbol{b}} = \operatorname{dom} \overline{\boldsymbol{b}}$ *and there is a map* g *such that*

$$h\colon \left(\operatorname{dom} \underline{\boldsymbol{b}}\right) \to \{f, g\}$$
$$\{0\} \subset \left\{f\left(\overline{\boldsymbol{b}}\right) - f\left(\underline{\boldsymbol{b}}\right), g\left(\overline{\boldsymbol{b}}\right) - g\left(\underline{\boldsymbol{b}}\right)\right\}.$$

This definition is devised exactly to formulate the following lemma, which is an easy arithmetical statement:

Lemma 3 (`lll69`)**.** *Assume that* $\left(\underline{\boldsymbol{b}}, \overline{\boldsymbol{b}}, h\right)$ *is a counterexample for* f*, and that* $2 \leqslant |\operatorname{dom} \underline{\boldsymbol{b}}| < +\infty$*. Then*

$$f\left(\underline{\boldsymbol{b}}\right) - \frac{1}{|\operatorname{dom} \underline{\boldsymbol{b}}|} \sum \left(h\left(i\right)\right)\left(\underline{\boldsymbol{b}}\right) \neq f\left(\overline{\boldsymbol{b}}\right) - \frac{1}{|\operatorname{dom} \overline{\boldsymbol{b}}|} \sum \left(h\left(i\right)\right)\left(\overline{\boldsymbol{b}}\right).$$

In turn, the lemma above can finally be combined with corollary 1 into the main theorem:

Theorem 1 (`tt01`)**.** *Assume that* $\left(\underline{\boldsymbol{b}}, \overline{\boldsymbol{b}}, h\right)$ *is a counterexample for* f*, with* $2 \leqslant |\operatorname{dom} \underline{\boldsymbol{b}}| < +\infty$*. Moreover, assume that*
$\forall i \in \operatorname{dom} \underline{\boldsymbol{b}}$ *there are* $\underline{j}_i$*,* $\underline{X}_i$*,* $\overline{j}_i$*,* $\overline{X}_i$ *satisfying*

$$\underline{j}_i \in \operatorname{dom} \underline{\boldsymbol{b}} \setminus \{i\}$$
$$\underline{X}_i \text{ is adequate for } \left(\underline{\boldsymbol{b}} - \left\{i, \underline{j}_i\right\}, \underline{\boldsymbol{b}}\left(\underline{j}_i\right), f, i, \underline{j}_i\right)$$
$$f\left(\operatorname{dom} \underline{\boldsymbol{b}} \times \left\{\underline{\boldsymbol{b}}\left(\underline{j}_i\right)\right\}\right) = \left(h\left(i\right)\right)\left(\underline{\boldsymbol{b}}\right)$$
$$\overline{j}_i \in \operatorname{dom} \overline{\boldsymbol{b}} \setminus \{i\}$$
$$\overline{X}_i \text{ is adequate for } \left(\overline{\boldsymbol{b}} - \left\{i, \overline{j}_i\right\}, \overline{\boldsymbol{b}}\left(\overline{j}_i\right), f, i, \overline{j}_i\right)$$

$$f\left(\operatorname{dom}\overline{\boldsymbol{b}} \times \left\{\overline{\boldsymbol{b}}\left(\bar{j}_i\right)\right\}\right) = \left(h\left(i\right)\right)\left(\overline{\boldsymbol{b}}\right).$$

Finally, suppose that

$$\forall \boldsymbol{b}' \in \bigcup_{i \in \operatorname{dom}\underline{\boldsymbol{b}}} \underline{X}_i \cup \overline{X}_i \qquad \sum_{k \in \operatorname{dom}\boldsymbol{b}'} P\left(\left\|\boldsymbol{b}' - k\right\|\right) = f\left(\boldsymbol{b}'\right).$$

Then

$$f\left(\underline{\boldsymbol{b}}\right) - \sum_{i \in \operatorname{dom}\underline{\boldsymbol{b}}} P\left(\left\|\underline{\boldsymbol{b}} - i\right\|\right) \neq f\left(\overline{\boldsymbol{b}}\right) - \sum_{i \in \operatorname{dom}\overline{\boldsymbol{b}}} P\left(\left\|\overline{\boldsymbol{b}} - i\right\|\right).$$

7 Conclusions

We developed a result characterizing imbalanced auction mechanisms. Both the theorem and the proof are new to the best of our knowledge, and we informally illustrated the idea behind the proof.

On the formal side, the proof has been implemented in the Isabelle/HOL theorem prover, which is especially important in this case, because the confidence added by the formalization is a significant soundness validation for any new result.

Given a rather general class of auction mechanisms, our theorem provides explicit conditions implying the imbalance condition, and in this sense it can also be regarded as a result in reverse game theory. Moreover, our theorem also explicitly constructs a finite set on which the imbalance condition holds: this can be exploited in concrete implementations to computationally check the imbalance condition only over that known, finite set.

The fact that the proof and the result are new also leaves open many avenues for possible generalizations and improvements. For example, assumption (12) was taken for the sake of simplicity, but less immediate assumptions are also possible. Similarly, Definition 2 is merely the simplest one granting that Lemma 3 holds: there are plenty of ways to achieve the same result, which can lead to different final requirements on f appearing in the statement of Theorem 1. Currently, ForMaRE is following these tracks to investigate further developments of possible interest to its application domain, auction theory.

References

1. Cavallo, R.: Optimal decision-making with minimal waste: Strategyproof redistribution of vcg payments. In: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. pp. 882–889. ACM (2006)
2. Laffont, J.J., Maskin, E.: A differential approach to dominant strategy mechanisms. Econometrica 48(6), 1507–1520 (September 1980)
3. Lange, C., Caminati, M.B., Kerber, M., Mossakowski, T., Rowat, C., Wenzel, M., Windsteiger, W.: A qualitative comparison of the suitability of four theorem provers for basic auction theory. In: Carette, J., Aspinall, D., Lange, C., Sojka, P., Windsteiger, W. (eds.) Intelligent Computer Mathematics: MKM, Calculemus, DML, and Systems and Projects 2013, pp. 200–215. No. 7961 in Lecture Notes in Artifical Intelligence, Springer-Verlag (2013)

4. Lange, C., Rowat, C., Kerber, M.: The formare project – formal mathematical reasoning in economics. In: Carette, J., Aspinall, D., Lange, C., Sojka, P., Windsteiger, W. (eds.) Intelligent Computer Mathematics. pp. 330–334. No. 7961 in LNCS, Springer (2013)
5. Maskin, E.: The unity of auction theory: Milgrom's master class. Journal of Economic Literature 42(4), 1102–1115 (December 2004), http://scholar.harvard.edu/files/maskin/files/unity_of_auction_theory.pdf
6. Milgrom, P.: Putting auction theory to work. Churchill lectures in economics, Cambridge University Press (2004)
7. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic. No. 2283 in LNCS, Springer (2002)

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Feasible Analysis of Algorithms with Mizar

Grzegorz Bancerek

Association of Mizar Users
Białystok, Poland
bancerek@mizar.org

Abstract. In this paper we present an approach to the analysis of algorithms with Mizar. It consists in the Mizar formalization of abstract concepts like algebra of instructions, execution function, termination and their substantiation in a model with integers as the only data type and in models with abstract data types. The proof of correctness of the algorithm Exponentiation by Squaring is described.
The approach is aimed at creating Mizar tools for verification of real programs, but primarily to be used in teaching.

1 Previous works

An analysis of algorithms (AofA) is always a very attractive task for developers of any computer-aided system for formal reasoning. It appears naturally at every stage of system development as developers are closely related to practical and theoretical problems in programming. Actually, it is one of the most common motivations behind developing many such systems. The verification of programs was also one of the motives for developing Mizar. The first documented uses of this system in that area concerned

- the proof of the correctness of a factorial computing program (Elżbieta Ramm and Edmund Woronowicz in 1978, [17]),
- the proof of the correctness of a list reversing program (Piotr Rudnicki and Włodzimierz Drabent in 1983, [18]).

The investigations in this area constitute also an important part of recent Mizar Mathematical Library, MML. The main stream concerns the theory of abstract computers started by Yatsuka Nakamura and Andrzej Trybulec [15]. It was developed in 66 of 1191 articles in MML. The base concept of this theory is AMI – Architecture Model for Instructions which is a Mizar type for random access stored program machines (RASP). Models of AMI which have been defined in MML compute over integers [15], finite sequences of integers [22], or elements of a ring [12]. The theory of macro-instructions developed for these models allowed to prove correctness, inter alia, of Euclid's Algorithm [20], Bubble Sort [8], Insertion Sort [5], Quick Sort [6], compiler of arithmetic expressions [4], etc.

Mathematical descriptions of algorithms which do not enroll abstract machines are also developed in MML. These investigations concern, inter alia, IDEA encryption/decryption algorithm [9], RSA algorithm [10], generic GCD algorithms [19], and graph algorithms [14] [7].

To start an AofA we need to introduce a mathematical apparatus allowing to denote algorithms as Mizar terms. In previous Mizar approaches algorithms were denoted in the following ways.

Direct description An algorithm is presented as a MIZAR formula describing each step and each loop in it (Euclid's Algorithm in [1], algebraic algorithms).

Procedural description An algorithm is presented as a MIZAR formula with MIZAR functors describing particular procedures and functions (Dijkstra and Prim Algorithms in [14], GCD in [19], Dijkstra Algorithm in [7]).

Meta-algebraization An algorithm is presented as a MIZAR term with MIZAR functors corresponding to constructions in programming languages like sequential composition or loops (composition of macro-instructions in [21], Euclid's Algorithm in [20]).

Algorithms denoted using these approaches are, however, rather inconvenient for further manipulations. It is difficult to describe and state their properties as they belong to a meta level. Finally, these approaches were developed for proving only partial correctness of algorithms. The complexity and termination is difficult to investigate in these approaches and it is hardly done.

2 Main ideas of the MIZAR formalization

The approach presented here can be considered as a classical approach. We decided to define a language for denoting algorithms as a mathematical object which can be investigated like, for example, a language for logic. Moreover, the language is defined independently from a machine.

Algorithms and their parts – instructions – are introduced as elements of a one-sorted universal algebra called an if-while algebra. The algebra has 4 operations: a constant – the empty instruction, a binary catenation of instructions (_;_), a ternary conditional instruction (*if* _ *then* _ *else* _), and a binary while instruction (*while* _ *do* _). An execution function is defined on pairs (s, I), where s is a state (an element of a certain set of states) and I is an instruction, and results in states. The execution function obeys control structures using the set of distinguished true states. I.e., a condition instruction is executed and the continuation of execution depends on the fact whether the resulting state is in true states, or not. Termination is also defined for pairs (s, I) and depends on the execution function. The existence of an execution function determined by elementary instructions and the uniqueness of it for terminating instructions is proved in the MIZAR formalization.

2.1 If-while algebra

The definition of an if-while algebra uses the type `Universal_Algebra` which is defined in [13] as a system

```
UAStr(# carrier -> set, charact -> PFuncFinSequence of the carrier #);
```

fulfilling some additional properties. It means that a universal algebra is an aggregate of two components: a set called `carrier` and a finite sequence of partial operations on the `carrier` called `charact`. If-while algebra is introduced in a few steps. First, a MIZAR attribute `with_empty_instruction` is defined to express a property of a universal algebra that its operation number 1 is nullary, (i.e., has arity zero).

```
definition
  let S be non empty UAStr;
```

```
  attr S is with_empty-instruction means
  1 in dom the charact of S &
  (the charact of S).1 is 0-ary non empty homogeneous
  quasi_total PartFunc of (the carrier of S)*, the carrier of S;
end;
```

The meaning of the definiens is that 1 is in the domain of the `charact` (so there is at least one element in the sequence of operations) and the value of the `charact` on 1 is a nullary operation. In other words, there exists the operation number 1 and it is nullary. In the same way we introduce attributes

- `with_catenation` – there exists the operation number 2 and it is binary,
- `with_if-instruction` – there exists the operation number 3 and it is ternary, and
- `with_while-instruction` – there exists the operation number 4 and it is also binary.

Then, `preIfWhileAlgebra` is introduced as a `Universal_Algebra` which possesses all 4 these properties.

```
definition
  mode preIfWhileAlgebra is with_empty-instruction
       with_catenation with_if-instruction with_while-instruction
       Universal_Algebra;
end;
```

Before this definition the existence of such an algebra had to be proved (*existential registration*). Of course, the best example is a free universal Ω-algebra where Ω is the signature of type $(0, 2, 3, 2)$ (see [16]) but the catenation operation in a free algebra is not associative and the result of the catenation of empty instruction is different from the argument. In our approach, we allow the if-while algebra to have some *compiler optimizations* (e.g., elimination of the empty instruction). The if-while algebra may have associative catenation (attribute `associative`) and may have the empty instruction as a neutral element of the catenation (attribute `unital`). However, some properties of free algebras are still welcome. For example, a conditional instruction could not be a loop instruction or the other way round; a conditional instruction could not be identical with its condition or sub-instruction, etc. This could be expressed with the adjective `non degenerated` which is an antonym of the attribute `degenerated` introduced in this approach. Other properties of a free algebra are that it is generated from a given set of atoms (attribute `well_founded`) and there is no other operation but the one specified by a signature (attribute `ECIW-strict`[1]). Our atoms are *elementary instructions* which are non compound elements of the `carrier` (they are not results of any operation). Finally, `IfWhileAlgebra` is introduced as

```
definition
  mode IfWhileAlgebra is
       non degenerated well_founded ECIW-strict preIfWhileAlgebra;
end;
```

The motivation behind `IfWhileAlgebra` is to have a set of assignments as the set of *elementary instructions* and to build compound instructions with algebra's operations as follows

[1] ECIW stands for Empty instruction, Catenation, If-instruction, and While-instruction

```
ei A    I1\;I2  ifte(C,I1,I2)   while(C,I)
```

The first expression is the empty instruction of the if-while algebra `A`, the second one is the sequential composition of instructions, the third one is conditional instruction with condition `C`, and the last one is the loop instruction with condition `C`. Additionally, we define a for-loop as an abbreviation

```
for-do(I0, C, J, I)   equals   I0\;while(C, I\;J)
```

Such an approach is best suitable for imperative programming and it can be extended to fulfill conditions of object-oriented programming (see Section 5). Functional paradigm could not be realized well as it requires quite a different approach.

The theory applying the concept of `IfWhileAlgebra` could be easily ported to algorithms defined on AMI machines by defining appropriate if-while algebras and this work is planed as student tasks (engineer or master theses).

2.2 Execution function

The execution function is defined as follows

```
definition
  let A be preIfWhileAlgebra;
  let S be non empty set;
  let T be Subset of S;
  mode ExecutionFunction of A,S,T
         -> Function of [:S, the carrier of A:], S  means
  it is complying_empty-instruction &
  it is complying_catenation &
  it complies_if_wrt T &
  it complies_while_wrt T;
end;
```

The non empty set S is intended to be an arbitrary set of states[2] and T to be a set of true states. An execution function is a function f from the Cartesian product $S \times |A|$ into S which

- does not change state for empty instruction (`complying_empty-instruction`),
  `f.(s, ei A) = s`
- is a function composition for catenation of instructions (`complying_catenation`),
  `f.(s, I1;I2) = f.(f.(s, I1), I2)`
- obeys conditional instructions with respect to the set of true states (`complies_if_wrt`),
  `f.(s, C) in T implies f.(s, ifte(C,I1,I2)) = f.(f.(s, C), I1)`
  `f.(s, C) nin T implies f.(s, ifte(C,I1,I2)) = f.(f.(s, C), I2)`
- obeys loop instructions (`complies_while_wrt`)
  `f.(s, C) nin T implies f.(s, while(C,I)) = f.(s, C)`
  `f.(s, C) in T implies f.(s, while(C,I)) = f.(f.(f.(s,C),I),while(C,I))`

It means that the f-execution in a state s of the instruction

$$\textit{if } C \textit{ then } I_1 \textit{ else } I_2 \qquad (\texttt{ifte(C, I1, I2)})$$

[2] S may be the set of states of a certain machine, but it is not required.

starts from executing C in state s and if the resulted state $s' = f(s, C)$ is an element of true states ($s' \in T$) then the instruction I_1 is executed in state s'. Otherwise, the instruction I_2 is executed in s'. The f-execution of

$$\textit{while } C \textit{ do } I \qquad (\texttt{while(C, I)})$$

starts also from executing C in state s and while the resulted state is a true state then the execution of I and after that C is repeated. The execution function is determined by the execution on loop instructions I and is unknown[3] for nonterminating loops.

2.3 Termination

The termination is defined for pairs (s, I) by induction on the structure of instruction I in the usual way. The induction base is that every elementary instruction terminates in any state. The loop instruction `while(C, I)` terminates in state `s` if `C` terminates in `s` and there exists a finite sequence (`s1`, ..., `sn`, `z`) which is a sequence of states of iteration of `I\;C` such that `s1` equals `f.(s, C)`, states `s1`, ..., `sn` are true states and `z` is not true state, and `I\;C` terminates in every state `si`. In Mizar it looks like

```
[s,C] in it &
(ex r being non empty FinSequence of S st
    r.1 = f.(s,C) & r.len r nin T &
    for i being Nat st 1 <= i & i < len r
    holds r.i in T & [r.i, I\;C] in it & r.(i+1) = f.(r.i, I\;C))
implies [s, while(C,I)] in it
```

where `it` stands for the set of terminating pairs.

The proof of the termination of an algorithm (for states satisfying the input condition) must be done following the structure of the algorithm. It could be semi-automatized with Mizar adjectives (attributes) and registrations or with Mizar redefinitions. For example, we introduce an attribute

```
I is absolutely-terminating
```

for the condition that for any state and execution function the instruction `I` terminates. Then we register elementary instructions to be absolutely-terminating. The same registration is done for the empty instruction, the composition and the conditional instructions built up of the absolutely-terminating instructions. Only a while-loop is omitted for the well-known reason.

```
registration
  let A be preIfWhileAlgebra;
  cluster ei A -> absolutely-terminating;
  let I,J be absolutely-terminating Element of A;
  cluster I\;J -> absolutely-terminating;
  let C be absolutely-terminating Element of A;
  cluster ifte(C,I,J) -> absolutely-terminating;
end;
```

After these registrations compound instructions without while-loops are known by the Mizar verifier to be terminating.

[3] The value of the execution function on a state and a nonterminating loop `while(C, I)` is a certain state but the state is not specified.

2.4 Correctness

To prove the correctness of an algorithm, we need to prove the termination for states satisfying the input condition and to prove that the result states satisfy the output condition. The most essential problems appear when the algorithm includes a while-loop. The termination of while-loop iteration is proved according to the following scheme

```
scheme Termination
  {A() -> preIfWhileAlgebra,    I() -> (Element of A()),
   S() -> non empty set,        s() -> (Element of S()),
   T() -> Subset of S(),        F(set) -> Nat,
   f() -> ExecutionFunction of A(),S(),T(),
   P[set]
  }:
     f() iteration_terminates_for I(),s()
  provided
A1:  s() in T() iff P[s()]
  and
A2:  for s being Element of S() st P[s]
     holds (P[f().(s,I())] iff f().(s,I()) in T()) &
            F(f().(s,I())) < F(s);
```

The idea carried by the scheme is that if in each step of the iteration some nonnegative value is reduced, then the iteration terminates. Literally, the scheme states that the iteration of the instruction I according to the execution f started with the state s terminates provided that conditions A1 and A2 hold. The condition A1 means that the state s is a true state if and only if s has some property P. The condition A2 means that the property P is actually the loop condition and that the nonnegative natural value $F(s)$ is reduced with each iteration.

The second scheme is prepared to prove loop-invariants and may be seen as a MIZAR adaptation of Hoare's while rule [11]. The loop-condition is denoted here with R and the invariant is denoted with P.

```
scheme InvariantSch
  {A() -> preIfWhileAlgebra,    C,I() -> (Element of A()),
   S() -> non empty set,        s() -> (Element of S()),
   T() -> Subset of S(),
   f() -> ExecutionFunction of A(),S(),T(),
   P,R[set]
  }:
  P[f().(s(),while(C(),I()))] & not R[f().(s(),while(C(),I()))]
provided
 P[s()]
and
 f() iteration_terminates_for I()\;C(), f().(s(),C())
and
 for s being Element of S() st P[s] & s in T() & R[s]
   holds P[f().(s,I())]
and
 for s being Element of S() st P[s] holds
   P[f().(s,C())] & (f().(s,C()) in T() iff R[f().(s,C())]);
```

3 Model with integers as the only data type

To start investigations on real algorithms we first need to introduce assignment instructions as elementary instructions. The assignment instruction should assign to a variable a value which is calculated for the current state according to some arithmetic expression. The expression may include variables, constants, and a limited number of arithmetic operations. In this model values are integers only and operations are the following:

```
+  -  *  div  mod  eq  leq  gt
```

States are functions from a set `X` of locations into the set `INT` of integers. In the Mizar notation, states are elements of the set `Funcs(X, INT)`. A variable is a function from states into locations and an expression is a function from states into integers.

```
definition
  let X be non empty set;
  mode Variable of X is Function of Funcs(X, INT), X;
  mode Expression of X is Function of Funcs(X, INT), INT;
end;
```

For a variable v and a state s, $v(s)$ is a location. Namely, $v(s)$ says in which location is the value of v in the state s. Such approach allows to have static variables (when v is a constant function) as well as dynamic ones. It means that we may easily model variables for arrays like `A[i]`.

The definition above gives only a rough approximation of variables and expressions. Not all functions are allowed to be expressions, especially, incomputable ones should not be. Therefore we give some additional conditions that limit the set of variables and expressions and, simultaneously, allow to write algorithms in a natural way (and to prove their correctness). For example, the correctness of the following algorithms has been proved using this approach.

```
s:=1\; for-do(i:=2, i leq n, i+=1, s*=i)

while(y gt 0, z:=x\; z%=y\; x:=y\; y:=z)

while(y gt 0,
  z:=(.x-.y)\; if-then(z lt 0, z*=-1)\; x:=y\; y:=z
)

x:=0\; y:=1\; for-do(i:=1, i leq n, i+=1, z:=x\;x:=y\;y+=z)

y:=1\; for-do(i:=1, i leq n, i+=1, y*=x)

y:=1\;
while(m gt 0,
  if-then(m is_odd, y*=x)\; x*=x\; m/=2
)
```

The algorithms above calculate respectively

- factorial $n!$,
- gcd by modulo operation (Euclid Algorithm 1),

- gcd by subtraction (Euclid Algorithm 2),
- Fibonacci numbers,
- exponentiation by multiplication,
- exponentiation by squaring.

4 An example – exponentiation by squaring

Let us consider the algorithm which calculates exponentiation by squaring:

```
y:=1\;
while(m gt 0,
  if-then(m is_odd, y*=x)\; x*=x\; m/=2
)
```

We want to prove that it calculates exponentiation, i.e.,

```
for x,y,m being Variable of g
st ex d being Function st d.b = 0 & d.x = 1 & d.y = 2 & d.m = 3
for s being Element of Funcs(X, INT)
for n being Nat st n = s.m holds
g.(s,  y:=1\; while(m gt 0, if-then(m is_odd, y*=x)\; x*=x\; m/=2)
   ).y = (s.x)|^n
```

where `g` is an execution function. The function `d` differentiates variables `b`, `x`, `y`, and `m`. Variable `b` is responsible for true states – a state is true if the value of `b` is different from 0. In the proof we denote the invariant of the while-loop by `P` and the value reduced in the while-loop by `F`. The scheme `Termination` is used to justify the formula labeled with `B` and the scheme `InvariantSch` to justify `E`.

```
  proof
    set S = Funcs(X, INT); set T = S\(b,0);
    let x,y,m be Variable of g;
    given d being Function such that
A0: d.b = 0 & d.x = 1 & d.y = 2 & d.m = 3;
D0: m <> x & m <> y & x <> y & b <> x & b <> y & b <> m by A0;
    let s be Element of Funcs(X, INT);
    let n be Nat such that
A1: n = s.m;
    set C = m gt 0;  set I = if-then(m is_odd, y*=x);
    set J = I\; m/=2\; x*=x;
    set s0 = g.(s, y:=1);  set s1 = g.(s0,C);  set fs = g.(s0, while(C,J));
A2: g.(s, y:=1\; while(C, J)) = fs by AOFA_000:def 29;
Z0: g complies_if_wrt T by AOFA_000:def 32;
A4: s1.x = s0.x & s1.m = s0.m & s1.y = s0.y &
    (s0.m > 0 implies s1.b = 1) & (s0.m <= 0 implies s1.b = 0) by Z0,D0,Th015;
   defpred P[Element of S] means
      (s.x)|^n = ($1.y)*(($1.x)to_power($1.m)) & $1.m >= 0;
   defpred R[Element of S] means
      $1.m > 0;
A:  P[s0] proof ... end;
```

```
    deffunc F(Element of S) = In($1.m, NAT);
B0: g.(s0,C) in T iff R[g.(s0,C)] by A4,LemTS;
B1: for s being Element of S st R[s]
    holds (R[g.(s,J\;C)] iff g.(s,J\;C) in T) & F(g.(s,J\;C)) < F(s)
      proof ... end;
B:  g iteration_terminates_for J\;C, g.(s0,C) from Termination(B0,B1);
C:  for s being Element of S st P[s] & s in T & R[s]
    holds P[g.(s,J)]
      proof ... end;
D:  for s being Element of S st P[s] holds
    P[g.(s,C)] & (g.(s,C) in T iff R[g.(s,C)])
      proof ... end;
E:  P[g.(s0,while(C,J))] & not R[g.(s0,while(C,J))]
    from InvariantSch(A,B,C,D); then
    fs.m = 0 & (fs.x) to_power 0 = 1 by XXREAL_0:1,POWER:29;
    hence g.(s, y:=1\; while(m gt 0, J)).y = (s.x)|^(s.m) by A2,E;
  end;
```

5 Model over an algebra (ADT)

Object oriented and generic programming paradigms require a different model than a model with one data type. We rather want to have a rich enough collection of types, a subtyping structure, and an algebra of constants of these types. The algebra should include operations like the n-th element of an array, dereferencing, the dot operator, etc. The types and operations could be specified by an algebraic signature. Then we want to build terms of this signature including constants and syntactic variables. Finally, some compound terms together with syntactic variables are intended to serve as program variables (l-values).

The base of this model is the structure, [3].

```
definition
  let J be non empty non void ManySortedSign;
  let T be MSAlgebra over J;   :: term algebra
  let X be GeneratorSet of T;  :: program variables
  struct (UAStr) ProgramAlgStr over J,T,X(#
    carrier -> set,
    charact -> PFuncFinSequence of the carrier,
    assignments ->
        Function of Union [|X, the Sorts of T|], the carrier
  #);
end;
```

The structure `PrograAlgStr` is defined over a signature `J` of types and an algebra `T` of terms. It includes the fields `carrier` and `charact` like `IfWhileAlgebra` and, moreover, the generator set `X` which chooses program variables from terms and the field `assignments` which for each pair (t_1, t_2) points the instruction $t_1 := t_2$ from the `carrier`. Terms t_1, t_2 are of the same type.

For example, the algebra of previously mentioned macro-instructions could be easily expressed in this manner.

6 Conclusions and further work

The approach presented here is the project in MML aimed at the analysis of algorithms. It allows to formalize the variety of algorithms including proofs of correctness and termination.

However, the formalization is not completed. It has just been started and still does not give all advantages of Hoare's logic [11]. For example, the invariants still belong to a metalanguage and the uniform axiomatization of them could not be done well.

Further work concerns the development of the formalization to reach an acceptable level of fluency of analysis of algorithms. It relies on the formalization of all elements of Hoare's logic as well as on the utility of Mizar type widening mechanisms (see [2]). The computer algebra of types available in Mizar promises a solution to some nasty jobs like tracking variable values. For this purpose we also need an algebra of Hoare's formulae. On the other hand, internal Mizar automation should be completed by external automation. It means, we need external software which could make automatically the Mizar description of an algorithm and complete as big as possible fragments of correctness proofs.

References

1. Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.
2. Grzegorz Bancerek. On the structure of Mizar types. *Electronic Notes in Theoretical Computer Science*, 85(7), 2003.
3. Grzegorz Bancerek. Program algebra over an algebra. *Formalized Mathematics*, 20(**4**):309–341, 2012.
4. Grzegorz Bancerek and Piotr Rudnicki. A compiler of arithmetic expressions for SCM. *Formalized Mathematics*, 5(**1**):15–20, 1996.
5. Jing-Chao Chen. Insert sort on $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 8(**1**):119–127, 1999.
6. Jing-Chao Chen. Quick sort on SCMPDS. *Formalized Mathematics*, 9(**2**):413–418, 2001.
7. Jing-Chao Chen. Dijkstra's shortest path algorithm. *Formalized Mathematics*, 11(**3**):237–247, 2003.
8. Jing-Chao Chen and Yatsuka Nakamura. Bubble sort on $\mathbf{SCM}_{\mathrm{FSA}}$. *Formalized Mathematics*, 7(**1**):153–161, 1998.
9. Yasushi Fuwa and Yoshinori Fujisawa. Algebraic group on fixed-length bit integer and its adaptation to IDEA cryptography. *Formalized Mathematics*, 7(**2**):203–215, 1998.
10. Yasushi Fuwa and Yoshinori Fujisawa. High-speed algorithms for RSA cryptograms. *Formalized Mathematics*, 9(**2**):275–279, 2001.
11. Charles A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–585, October 1969.
12. Artur Korniłowicz. The construction of **SCM** over ring. *Formalized Mathematics*, 7(**2**):295–300, 1998.
13. Jarosław Kotowicz, Beata Madras, and Małgorzata Korolkiewicz. Basic notation of universal algebra. *Formalized Mathematics*, 3(**2**):251–253, 1992.
14. Gilbert Lee and Piotr Rudnicki. Correctness of Dijkstra's shortest path and Prim's minimum spanning tree algorithms. *Formalized Mathematics*, 13(**2**):295–304, 2005.

15. Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
16. Beata Perkowska. Free universal algebra construction. *Formalized Mathematics*, 4(**1**):115–120, 1993.
17. Elżbieta Ramm and Edmund Woronowicz. A computerized system of proving properties of programs. Technical Report 403, ICS PAS, Warszawa, March 1980.
18. Piotr Rudnicki and Włodzimierz Drabent. Proving properties of pascal programs in mizar-2. *Acta informatica*, 22:311–331, erratum: 699–707, 1985.
19. Christoph Schwarzweller. The correctness of the generic algorithms of Brown and Henrici concerning addition and multiplication in fraction fields. *Formalized Mathematics*, 6(**3**):381–388, 1997.
20. Andrzej Trybulec and Yatsuka Nakamura. Euclid's algorithm. *Formalized Mathematics*, 4(**1**):57–60, 1993.
21. Andrzej Trybulec, Yatsuka Nakamura, and Noriko Asamoto. On the compositions of macro instructions. Part I. *Formalized Mathematics*, 6(**1**):21–27, 1997.
22. Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. An extension of **SCM**. *Formalized Mathematics*, 5(**4**):507–512, 1996.

Part II

Interactive Theorem Proving

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Equalities in Mizar

Artur Korniłowicz

Institute of Informatics
University of Białystok, Poland
arturk@math.uwb.edu.pl

Abstract. The usage of extensionality of sets, possibly satisfying some additional properties, by proof checkers will be presented. In particular, we will show how extensionality influences proof tactics and equational calculus. In the paper, we collect short descriptions of Mizar constructions, which have an impact on two basic Mizar modules: Reasoner for supporting definitional expansions, and Equalizer for computing the congruence closure of a given set of equalities.

1 Introduction

Axiom of extensionality is a very basic axiom of many set theories. It says that *two sets are equal (are the same set) if they have the same elements*, what in systems with equality could be symbolically written as:

$$\forall_{X,Y} : (\forall_z : z \in X \Leftrightarrow z \in Y) \Rightarrow X = Y.$$

To see how the axiom can be used to prove equality of two particular sets in practice let's examine a simple example, and prove that intersection of two sets is commutative, that is $A \cap B = B \cap A$. To justify this fact, the statement $\forall_z : z \in A \cap B \Leftrightarrow z \in B \cap A$ has to be proven. Let's then fix a z and assume that $z \in A \cap B$. By definition of the intersection we know that $z \in A \wedge z \in B$. Then, by commutativity of the conjunction we know that $z \in B \wedge z \in A$, and hence $z \in B \cap A$. Opposite direction of the equivalence can be proven similarly.

However, in a case of more complex sets, sets which elements are of some particular structure or satisfy some particular properties (e.g. relations or functions), it is often better to express equality of such sets in terms adequate to the domain to which those sets belong. A good example is the equality of two functions. The standard definition says that two functions are equal if their domains equal each other and assignments for each argument are equal, that is

$f = g \quad \Leftrightarrow \quad \mathrm{dom}(f) = \mathrm{dom}(g) \ \wedge \ \forall_x : x \in \mathrm{dom}(f) \Rightarrow f(x) = g(x)$.

The definition looks more complex than the extensionality axiom, but it is easier to use to prove the equality of functions.

Let's consider two functions $f, g : \mathbb{N} \to \mathbb{N}$ such that $f(n) = n$ and $g(n) = |n|$ for every natural number n. These functions are equal because $\mathrm{dom}(f) = \mathbb{N} = \mathrm{dom}(g)$, and by definitions of f and g we obtain that for every natural number n holds $f(n) = n = |n| = g(n)$.

One can imagine a proof, that would treat these functions as sets of ordered pairs, and use the extensionality axiom directly. Firstly, f should be expressed as

$\{[0,0],[1,1],[2,2],\dots\}$, and g as $\{[0,|0|],[1,|1|],[2,|2|],\dots\}$. Then these two sets of pairs should be compared.

In the paper, we present short descriptions of all MIZAR constructions [5, 17], which have an impact on processing equalities. In Section 2 we describe how MIZAR users can control structures of proofs. In Section 3 we show how to formulate different facts about equalities to increase computational power of the MIZAR VERIFIER. We will focus on practical aspects of two basic MIZAR modules: REASONER for supporting definitional expansions, and EQUALIZER for computing the congruence closure of a given set of equalities. We describe how to utilize functionalities of the modules in practice, rather than how the modules are structured and implemented.

2 Reasoning

Proof system of the MIZAR is based on logical reasoning in the Jaśkowski style of the natural deduction [7]. Structures of proofs are related to the structures of the formulas being proved. However, in many cases it is easier to trace the reasonings not by following the structure of the statement, but by following the structure of the definition of the main symbol of the statement, that is to exploit *definitional expansions* [2, 13]. Definitional expansions are then strictly related to definitions. So, a natural question "how to manage expansions of overloaded definitions" arises. Overloading can be understood syntactically, that is when a symbol is overloaded to spell different notions in the same way (e.g. + as the addition of complex numbers, and + as the strings concatenation). Overloading can be also understand semantically, when the same operation is applied to arguments of different types and its meaning can be defined using symbols and notions specific to the domain of the arguments (e.g. the composition $*$ of relations applied to functions). In the MIZAR language, syntactically overloaded definitions are understood simply as different definitions, while definitions overloaded semantically are understood as *redefinitions* of the notion.

In this paper we are interested in processing equalities, so we are focusing on semantical overloading of the axiom of extensionality. However, the theory of overloading concerns not only equality, but also other relations between objects.

As a case study, we propose a system of redefinitions of equalities of functions. Of course, equality of two particular functions can be proven from the axiom of extensionality directly. However, as it was shown in the Introduction, it is often easier and more natural to prove it using notions like domain and value assignment. Moreover, we will show, that even in the such easy case, one redefinition may not be so useful as a system of redefinitions depending on exact types of functions. Namely, one can consider functions, of which declared types are without explicitly determined domains and codomains; or functions between determined, but possibly not equal, sets; or functions, of which declared type is just the same type. We then propose to consider four different redefinitions with different definientia, which prevents from unnecessary repetitions of proofs of facts, which can be inferred from declared types of arguments.

1) General case

```
definition
 let f1,f2 be Function;
 redefine pred f1 = f2 means
 dom f1 = dom f2 &
```

```
 for x being object st x in dom f1 holds f1.x = f2.x;
 compatibility;
end;
```

2) $f_1 : A_1 \to B_1$, $f_2 : A_2 \to B_2$

```
definition
 let A1,A2 be set, B1,B2 be non empty set;
 let f1 be Function of A1,B1, f2 be Function of A2,B2;
 redefine pred f1 = f2 means
 A1 = A2 & for a being Element of A1 holds f1.a = f2.a;
 compatibility;
end;
```

3) $f_1 : A \to B_1$, $f_2 : A \to B_2$

```
definition
 let A be set, B1,B2 be non empty set;
 let f1 be Function of A,B1, f2 be Function of A,B2;
 redefine pred f1 = f2 means
 for a being Element of A holds f1.a = f2.a;
 compatibility;
end;
```

4) $f_1, f_2 : A \to B$

```
definition
 let A,B be set;
 let f1,f2 be Function of A,B;
 redefine pred f1 = f2 means
 for a being Element of A holds f1.a = f2.a;
 compatibility;
end;
```

Case 1) is applicable for all types that behave as functions. Case 2) can be used for functions defined between possibly different sets. Comparing domains of functions can be reduced to comparing sets `A1` and `A2`, because it is provable that these sets are the domains. Case 3) is applicable for functions with common domains. Then, the definiens can be simplified, and comparison of domains can be removed. Case 4) can be used for functions with common domains and codomains, which could be possibly empty sets.

Advantages of proposed redefinitions are illustrated by two different proofs of the same fact:

```
for f,g being Function of NAT,NAT st
 (for n being Nat holds f.n = n) &
 (for n being Nat holds g.n = |.n.|)
holds f = g
proof
  let f,g be Function of NAT,NAT such that
A0: for n being Nat holds f.n = n and
A1: for n being Nat holds g.n = |.n.|;
A2: dom f = NAT by FUNCT_2:def 1;
  hence dom f = dom g by FUNCT_2:def 1;
  let n be object;
  assume n in dom f;
  then reconsider m = n as Nat by A2;
  thus f.n = m by A0
  .= |.m.| by ABSVALUE:def 1
  .= g.n by A1;
end;
```

```
for f,g being Function of NAT,NAT st
 (for n being Nat holds f.n = n) &
 (for n being Nat holds g.n = |.n.|)
holds f = g
proof
  let f,g be Function of NAT,NAT such that
A0: for n being Nat holds f.n = n and
A1: for n being Nat holds g.n = |.n.|;
  let n be Element of NAT;
  thus f.n = n by A0
  .= |.n.| by ABSVALUE:def 1
  .= g.n by A1;
end;
```

The right proof is simpler and shorter than the left one. It is because, due to declared types of considered functions, in the right proof the domains of the functions do not have to be touched. All reasoning is about the values assigned by the functions.

System of redefinitions of equality of particular objects can be expanded further, for example for relations, partial functions, sequences, finite sequences, etc. We could also imagine, that for every new type of objects, a new redefinition of the equality of elements of that type could be introduced to the Mizar Mathematical Library [1].

3 Equalizer

The Equalizer is a Mizar module dealing with equational calculus [11]. It collects all terms from the processed formula (inference), and computes the *congruence closure* over equalities available in the inference, where the congruence closure of a relation R defined on a set A is a minimal *congruence relation* (a relation that is reflexive, symmetric, transitive, and monotonic) containing the original relation R.

The computed congruence closure of a given inference is used by the Mizar Checker to detect a possible contradiction and to refute the inference (Mizar is a disprover), what can happen when one of the following cases holds

- there are two premises of the form $P[x]$ and $\neg P[y]$ and x, y are congruent, or
- there is a premise of the form $x \neq y$ again when x, y are congruent.

where two elements x and y are congruent, if the pair $[x, y]$ belongs to the congruence closure.

The second case can be applied to check if an equality is a consequence of a given set of equalities.

3.1 Sources of equalities

In this section we present possible sources of equalities, which can be taken into account processing a given inference. The sources can be classified into the following categories:

- occurring explicitly in a given inference,
- term expansions (`equals`),
- properties,

- term reductions,
- term identifications,
- arithmetic,
- type changing (`reconsider`),
- others, e.g. processing structures.

Equalities occurring explicitly in a given inference

A very basic case are equalities, which are stated in the statement to be proven, e.g.

```
for a,b being Element of INT.Ring
  for c,d being Complex st a = c & b = d
    holds a + b = c + d;
```

Term expansions

One of the methods defining new functors can use the following syntax:

```
definition
 let x1 be θ1, x2 be θ2, ..., xn be θn;
 func ⊗(x1,x2,...,xn) -> θ equals :ident:
  τ(x1,x2,...,xn);
 coherence
 proof
  thus τ(x1,x2,...,xn) is θ;
 end;
end;
```

which introduces a new functor $\otimes(x_1, x_2, \ldots, x_n)$ equals to $\tau(x_1, x_2, \ldots, x_n)$. Such definitions, whenever terms $\otimes(x_1, x_2, \ldots, x_n)$ occur in an inference, allow the Verifier to generate equalities $\otimes(x_1, x_2, \ldots, x_n) = \tau(x_1, x_2, \ldots, x_n)$. For example,

```
definition
 let x,y be Complex;
 func x - y -> Complex equals
 x + (-y);
 coherence;
end;
```

causes that all instantiations of terms `x-y` are expanded to `x+-y`. As a gain of such expansions, for example, the equality `a-b-c = a+(-b-c)` is a direct consequence of associativity of addition. It holds because the term `a-b` is expanded to the term `a+-b`, and the term `a-b-c` is expanded to the term `a-b+-c`, what both results in `a+-b+-c`. On the other hand, the term `-b-c` is expanded to the term `-b+-c`, what creates the term `a+(-b+-c)`, that is the associative form of `a+-b+-c`.

An important feature of this kind of term expansions is that it is "a one way" expansion, in the sense, that terms $\otimes(x_1, x_2, \ldots, x_n)$ are expanded to $\tau(x_1, x_2, \ldots, x_n)$, but not vice-versa. The reason of such treatment is to avoid ambiguity of expansions and over-expanding terms.

Properties

Properties in Mizar are special formulas, which can be registered while defining functors [9]. Mizar supports `involutiveness` and `projectivity` for unary operations and `commutativity` and `idempotence` for binary operations. If a property is registered for some functor, the Equalizer processes appropriate equalities adequate to the property, where for `involutiveness` the equality is $f(f(x)) = x$, for `projectivity` it is $f(f(x)) = f(x)$, for `commutativity` it is $f(x, y) = f(y, x)$, and for `idempotence` $f(x, x) = x$.

Term reductions

Another method of imposing the Equalizer to generate extra equalities based on terms occurring in processed inferences are *term reductions* [8] with syntax:

```
registration
 let x1 be θ1, x2 be θ2, ..., xn be θn;
 reduce τ1(x1, x2, ..., xn) to τ2(x1, x2, ..., xn);
 reducibility
 proof
  thus τ1(x1, x2, ..., xn) = τ2(x1, x2, ..., xn);
 end;
end;
```

Term reductions can be used to simplify terms to their proper subterms. This simplification relies on matching terms existing in the processed inference with left-side terms of all accessible reductions, and whenever the Equalizer meets an instantiation σ of the term $\tau_1(x_1, x_2, \ldots, x_n)$, it makes σ equal to its subterm equivalent to $\tau_2(x_1, x_2, \ldots, x_n)$.

The restriction about simplifying terms to their proper subterms, not to any terms, is to fulfill the general rule established for the system, that the Equalizer does not generate extra terms and does not grow up the universe of discourse.

An example and counterexample of possible reductions could be reducing the complex conjugate of a real number to the number, and reducing the subtraction a number from itself to the zero.

example	counterexample

`registration`
` let r be Real;`
` reduce r*' to r;`
` reducibility;`
`end;`
	`registration`
	` let r be Real;`
	` reduce r - r to 0;`
	` reducibility;`
	`end;`

Term identifications

In mathematics, there are different theories which at some their parts are about the same objects. For example, when one considers complex numbers and extended real numbers (reals augmented by $+\infty$ and $-\infty$) and discusses basic operations on such numbers (like addition, subtraction, etc.), it can be quickly recognized that if the numbers are reals, the results of these operations are equal each other. That is, there is no difference, if one adds reals in the sense of complex numbers or in the sense of extended real numbers. Therefore, pairs of such operations could be identified on appropriate sets of arguments.

Mizar provides a special construction for such identifications [3] with syntax:

```
registration
 let x1 be θ1, x2 be θ2, ..., xn be θn;
 let y1 be Ξ1, y2 be Ξ2, ..., yn be Ξn;
 identify τ1(x1,x2,...,xn) with τ2(y1,y2,...,yn)
  when x1 = y1, x2 = y2, ..., xn = yn;
 compatibility
 proof
  thus x1 = y1 & x2 = y2 & ... & xn = yn
   implies τ1(x1,x2,...,xn) = τ2(y1,y2,...,yn);
 end;
end;
```

and, whenever the Equalizer meets an instantiation σ of the term $\tau_1(x_1, x_2, \ldots, x_n)$, it makes σ equal to the appropriate instantiation of $\tau_2(y_1, y_2, \ldots, y_n)$. A gain of using such identifications is that all facts proven about $\tau_2(y_1, y_2, \ldots, y_n)$ are applicable for $\tau_1(x_1, x_2, \ldots, x_n)$, as well.

An example of identification taken from the Mizar Mathematical Library could be the lattice of real numbers with operations `min`, `max` as the infimum and supremum of two elements of the lattice [4].

```
registration
 let a,b be Element of Real_Lattice;
 identify a "\/" b with max(a,b);
 identify a "/\" b with min(a,b);
end;
```

Built-in computations

Another source of equalities processed by the Equalizer are special built-in procedures for processing selected objects. Generating equalities by these routines is controlled by the environment directive **requirements** [9]. In our interest are two procedures dealing with boolean operations on sets (`BOOLE`) and basic arithmetic operations on complex numbers (`ARITHM`).

Requirements `BOOLE`

```
X \/ {} = X;     X /\ {} = {};     X \ {} = X;
{} \ X = {};     X \+\ {} = X;
```

Requirements ARITHM

```
x + 0 = x;    x * 0 = 0;    1 * x = x;
x - 0 = x;    0 / x = 0;    x / 1 = x;
```

Moreover, requirements ARITHM fires procedures for solving systems of linear equations over complex numbers.

Type changing

It is quite common situation, when one object can be treated as an element of different theories or different structures. For example, the empty set is the empty set in set theories, but it is also the zero number is some arithmetics. In computerized mathematics, to allow systems distinguish and understand symbols clearly and to avoid ambiguity, it is often required to express types of objects explicitly.

Mizar provides a special rule (**reconsider**) for forcing the system to treat a given term as if its type was the one stated.

For example, to consider 0 as an element of the field of real numbers (for example, to prove that it is the neutral element of its additive group), one can state

```
reconsider z = 0 as Element of F_Real;
```

The equality `z = 0` is obviously processed by the Equalizer.

3.2 Example with automatization

In this section we present an example how all described above techniques can automatize reasoning and make proofs shorter or even make theorems obvious. The working example (about elements of the additive group of real numbers G_Real) with all automatizations switched-off and all basic proof steps written within the proof is:

```
for a being Element of G_Real holds a + 0.G_Real = a
proof
  let a be Element of G_Real;
  reconsider x = a as Real;
A: 0.G_Real = the ZeroF of G_Real by STRUCT_0:def 6
  .= In(0,REAL) by VECTSP_1:def 1
  .= 0 by A1,SUBSET_1:def 8;
  thus a + 0.G_Real = (the addF of G_Real).(a,0.G_Real) by ALGSTR_0:def 1
  .= addreal.(a,0.G_Real) by VECTSP_1:def 1
  .= x + 0 by A,BINOP_2:def 9
  .= x by ARITHM:1
  .= a;
end;
```

while the theorem is obvious when all provided mechanism are utilized.
The equality `a + 0.G_Real = (the addF of G_Real).(a,0.G_Real)` is a consequence of the "equals" expansion of the definition:

```
definition
 let M be addMagma;
 let a,b be Element of M;
 func a+b -> Element of M equals
:: ALGSTR_0:def 1
 (the addF of M).(a,b);
end;
```

The equality `a + 0.G_Real = x + 0` is a consequence of the equality `x = a` (reconsider), the equality `0.G_Real = 0` and the term identification:

```
registration
 let a,b be Element of G_Real, x,y be Real;
 identify a+b with x+y when a = x, b = y;
 compatibility by BINOP_2:def 9;
end;
```

The equality `0.G_Real = 0` is a consequence of the "equals" expansion of the definition:

```
definition
 let S be ZeroStr;
 func 0.S -> Element of S equals
:: STRUCT_0:def 6
 the ZeroF of S;
end;
```

and the "equals" expansion of the definition:

```
definition
 func G_Real -> strict addLoopStr equals
:: VECTSP_1:def 1
 addLoopStr (# REAL,addreal,In(0,REAL) #);
end;
```

and the term reduction:

```
registration
 let r be Real;
 reduce In(r,REAL) to r;
 reducibility;
end;
```

The equality `x + 0 = x` is a consequence of built-in calculations over complex numbers. And finally, the equality `x = a` is a trivial consequence of the "reconsider".

4 Conclusions

Different aspects of dealing with equalities by the Mizar proof-checker are presented in the paper.

In Section 2 we introduced a system of redefinitions suitable for proving equalities of particular functions depending on their exact types. Similar systems could be defined

for other notions. A revision of the Mizar Mathematical Library concerning such redefinitions could be performed [10].

In Section 3 we described different techniques, which increase the number of equalities accessible for the Checker making more statements easier to prove. Some of the mechanisms were implemented in the system quite recently, and they are not exploited in articles stored in the Mizar Mathematical Library before the implementation. Then some revisions of the repository are necessary and they are successively carried. For example, our example about the group of real numbers

```
theorem
 for a being Element of G_Real holds a + 0.G_Real = a;
```

could be and should be reformulated as a term reduction, i.e.

```
registration
 let a be Element of G_Real;
 reduce a + 0.G_Real to a;
 reducibility;
end;
```

Acknowledgements

I would like to express my gratitude to Professor Andrzej Trybulec† for his support of my scientific growth. I am thankful for promoting my master thesis and Ph.D. thesis. I thank Him for every inspiring talk about Mizar and formal reasoning [23].
I am very honored that Professor Andrzej Trybulec was my Scientific Advisor and Chief.

References

1. Alama, J., Kohlhase, M., Mamane, L., Naumowicz, A., Rudnicki, P., Urban, J.: Licensing the Mizar Mathematical Library. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) Proceedings of the 18th Calculemus and 10th international conference on Intelligent computer mathematics. MKM'11, Lecture Notes in Computer Science, vol. 6824, pp. 149–163. Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2032713.2032726
2. Bishop, M., Andrews, P.B.: Selectively instantiating definitions. In: Automated Deduction - CADE-15, 15th International Conference on Automated Deduction, Lindau, Germany, July 5–10, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1421, pp. 365–380. Springer (1998)
3. Caminati, M.B., Rosolini, G.: Custom automations in Mizar. Journal of Automated Reasoning 50(2), 147–160 (February 2013), http://dx.doi.org/10.1007/s10817-012-9266-1
4. Chmur, M.: The lattice of real numbers. The lattice of real functions. Formalized Mathematics 1(**4**), 681–684 (1990), http://fm.mizar.org/1990-1/pdf1-4/real_lat.pdf
5. Grabowski, A., Korniłowicz, A., Naumowicz, A.: Mizar in a nutshell. Journal of Formalized Reasoning, Special Issue: User Tutorials I 3(2), 153–245 (December 2010)

6. Grabowski, A., Schwarzweller, C.: Revisions as an essential tool to maintain mathematical repositories. In: Proceedings of the 14th symposium on Towards Mechanized Mathematical Assistants: 6th International Conference. pp. 235–249. Calculemus '07 / MKM '07, Springer-Verlag, Berlin, Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-73086-6_20
7. Jaśkowski, S.: On the rules of suppositions in formal logic. Studia Logica, Nakładem Seminarjum Filozoficznego Wydziału Matematyczno-Przyrodniczego Uniwersytetu Warszawskiego (1934), http://books.google.pl/books?id=6w0vRAAACAAJ
8. Korniłowicz, A.: On rewriting rules in Mizar. Journal of Automated Reasoning 50(2), 203–210 (February 2013), http://dx.doi.org/10.1007/s10817-012-9261-6
9. Naumowicz, A., Byliński, C.: Improving Mizar texts with properties and requirements. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) Mathematical Knowledge Management, Third International Conference, MKM 2004 Proceedings. MKM'04, Lecture Notes in Computer Science, vol. 3119, pp. 290–301 (2004), http://dx.doi.org/10.1007/978-3-540-27818-4_21
10. Naumowicz, A., Korniłowicz, A.: A brief overview of Mizar. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics. TPHOLs'09, Lecture Notes in Computer Science, vol. 5674, pp. 67–72. Springer-Verlag, Berlin, Heidelberg (August 2009), http://dx.doi.org/10.1007/978-3-642-03359-9_5
11. Rudnicki, P., Trybulec, A.: Mathematical knowledge management in Mizar. In: Proc. of MKM 2001 (2001)
12. Trybulec, A., Korniłowicz, A., Naumowicz, A., Kuperberg, K.: Formal mathematics for mathematicians. Journal of Automated Reasoning 50(2), 119–121 (February 2013), http://dx.doi.org/10.1007/s10817-012-9268-z
13. Wos, L.: Automated Reasoning: 33 Basic Research Problems. Prentice-Hall, Englewood Cliffs, N.J. (1987)

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Improving Legibility of Proof Scripts Based on Quantity of Introduced Labels*

Karol Pąk

Institute of Computer Science,
University of Bialystok, Poland
pakkarol@uwb.edu.pl

Abstract. Formal proof checking systems such as Mizar or Isabelle/Isar can verify the correctness of proof scripts, both easily readable and obscure. However for humans, e.g., those who analyse the main idea of a formal proof or redevelop fragments of reasoning to make them stronger, the legibility has substantial significance. Furthermore, proof writers create still more and more complex deductions that cannot be shortened to several steps by any tools currently available. Therefore, it is important to better understand how we can facilitate the work of script readers modifying the order of independent deduction steps or reorganise the proof structure by extracting lemmas that are obtained automatically.
In this paper we present experimental result obtained with a method that improves proof legibility based on human short-term memory and we explore its impact for realisation of other, also popular methods.

Keywords: Operations on languages, Legibility of proofs, Proof assistants

1 Introduction

1.1 Motivations

One of the most important tasks of proof assistants such as Mizar [17, 23] or Isabelle/Isar [24] is checking proof step correctness. Clearly, inferences that are obvious for proof writers should be also obvious for checkers and vice versa. However, the notion of "obviousness" has often different meaning for humans and checkers [5, 21]. Generally, this difference is not so important problem if we do not need to analyze, adapt or modify the existing formal proofs scripts [10, 12]. According to the opinion of some proof writers, repairing a justification in re-developed fragments of reasoning that ceased to be acceptable, can be very difficult. We can observe similar situation in the area of proof script legibility. A proof assistant can check correctness of every syntactically correct formal proof scripts, even automatically generated, excessively large, or written in a chaotic way. However, any attempt to analyse such scripts by a human is extremely difficult or even impossible. The main reason for this situation is often the fact that proof scripts are created in an artificial language which "tries" to be similar to the one

* The paper has been financed by the resources of the Polish National Science Center granted by decision n°DEC-2012/07/N/ST6/02147.

that occurs in the traditional mathematical practice. Another reason is that readability does not come of zero cost on the side of proof script developers. Namely, the legibility strongly depends on the amount of effort invested in enhancing the readability by the author of a given proof script. Unfortunately, authors often do not care about legibility of their proof scripts. It is often a consequence of their assumption that no one, with the exception of a proof checker, will want to analyse their proof scripts. Furthermore if somebody would want to do it, then some tool rather than themselves should improve the legibility of their scripts. The experience with proof assistants shows that reading proof script is often unavoidable, e.g., if we adapt or modify existing proofs [8, 11], or if several authors cooperate on a common formalisation. The important point to note here is that creating such a tool that enhances legibility can in general be NP-complete [20].

1.2 Proposed approach

In this paper we present a next stage [18, 20] in the enhancing legibility of proof scripts based on a modification of the order of independent deduction steps. We concentrate on a modification that minimises the number of steps decorated by labels. Note that we need to refer to a premise, if we want to use it in the justification of a step. But in some cases to do this in the Mizar system we do not have to use the label of this premise. Indeed, referencing a step by its label may be replaced in the Mizar system by the `then` construction, if the step is located in the directly preceding step that refers to it (for more detail see [9, 17]). Additionally, if each reference to a fixed step can be realised by the `then` construction, then the label that decorates this step in unnecessary and can be removed from the proof script. Therefore, it is possible to minimise the number of labels that are introduced in proof scripts.

Analysing deductions contained in proof scripts of the Mizar Mathematical Library (MML), one may reasonably conclude that the number of introduced labels in a reasoning can be rarely minimised so that a proof reader can keep them in human short-term memory. However, minimisation to such a number is often possible in one-level deductions (i.e., substructures of a proof where nested lemmas are ignored) that are located on all levels of nesting (see Fig. 2). Note that only 4.5% of one-level deductions that occur in MML (Version `5.22.1191`) have more than 7 labels. Additionally, G. A. Miller shows that the capacity of human short-term memory is 7 ± 2 elements [15]. This limitation is also recognised in modern scientific literature that concerns human perception [3, 4]. Clearly, the capacity of memory decreases quickly with time and it is smaller in the case of similar information [25]. However, this capacity can be extended through training [6]. Therefore, small departure beyond the number 7 should be acceptable and this is the case for MML where the number of labels is in the range 5-10 [14].

In this paper we represent experimental results obtained with minimisation of the number of introduced labels. We combined this result with other criteria that improve proof scripts legibility and have been already recognised by the scientific community of people who write proof scripts in Mizar [18, 19] as well as in other systems [2, 13, 22]. Since, optimisation of legibility criteria in most cases is NP-hard [20], we present readability enhancements obtained with the help of the SMT-solver Z3 [16].

2 Labeled steps in terms of proof graphs

To formulate a criterion that minimises the number of introduced labels and the influence of this criterion implementation for the realisation of other similarly popular

criteria, we need to set the terminology and notation. Let $G = \langle V, A\rangle$ be a DAG and a vertex $u \in V$. We use the following notation:

$$\begin{array}{ll} N_G^-(u) :- \{v \in V : \langle v, u\rangle \in A\} & \text{(incoming arcs)}, \\ N_G^+(u) :- \{v \in V : \langle u, v\rangle \in A\} & \text{(outgoing arcs)}. \end{array} \tag{1}$$

Let A_1 be a subset of A. An arc $\langle u, v\rangle \in A$ is called an A_1*-arc* if $\langle u, v\rangle \in A_1$. A sequence $p = \langle u_1, u_2, \ldots, u_n\rangle$ of vertices of G is called an A_1*–path* if $\langle u_1, u_{i+1}\rangle$ is an A_1-arc for each $i = 1, 2, \ldots, n{-}1$. We identify here topological sortings of G, called also *linearisations*, with one-to-one functions $\tau : V \to \{1, 2, \ldots, |V|\}$, where $\tau(u) < \tau(v)$ for each A–arc $\langle u, v\rangle$.

An abstract model of a proof graph that represents the structure of natural deduction proofs, even with potentially nested subreasonings is fully explained in [18, 20]. However for our purposes it is enough to consider a simplified model which is represented by a DAG with the structure that ignores nested lemmas (i.e., one-level deductions). It is worth pointing out that the number of introduced labels on a one-level deduction in a proof script is independent of the number of introduced labels on another one-level deduction of such script. A DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A}\rangle$ with a distinguished set of arcs $\mathfrak{R}(\mathcal{D}) \subseteq \mathcal{A}$ is called a *simple abstract proof graph*. The vertices of $\mathcal{D}$ represent steps of the reasoning and $\mathcal{A}$–arcs represent the flow of information between different steps of the reasoning. A $\mathfrak{R}(\mathcal{D})$–arc, called a *reference arc*, represents the information flow between a premise (the tail of the arc) and a place of its use (the head of the arc). The other $\mathcal{A}$-arcs represent all kinds of additional constraints that force one step to precede another one, e.g., the dependence between a step that introduces a variable into the reasoning and a step that uses this variable in its expression.

```
1: reserve i, n, m for Nat;

   theorem
2:    i in Seg n implies i+m in Seg(n+m)
   proof
3:    assume A1: i in Seg n;
4:    then A2: 1 <= i by FINSEQ_1:1;
5:    i <= i+m by NAT_1:11;
6:    then A3: 1 <= i+m by A2,XXREAL_0:2;
7:    i <= n by A1,FINSEQ_1:1;
8:    then i+m <= n+m by XREAL_1:7;
9:    hence thesis by A3,FINSEQ_1:1;
   end;
```

```
   theorem
1:    fixes n m i::nat
2:    shows "i ∈ {k::nat. 1<=k & k<=n}==>
         i+m ∈ {k::nat. 1 <= k & k <= n+m}"
   proof -
3:    assume A1: "i ∈ {k::nat. 1 <= k & k <= n}"
4:    then have A2: "1 <= i" by simp
5:    have "i <= i+m" by simp
6:    then have A3: "1 <= i+m" using A2 by simp
7:    have "i <= n" using A1 by simp
8:    then have "i+m <= n+m" by simp
9:    then show ?thesis using A3 by simp
   qed
```

Fig. 1. An example proof script written in the Mizar language that is contained in [1] and its reformulation to the Isabelle/Isar language.

As an illustration, let us consider an example shown in Fig. 2 that represents the structure of proof scripts presented in Fig. 1, where solid arrows correspond to reference arcs that are also $\mathcal{A}$-arcs, and dashed arrows correspond to $\mathcal{A}$-arcs that are not reference arcs. Additionally, the term `Seg` n that occurs in Fig. 1 represents the set $\{1, 2, \ldots, n\}$. Clearly, both arcs and nodes of the abstract proof graph are not labeled. However we label each element in the actual graph only to simplify their identification. Note that this graph contains two one-level deductions (vertices 1–2 and vertices 3–9) and additionally $\twoheadrightarrow$ arrows that correspond to meta-edges of proof graphs,

which do not occur in our simplified model. We only recall that meta-edges represent dependencies between one-level deductions, i.e., between a step (e.g., the vertex 2) that as a justification contains the nested reasoning and each step of this reasoning (e.g., vertices 3–9). It is easily seen that in such a simplified model we have to take into consideration additional hidden dependencies that can occur between steps in one-level deductions. As an illustration note that the 1st step has to occur before the 2nd step, even if variables introduced in the 1st step do not occur in the statement of the 2nd step. Indeed, e.g., the variable `i` is used in the statement of the 3rd step that occurs in the nested reasoning that justify the 2nd step.

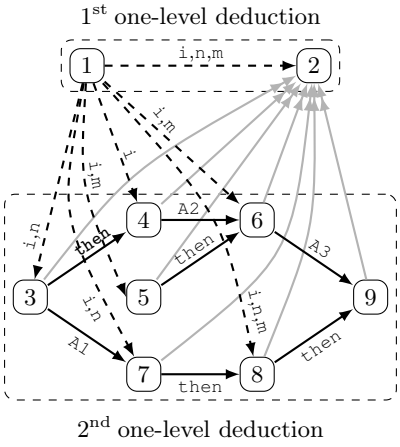


Fig. 2. The abstract proof graph illustrating the structure of the reasoning presented in Fig. 1.

Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$ be a one-level deduction. For simplicity, we assume that $\mathcal{A}$ contains additionally every hidden dependence between vertices of $\mathcal{V}$, and denote by $\mathfrak{R}(\mathcal{D})$ the set of reference arcs and hidden ones. We mark by $\texttt{then}(\mathcal{D})$ the set of references that can be replaced by the `then` construction. However, to study the general case, without the Mizar context, we will assume only the relation between distinguished sets of arcs in $\mathcal{D}$ that $\texttt{then}(\mathcal{D}) \subseteq \mathfrak{R}(\mathcal{D}) \subseteq \mathcal{A}$. Therefore, in further considerations we mean $\texttt{then}(\mathcal{D})$, $\mathfrak{R}(\mathcal{D})$ simply as two sets $\mathcal{A}_1$, $\mathcal{A}_2$, respectively, where $\mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \mathcal{A}$.

Recall that we identify the arrangement of reasoning steps that correspond to $\mathcal{V}$ in a proof script with a topological sortings of $\mathcal{D}$. Let us consider $\sigma \in TS(\mathcal{D})$. We define a metric $d_\sigma : \mathcal{V} \times \mathcal{V} \to \mathbb{N}$ that is called σ–distance and is given by $d_\sigma(v,u) = |\sigma(v) - \sigma(u)|$ for all $v, u \in \mathcal{V}$. We call a vertex $v \in \mathcal{V}$ a $\texttt{then}_{\mathcal{A}_1}(\sigma)$–step if v corresponds to a step that refers to the directly preceding step using a $\mathcal{A}_1$–arc (e.g., the vertex 4). We denote by $\texttt{then}_{\mathcal{A}_1}(\sigma)$ the set of such steps given by

$$v \in \texttt{then}_{\mathcal{A}_1}(\sigma) \iff \big(\sigma(v) \neq 1 \wedge \langle \sigma^{-1}(\sigma(v)-1), v \rangle \in \mathcal{A}_1\big)\,. \tag{2}$$

We call a vertex $v \in \mathcal{V}$ σ–*labeled* if at least once we have to use a label to refer to the statement of a step that corresponds to $\mathcal{V}$ (e.g., the vertex 3). The set of all σ–labeled

vertices, denoted by $\mathtt{lab}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma)$, is defined as follows:

$$v \in \mathtt{lab}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma) \iff \underset{u\in\mathcal{V}}{\exists}\langle v,u\rangle \in \mathcal{A}_2 \ \wedge \big(\langle v,u\rangle \in \mathcal{A}_1 \Longrightarrow d_\sigma(v,u) > 1\big)\,. \tag{3}$$

However, according to an additional syntax restriction of Mizar that prohibits referring to steps that introduce variables into the reasoning, we have to consider also the following set of σ–labeled vertices:

$$v \in \mathtt{lab}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma) \iff \underset{u\in\mathcal{V}}{\exists}\langle v,u\rangle \in \mathcal{A}_2 \wedge \Big(\big(\langle v,u\rangle \in \mathcal{A}_1 \Longrightarrow d_\sigma(v,u) > 1\big) \vee \big(\underset{w\in\mathcal{V}}{\exists}\langle v,w\rangle \in \mathcal{A}\setminus\mathcal{A}_1\big)\Big)\,. \tag{4}$$

We call $|\mathtt{lab}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}|$ the *lab–parameter*.

Based on the notions described above we can formulate the method of improving legibility that corresponds to the lab-parameter as the following decision problems:

The 1st Method of Improving Legibility ($\mathbf{MIL}_{\mathtt{lab}}$):
INSTANCE: A DAG $\mathcal{D}=\langle\mathcal{V},\mathcal{A}\rangle$, subsets $\mathcal{A}_1\subseteq\mathcal{A}_2\subseteq\mathcal{A}$, and a positive integer $K\leqslant|\mathcal{V}|$.
QUESTION: Does there exist $\sigma\in TS(\mathcal{D})$ for which $|\mathtt{lab}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma)|\leqslant K$?

The 1st Method of Improving Legibility limited to the Mizar system ($\mathbf{MIL}^{\mathtt{MIZ}}_{\mathtt{lab}}$):
INSTANCE: A DAG $\mathcal{D}=\langle\mathcal{V},\mathcal{A}\rangle$, subsets $\mathcal{A}_1\subseteq\mathcal{A}_2\subseteq\mathcal{A}$, and a positive integer $K\leqslant|\mathcal{V}|$.
QUESTION: Does there exist $\sigma\in TS(\mathcal{D})$ for which $|\mathtt{lab}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma)|\leqslant K$?

3 Optimisation of the lab–parameter

The complexity problem of improving legibility methods that corresponds to the lab–parameter optimisation has been studied in [20]. It has been shown that $\mathrm{MIL}_{\mathtt{lab}}$ is NP-complete and $\mathrm{MIL}^{\mathtt{MIZ}}_{\mathtt{lab}}$ is solvable in polynomial time. Here we concentrate first on properties of the polynomial time procedure that optimises the lab–parameter for Mizar proof scripts. Then we show that the $\mathrm{MIL}_{\mathtt{lab}}$ method for one-level deductions that potentially occur in Isabelle proof scrips is NP-hard.

3.1 The lab–parameter in the Mizar system

Let us fix a one-level deduction DAG $\mathcal{D}=\langle\mathcal{V},\mathcal{A}\rangle$ with two distinguished subsets of arcs $\mathcal{A}_1\subseteq\mathcal{A}_2\subseteq\mathcal{A}$. First note that some of the vertices of $\mathcal{D}$ have to be σ–labeled regardless of the σ choice. Indeed, every $v\in\mathcal{V}$ for which at last one of the following properties holds:
(i) $|N^+_{\langle\mathcal{V},\mathcal{A}_1\rangle}(v)|>1$,
(ii) $|N^+_{\langle\mathcal{V},\mathcal{A}_2\rangle}(v)|>|N^+_{\langle\mathcal{V},\mathcal{A}_1\rangle}(v)|$,
(iii) $|N^+_{\mathcal{D}}(v)|>|N^+_{\langle\mathcal{V},\mathcal{A}_2\rangle}(v)|>0$,
has to be σ–labeled in all $\sigma\in TS(\mathcal{D})$. Mark the set of such vertices by $\mathcal{L}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}$. Note also that if we remove all $\mathcal{A}$–arcs outgoing from vertices of $\mathcal{L}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}$ then the digraph obtained in this way, denoted by $\mathcal{D}'$, is a forest where every connected maximal tree is an arborescence (i.e., a rooted tree where all arcs are directed from leaves to the root). Additionally, every arc of $\mathcal{D}'$ is simultaneously $\mathcal{A}_2$–arc, $\mathcal{A}_1$–arc, and $\mathcal{A}$–arc, hence

$\mathtt{lab}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma) \setminus \mathcal{L}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}$ has to contain at least $|N^+_{\mathcal{D}'}(v)| - 1$ elements of $N^+_{\mathcal{D}'}(v)$ if only $N^+_{\mathcal{D}'}(v)$ is nonempty for each $v \in \mathcal{V}$. As it has been proven in [20], for each set of vertices that contain $\mathcal{L}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}$ and exactly $|N^+_{\mathcal{D}'}(v)| - 1$ elements of every nonempty $N^+_{\mathcal{D}'}(v)$, there exists a topological sorting $\sigma \in TS(\mathcal{D})$ for which $\mathtt{lab}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma)$ is equal to this set. Clearly, in this topological sorting every non-selected vertex $u \in N^+_{\mathcal{D}'}(v) \setminus \mathtt{lab}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma)$ has to be located in the directly preceding step v, since u is not decorated by a label. Additionally, this holds for each choice of $|N^+_{\mathcal{D}'}(v)| - 1$ elements of $N^+_{\mathcal{D}'}(v)$. Therefore, we can modify this choice in such a way that an arbitrary step of $N^+_{\mathcal{D}'}(v)$ can become not labeled. Hence from this we can conclude that the `lab`–parameter is minimal if each vertex v that "refers" to at least one "premise" with exactly one incoming $\mathcal{A}$-arc, has to contain at least one such premise that is located directly before v or more precisely:

Proposition 1. *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$ be a DAG with two distinguished sets of arcs $\mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \mathcal{A}$. Then $\mathtt{lab}^{\mathtt{MIZ}}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma)$ has the smallest possible size if and only if, for every $v \in \mathcal{V}$ it holds that*

$$N^-_{\mathcal{D}}(v) \cap L \neq \emptyset \Longrightarrow \sigma^{-1}(\sigma(v)-1) \in L, \tag{5}$$

where $\sigma \in TS(\mathcal{D})$ and $L = \{v \in \mathcal{V} : |N^+_{\langle \mathcal{V}, \mathcal{A}_1 \rangle}(v)| = |N^+_{\langle \mathcal{V}, \mathcal{A} \rangle}(v)| = 1\}$.

3.2 The `lab`–parameter in the Isabelle/Isar system

Now we show that the minimisation of the `lab`–parameter for Isabelle/Isar proof scripts is NP-hard. To achieve this we indicate a family of correct proof scripts for which the minimisation of the `lab`–parameter is equally hard as the minimisation of the size of a vertex cover.

In this paper, we do not concentrate on a full explanation of how the known NP-complete problem Vertex Cover (see GT1 in [7]) is reducible to the $\mathrm{MIL}_{\mathtt{lab}}$ problem (for more details see [20]). We present only a way to create proofs written in the Isabelle/Isar system that have structures described by graphs obtained in this reduction. In this way we show that difficult proof structures are indeed representable there.

Vertex Cover (**VC**):

Instance: An (undirected) graph $G = \langle V, E \rangle$ and a positive integer $K \leqslant |V|$.

Question: Is there a vertex cover of size at most K, i.e., a subset $V' \subseteq V$ of size at most K such that for each edge $\{u, v\} \in E$ at least one of u or v belongs to V'?

Let $G = \langle V, E \rangle$, $K \leqslant |V|$ be an instance of VC. For simplicity we assume that $V = \{1, 2 \ldots, |V|\}$. The instance of $\mathrm{MIL}_{\mathtt{lab}}$ that is used in the reduction of VC to $\mathrm{MIL}_{\mathtt{lab}}$ is defined as follows. We construct a digraph $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$ and subsets $\mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \mathcal{A}$ given by:

$$\begin{aligned} \mathcal{V} &:= V \times \{0, 1\}, \\ \mathcal{A}_1 &:= \{\langle \langle v, 0 \rangle, \langle v, 1 \rangle \rangle : v \in V\}, \\ \mathcal{A}_2 &:= \{\langle \langle v, 0 \rangle, \langle v, 1 \rangle \rangle : v \in V\} \cup \{\langle \langle v, 0 \rangle, \langle u, 1 \rangle \rangle : \{v, u\} \in E\}, \\ \mathcal{A} &:= \mathcal{A}_2. \end{aligned} \tag{6}$$

Obviously, $\mathcal{D}$, $\mathcal{A}_1$, and $\mathcal{A}_2$ determine a one-level deduction with two distinguished subsets of arcs. Additionally, this deduction together with K is an instance of $\mathrm{MIL}_{\mathtt{lab}}$ problem. Let us remind that the main idea of this reduction based on the fact that to obtain a vertex cover, for every edge $\{v, u\} \in E$, at least one of $\langle v, 0 \rangle$, $\langle u, 0 \rangle$, has to belong to $\mathtt{lab}_{\mathcal{A}_1,\mathcal{A}_2}(\sigma)$ for each $\sigma \in TS(\mathcal{D})$.

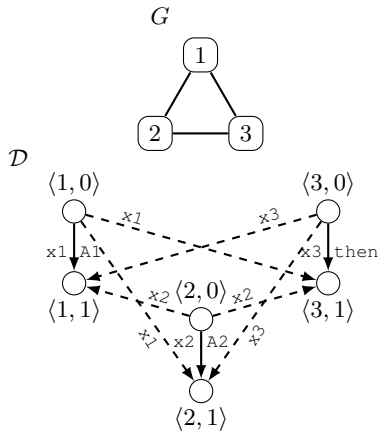
To create Isabelle/Isar proof scripts that correspond to the constructed deduction, we associate:

```
obtain xi::nat where Ai: "xi=i" by simp
```

with every vertex of the form $\langle i, 0\rangle$ and

```
have "xi=i & (xj1=xj1 & ... & xjn=xjn)" using Ai by simp
```

with every vertex of the form $\langle i, 1\rangle$, where $i \in V$, $\{j_1, j_2, \ldots, j_n\} = N^-_{\langle \mathcal{V}, \mathcal{A}_2 \setminus \mathcal{A}_1\rangle}(v)$. For illustration, an example of a reasoning that follows this pattern is presented in Fig. 3. It is simple to observe that every topological sorting of $\mathcal{D}$ organises such steps in the reasoning acceptable by the proof checker, since every occurring statement is "obvious". Additionally, this linearisation ensures that none of the variables and label identifers is used before their introduction in the reasoning. This completes the justification that computationally difficult instances can potentially occur in Isabelle/Isar proof scrips.




```
⟨1,0⟩: obtain x1::nat where
         A1: "x1=1" by simp
⟨2,0⟩: obtain x2::nat where
         A2: "x2=2" by simp
⟨3,0⟩ : obtain x3::nat where
          "x3=3" by simp
⟨3,1⟩ : then have"x3=3 & (x1=x1 & x2=x2)"
         by simp
⟨1,1⟩ : have "x1=1 & (x2=x2 & x3=x3)"
         using A1 by simp
⟨2,1⟩ : have "x2=2 & (x1=x1 & x3=x3)"
         using A2 by simp
```

Fig. 3. An example that illustrates the construction from Section 4, where considered vertex cover is equal to $\{1, 2\}$ and corresponds to vertices $\langle 1, 0\rangle, \langle 2, 0\rangle$, represented as steps decorated by `A1` and `A2`, respectively.

4 The `lab`-parameter in the process of improving other determinants of legibility

Our research is focused on the impact of the methods presented above on other popular methods such as increasing the number of **`then`** constructions (called **`then`**–parameter) or reducing the sum of all σ–distances of references (called **`dist`**–parameter). These methods of improving legibility of proofs can be formulated as the following two problems:

The 2nd Method of Improving Legibility ($\mathbf{MIL}_{\texttt{then}}$):
INSTANCE: A DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A}\rangle$, a subset $\mathcal{A}_1 \subseteq \mathcal{A}$, and a positive integer $K \leqslant |\mathcal{V}|$.

QUESTION: Does there exist $\sigma \in TS(\mathcal{D})$ for which $|\texttt{then}_{\mathcal{A}_1}(\sigma)| \geqslant K$?

The 3rd Method of Improving Legibility ($\mathbf{MIL}_{\texttt{dist}}$):

INSTANCE: A DAG $\mathcal{D} = \langle \mathcal{V}, \mathcal{A} \rangle$, a subset $\mathcal{A}_2 \subseteq \mathcal{A}$, and a positive integer $K \leqslant \binom{|\mathcal{V}|+1}{3}$.

QUESTION: Does there exist $\sigma \in TS(\mathcal{D})$ for which $\sum_{\langle v,u \rangle \in \mathcal{A}_2} \sigma(u) - \sigma(v) \leqslant K$?

This impact has been studied on the MML database Version `5.22.1191` that includes 208590 one-level deductions. To obtain the result we use a brute-force method to check the existence of a solution for simple instances of this problem (e.g., one-level deductions that have at most 1000000 of possible linearisations) and the SMT-solver Z3 [16] to check more computationally complex ones, since both problems, $\mathrm{MIL}_{\texttt{then}}$ and $\mathrm{MIL}_{\texttt{dist}}$, are NP-complete [20]. There was a time limit of 10 minutes set for each combination of the transformation strategies. With this threshold only 1.92% and 0.03% remained untransformed in `then` and `dist` parameters optimisation, respectively.

Using a polynomial time algorithm that is sketched in Prop. 1, we reduced the number of labeled steps only in 4.49% of deductions. Additionally, these deductions were mainly located in proof scripts recently added to the MML database. This observation is a simple consequence of the fact that the `lab`–parameter in older scripts was reduced in a revision of MML database Version `4.121.1054` and obtained results were introduced to the Version `4.127.1060` [18]. Note that this situation was not intentional, since the main aim of this revision was not to minimise the `lab`–parameter, but generally to increase the `then`–parameter in proof scripts based on a greedy algorithm. However, topological sortings obtained by this algorithm fulfil the conditions (5).

	`then`–parameter	`dist`–parameter
Improved	1.19%	8.27%
Unchanged	76.36%	74.42%
Worsened	22.45%	17.31%

Table 1. Modification of `then` and `dist` parameters obtained by a polynomial time algorithm, sketched in Section 4, in comparison to the initial situation.

	`then`–parameter	`dist`–parameter
Improved	6.02%	18.66%
Unchanged	93.89%	80.89%
Worsened	0.09%	0.45%

Table 2. Modification of `then` and `dist` parameters obtained by a brute-force method or Z3 solver, if we restrict the search to the linearisation with optimal `lab`–parameter, in comparison to the initial situation. Clearly, we limited results to cases, in which at least one strategy solved the problem.

Analysing the impact of this polynomial time algorithm application for `then` and `lab` parameters, we observe that this parameters are more often worsened than improved. These results are summarised in Tab. 1. However, since we can determine efficiently the `lab`–parameter, we explored also the improvement of these parameters among such topological sortings that have optimal `lab`–parameter (see Tab. 2).

It is important to pay attention to the 0.45% percent of deductions for which it is certainly impossible to obtain the optimal value for both, `lab` and `dist` parameters simultaneously. More precisely, the analysis shows also that these 0.45% of deductions constitute only part (18.91%) of 2.37% cases where we cannot obtain the optimal `dist`-parameter if we have optimal `lab`-parameter. The other 79.36% and 1.73% of 2.37% cases are obtained for deductions where, despite optimal value of `lab`-parameter, we can improve or unchange respectively the `dist`-parameter in comparison to the initial situation.

As an illustration of such a conflict between `lab` and `dist` parameters, let us consider a simple abstract proof graph that is presented in Fig. 4. It is easy to see that this graph has exactly two possible linearisations σ: $1, 2, 3, 4$ and $2, 1, 3, 4$. Additionally, the number od σ-labeled steps is equal to 2 and 1 respectively, but the sum of all σ–distances of references is equal to 6 and 7, respectively.

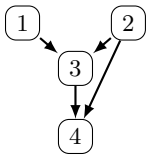


Fig. 4. A simple abstract proof graph for which the conflict between `lab` and `dist` parameters occurs.

Let us note also that the situation where we have to reduce `then`-parameter to obtain optimal `lab`-parameter is a rare situation that occurs mainly in a complex one-level deductions of MML. Additionally, existing examples of this conflict have more than one million of possible linearisations. However, to illustrate the conflict between `lab` and `dist` parameter, we can consider an artificial simple abstract proof graph, presented in Fig. 5 that can potentially occur in Mizar proof scripts. Based on the analysis carried out in Section 4 and Prop. 1 we infer that every topological sorting of this deduction has to have at last 5 labeled steps and this value is obtained, e.g., for an arrangement $1, 2, 4, 5, 7, 8, 3, 6, 9$. Indeed, vertices $1, 2, 4, 5$ have to be labelled, since there exist at least two tail endpoints references arc adjacent to these vertices. Additionally, at most one of the references to premises that correspond to vertices $6, 8$ can be realised without a label. The analysis of all possible, 42, topological sortings shows that the maximal value of `then`-parameter is equal to 6 and it is obtained in exactly two arrangements $1, 2, \ldots, 9$ and $1, 4, 7, 2, 5, 8, 3, 6, 9$ where there exist exactly 6 labeled steps. This shows that the conflict between `lab` and `then` parameters can occur even in "small" one-level deductions.

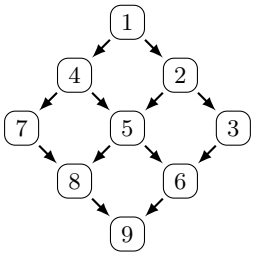


Fig. 5. A simple abstract proof graph for which the conflict between `lab` and `then` parameter occurs.

5 Conclusion

In this paper we have focused on reducing the number of labels introduced to formal reasoning in the process that improves legibility of proof scripts. We have showed that such a reduction can be NP-hard in a formal proof checking environment, even if it is computationally easy in another. Additionally, initial experiments with proof scripts occurring in the MML database show that optimisation of the labeled steps number can be in a conflict with other frequently employed methods of improving proof legibility. However, the presented research shows that this conflict occurs not so often and it appears mainly in deductions that have complex structures.

References

1. Bancerek, G., Hryniewicki, K.: Segments of Natural Numbers and Finite Sequences. Formalized Mathematics (1), 107–114 (1990)
2. Blanchette, J.C.: Redirecting Proofs by Contradiction. In: Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013. EPiC Series, vol. 14, pp. 11–26. EasyChair (2013)
3. Cowan, N.: Attention and Memory: An Integrated Framework. Oxford University Press (1998)
4. Cowan, N.: The magical number 4 in short-term memory: A reconsideration of mental storage capacity. Behavioral and Brain Sciences 24(1), 87–114 (2001)
5. Davis, M.: Obvious Logical Inferences. In: Proc. of the 7th International Joint Conference on Artificial Intelligence. pp. 530–531. William Kaufmann (1981)
6. Ericsson, K.A.: Analysis of memory performance in terms of memory skill, Advances in the psychology of human intelligence, vol. 4. Hillsdale, NJ: Lawrence Erlbaum Associates Ins. (1988)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. A Series of Books in the Mathematical Science, W. H. Freeman and Company, New York (1979)
8. Gonthier, G.: Formal Proof—The Four-Color Theorem. Notices of the AMS 55(11), 1382–1393 (2008)
9. Grabowski, A., Korniłowicz, A., Naumowicz, A.: Mizar in a Nutshell. Journal of Formalized Reasoning 3(2), 153–245 (2010)

10. Grabowski, A., Schwarzweller, C.: Revisions as an Essential Tool to Maintain Mathematical Repositories. In: Proceedings of the 14th symposium on Towards Mechanized Mathematical Assistants: 6th International Conference, Lecture Notes in Computer Science, vol. 4573. pp. 235–249. Springer-Verlag (2007)
11. Grabowski, A., Schwarzweller, C.: Improving Representation of Knowledge within the Mizar Library. Studies in Logic, Grammar and Rhetoric 18(31), 35–50 (2009)
12. Grabowski, A., Schwarzweller, C.: On duplication in mathematical repositories. In: Autexier, S., Calmet, J.e.a. (eds.) Intelligent Computer Mathematics, 10th International Conference, AISC 2010, 17th Symposium, Calculemus 2010, and 9th International Conference, MKM 2010, Lecture Notes in Artificial Intelligence, vol. 6167, pp. 300–314. Springer-Verlag (2010)
13. Kaliszyk, C., Urban, J.: PRocH: Proof Reconstruction for HOL Light. In: Bonacina, M.P. (ed.) 24th International Conference on Automated Deduction, CADE-24. Lecture Notes in Computer Science, vol. 7898, pp. 267–274. Springer-Verlag (2013)
14. Matuszewski, R.: On Automatic Translation of Texts from Mizar-QC language into English. Studies in Logic, Grammar and Rhetoric 4 (1984)
15. Miller, G.A.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. Psychological Review 63, 81–97 (1956)
16. de Moura, L., Bjørner, N.: Z3: An effcient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008. Lecture Notes in Computer Science, vol. 4963, pp. 337–340. Springer-Verlag (2008)
17. Naumowicz, A., Korniłowicz, A.: A Brief Overview of Mizar. In: TPHOLs'09, Lecture Notes in Computer Science, vol. 5674. pp. 67–72. Springer-Verlag (2009)
18. Pąk, K.: The Algorithms for Improving and Reorganizing Natural Deduction Proofs. Studies in Logic, Grammar and Rhetoric 22(35), 95–112 (2010)
19. Pąk, K.: Methods of Lemma Extraction in Natural Deduction Proofs. Journal of Automated Reasoning 50(2), 217–228 (2013)
20. Pąk, K.: The Algorithms for Improving Legibility of Natural Deduction Proofs. Ph.D. thesis, University of Warsaw (2013)
21. Rudnicki, P.: Obvious Inferences. Journal of Automated Reasoning 3(4), 383–393 (1987)
22. Smolka, S.J., Blanchette, J.C.: Robust, Semi-Intelligible Isabelle Proofs from ATP Proofs. In: Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013. EPiC Series, vol. 14, pp. 117–132. EasyChair (2013)
23. Trybulec, A., Korniłowicz, A., Naumowicz, A., Kuperberg, K.: Formal mathematics for mathematicians. Journal of Automated Reasoning 50(2), 119–121 (February 2013), http://dx.doi.org/10.1007/s10817-012-9268-z
24. Wenzel, M.: The Isabelle/Isar Reference Manual. University of Cambridge (2013), http://isabelle.in.tum.de/dist/Isabelle2013-2/doc/isar-ref.pdf
25. Wicknes, D.D., Born, D.G., Allen, C.K.: Proactive Inhibition and Item Similarity in Short-Term Memory. Journal of Verbal Learning and Verbal Behavior 2, 440–445 (1963)

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

On Logical and Semantic Investigations in Kyiv School of Automated Reasoning

Alexander Lyaletski and Mykola Nikitchenko

Faculty of Cybernetics, Taras Shevchenko National University of Kyiv, Ukraine
lav@unicyb.kiev.ua nikitchenko@unicyb.kiev.ua

Abstract. The paper contains a brief description of the logical and semantic approaches to automated reasoning that can be traced back to 1962, when academician V. Glushkov initiated research on automated theorem proving. Carried out first at the Institute of Cybernetics of NASU, in 1998 these investigations were moved to the Faculty of Cybernetics of the Taras Shevchenko National University of Kyiv, where now they are being developed in two directions. The first direction is the traditional one, centered mainly on the construction of proof methods in various first-order logics including the resolution-type technique. The second direction aims to develop logics based on the principles of compositionality and nominativity. Such logics are oriented towards semantic models of programs represented by special algebras. Currently, work is being done to consolidate the investigations carried out in these directions. The main results obtained by the authors and the state of art of their research in automated reasoning are presented.

1 Introduction

The turn of the 1960s is characterized by the appearance of the first Western papers on the problem of automated theorem proving, focused on the construction of automatic provers (see, for example [1, 47]; more information can be found in [2]). About the same time, in Ukraine, Academician Victor Glushkov proposed an approach to its solution called the Evidence Algorithm (EA in accordance with [3]). It includes: a formal language designed for writing mathematical texts and close to natural language; the evolutionary development of methods for automated theorem proving; the construction of an information environment (a mathematical knowledge base in modern terminology) that contains the data necessary for a successful search of a theorem's proof; and the interface between man and computer. The current state of research in the field of automated theorem proving has confirmed the viability of Glushkov's approach and many of the systems, oriented to a comprehensive logical processing of mathematical (or, more generally, formal) texts, implemented it in one way or another.

Historically, the time span of automating theorem proving in Kyiv (Kiev) can be divided into four stages [4]: 1962-1969, when the first steps were made in this field; 1970-1982, when such studies were carried out in accordance with the so-called Evidence Algorithm programme initiated by V. Glushkov, which led to the design and implementation of the Russian-language SAD system (1978); 1983-1992, when the Russian SAD system was improved by means of the building-in of human-like proof technique into it, and 1998 - present, when, after a certain period of absence of any

research on the EA-style reasoning, Glushkov's position has been revised in the light of the new paradigms and advances in automated reasoning, which led to the construction of the English-language SAD system (2002) [5] accessible on the site "nevidal.org".

In this connection, the following should be noted. In Ukraine, all the logical studies related to the first three stages of the EA investigations were performed at the Institute of Cybernetics of NASU, while since 1998 they completely moved to the Faculty of Cybernetics of the Taras Shevchenko National University of Kyiv, where they proceeded in two directions. One continues the research done by the first author since the 1970s, and the other, initiated by V. Glushkov in his works on systems of algorithmic algebras, has been developed by the second author since the 1990s.

The paper consists of two main sections. The first section is written by Alexander Lyaletski and devoted to the description of his results connected with EA. The second one is written by Mykola Nikitchenko and devoted to program-oriented algebra-based logics.

The standard terminology for first-order logics is used.

2 EA Deduction in First-order Logics

The first author started his logical investigations in the framework of EA in the 1970s. They can be divided into three groups: resolution-type methods, interpretation of Maslov's Inverse Method, and sequent inference search in the EA-style.

These investigations have already been presented in [6] in a complete enough form. That is why we are focusing here only on the description of the main approaches and corresponding method and results.

2.1 Resolution Technique and Maslov's Inverse Method

This subsection concerns the problem of the establishing of unsatisfiability in classical logic on the basis of resolution technique [7] and Maslov's Inverse Method (MIM) [8].

Obviously, we can restrict ourselves to the consideration of closed formulas only. It is known that the problem of the establishing of the validity of a set $\{F_1, \ldots, F_m\}$ of formulas is equivalent to the deducibility of a sequent S of the form $\to F_1, \ldots, F_m$ ($m \geqslant 1$) in any complete and sound sequent calculus. Moreover, for any first-order sequent $\to F_1, \ldots, F_m$ being investigated for deducibility, any F_i can be reduced to its (closed) Skolem functional form containing only negative [9] quantifiers. That is, F_i may be considered as a formula presented in the form $\exists x_{i,1} \ldots \exists x_{i,m_i} G_i(x_{i,1}, , x_{i,m_i})$, where $x_{i,1}, , x_{i,m_i}$ are all its variables and each $G_i(x_{i,1}, , x_{i,m_i})$ is presented in conjunctive normal form, which leads to the notions and denotations given below.

2.1.1 Resolution-type Technique

Following tradition, we move to reasoning on satisfiability replacing the establishing of the deducibility of a sequent $\to F_1, \ldots, F_n$ (where $F_1, \ldots, F_n$ are quantifier-free formulas in conjunctive normal form) by the establishing of the unsatisfiability of the set $\{\neg F_1, \ldots, \neg F_n\}$. As a result, we can assume that $\neg F_i$ is presented in disjunctive normal form, each variable of which is bounded by a universal quantifier.

A literal is an atomic formula A or its negation $\neg A$. If a literal L is of the form $\neg A$, its *complementary* $\tilde{L}$ is A; otherwise, $\tilde{L}$ presents $\neg A$.

If $L_1, \ldots, L_n$ are literals, then the expression $L_1 \wedge \ldots \wedge L_n$ $(L_1 \vee \ldots \vee L_n)$ is called a *conjunct* (a *disjunct*).

Any conjunct (or disjunct) is identified with the set of its literals[1].

An expression of the form $C_1 \vee \ldots \vee C_n$ $(D_1 \wedge \ldots \wedge D_n)$, where $C_1, \ldots, C_n$ are conjuncts ($D_1, \ldots, D_n$ are disjuncts), is called a *conjunctive disjunct* or *c-disjunct* (*disjunctive conjunct* or *d-conjunct*).

Any c-disjunct (d-conjunct) is identified with the set of its conjuncts (disjuncts). Therefore, the order of the writing of its conjuncts (disjuncts) is immaterial.

A c-disjunct (d-conjunct) not containing any conjunct (disjunct) is called an *empty* disjunct (conjunct) and denoted by $\emptyset$.

The definition of a c-disjunct (d-conjunct) as a first-order formula allows us to use all the semantic notions of first-order classical logic for c-disjuncts (d-conjuncts) and sets of c-disjuncts (sets of d-conjuncts) on the understanding that each variable in any c-disjunct (d-conjunct) is considered to be bounded by the universal quantifier in the case of reasoning on unsatisfiability and by the existential quantifier in the case of reasoning on validity.

As we see, the notions of c-disjuncts and d-conjuncts are "dual" in the sense that replacing the disjunction sign by the conjunction sign and vice versa, we transform the problem of the establishing of the unsatisfiability of a set of c-disjuncts to the problem of the establishing of the validity of the set of the "dual" d-conjuncts taking into mind the quantifier boundness of variables. This "duality" can be extended to the rules of the described calculi of c-disjuncts transforming the calculi of c-disjuncts to the calculi of d-conjuncts for to the establishing of the validity of sets of d-conjuncts. (We leave such a re-wording to a reader as an exercise.)

Having the notion of a c-disjunct, we can introduce two different inference rules, thereby determining two calculi of c-disjuncts which use the standard notion of a most general simultaneous unifier[2] (mgsu) of sets of expressions [10].

The deducibility of a c-disjunct D from a set M of c-disjuncts in any c-disjunct calculus is understood as the existence of a sequence $D_1, \ldots, D_n$, such that D_n is D and each D_i $(1 \leqslant i \leqslant n)$ is a variant of a c-clause from M or can be deduced from variants of early deduced c-clauses.

We start with the consideration of a calculus based on the analog of Robinson's clash-resolution rule [11].

CR-rule. Let c-disjuncts $D_0, D_1, \ldots, D_q$ $(q \geqslant 1)$, not containing common variables in pairs, be of the forms $D'_0 \vee M_{1,1} \vee \ldots \vee M_{1,r_1} \vee \ldots \vee M_{q,1} \vee \ldots \vee M_{q,r_q}$, $D'_1 \vee C_{1,1} \vee \ldots \vee C_{1,p_1}$, …, $D'_q \vee C_{q,1} \vee \ldots \vee C_{q,p_q}$, respectively, where $D'_0, \ldots, D'_q$ are c-clauses and $M_{1,1}, \ldots, M_{q,r_q}, \ldots, C_{1,1}, \ldots C_{q,p_q}$ are conjuncts. Suppose that $M_{1,1}, \ldots, M_{q,r_q}$ contain literals $L_{1,1}, \ldots, L_{q,r_q}$ respectively, and for each j $(1 \leqslant j \leqslant q)$ $C_{j,1}, \ldots C_{j,p_j}$ contain literals $E_{j,1}, \ldots E_{j,p_j}$ respectively, such that there exists the mgsu σ of the sets $\{L_{1,1}, \ldots, L_{1,r_1}, \tilde{E}_{1,1}, \ldots \tilde{E}_{1,p_1}\}$, …, $\{L_{q,1}, \ldots, L_{q,r_q}, \tilde{E}_{q,1}, \ldots \tilde{E}_{q,p_q}\}$. Then the c-disjunct $D'_0 \cdot \sigma \vee D'_1 \cdot \sigma \vee \ldots \vee D'_q \cdot \sigma$ is said to be deduced from $D_0, D_1, \ldots, D_q$ by the rule CR.

The following result was proved in [12] for the c-disjunct calculus with the CR-rule.

Theorem 1. *(Soundness and completeness w.r.t. CR.) A set M of c-clauses is unsatisfiable if, and only if, $\emptyset$ is deducible from M in the c-disjunct calculus with CR.*

[1] Thus, a disjunct is the same as a clause in the resolution method [7].

[2] We use the usual notion of a substitution; the result of the application of a substitution σ to an expression Ex is denoted by $Ex \cdot \sigma$.

Now, let us construct another c-disjunct calculus.

IR-rule. Let c-disjuncts $D_0, D_1, \ldots, D_q$ $(q \geqslant 1)$, not containing common variables in pairs, be of the forms $D'_0 \vee M_1 \vee \ldots \vee M_q$, $D'_1 \vee C^1_{1,1} \vee \ldots \vee C^1_{1,p_{1,1}} \vee C^{r_1}_{1,1} \vee \ldots \vee C^{r_1}_{1,p_{1,r_1}}, \ldots$, $D'_q \vee C^1_{q,1} \vee \ldots \vee C^1_{q,p_{q,1}} \vee C^{r_q}_{q,1} \vee \ldots \vee C^{r_q}_{q,p_{n,r_n}}$ respectively, where $D'_0, \ldots, D'_q$ are c-clauses and $M_1, \ldots, M_q, C^1_{1,1}, \ldots C^{r_q}_{q,p_{n,r_n}}$ are conjuncts. Suppose that for each j $(1 \leqslant j \leqslant q)$ M_j contains literals $L_{j,1}, \ldots, L_{j,r_j}$ and $C^1_{j,1}, \ldots, C^1_{j,p_{j,1}}, \ldots, C^{r_j}_{j,1}, \ldots, C^{r_j}_{j,p_{j,1}}$ contain literals $E^1_{j,1}, \ldots, E^1_{j,p_{j,1}}, \ldots, E^{r_j}_{j,1}, \ldots, E^{r_j}_{j,p_{j,1}}$ respectively, such that there exists the mgsu σ of the sets $\{\tilde{L}_{1,1}, E^1_{1,1}, \ldots, E^1_{1,p_{1,1}}\}, \ldots, \{\tilde{L}_{1,r_1}, E^{r_1}_{1,1}, \ldots, E^{r_1}_{1,p_{1,r_1}}\}, \ldots, \{\tilde{L}_{q,1}, E^1_{q,1}, \ldots, E^1_{q,p_{q,1}}\}, \ldots, \{\tilde{L}_{q,r_q}, E^{r_q}_{q,1}, \ldots, E^{r_q}_{q,p_{q,r_q}}\}$. Then the c-disjunct $D'_0 \cdot \sigma \vee D'_1 \cdot \sigma \vee \ldots \vee D'_q \cdot \sigma$ is said to be inferred from $D_0, D_1, \ldots, D_q$ by the rule IR.

The following result was proved in [13] for the c-disjunct calculus with the IR-rule.

Theorem 2. *(Soundness and completeness w.r.t. IR.) A set M of c-clauses is unsatisfiable if, and only if, $\emptyset$ is deducible from M in the c-disjunct calculus with IR.*

Example 1. Let $M = \{(A \wedge \neg A) \vee (B \wedge C) \vee (E \wedge L), \neg B \vee \neg C, \neg E \vee \neg L\}$, where A, B, C, E, and L are atomic formulas.

The (minimal) inference in the calculus with CR is as follows:

1. $(A \wedge \neg A) \vee (B \wedge C) \vee (E \wedge L)$ $(\in S)$,
2. $(A \wedge \neg A) \vee (B \wedge C) \vee (E \wedge L)$ $(\in S)$,
3. $(B \wedge C) \vee (E \wedge L)$ (from (1) and (2) by CR),
4. $(B \wedge C) \vee (E \wedge L)$ (a variant of (3)),
5. $\neg B \vee \neg C$ $(\in S)$,
6. $E \wedge L$ (from (5), (3), and (4) by CR),
7. $E \wedge L$ (a variant of (6)),
8. $\neg E \vee \neg L$ $(\in S)$,
9. $\emptyset$ (from (8), (6), and (7) by CR).

The (minimal) inference in the calculus with IR is as follows:

1. $(A \wedge \neg A) \vee (B \wedge C) \vee (E \wedge L)$, $(\in S)$,
2. $(A \wedge \neg A) \vee (B \wedge C) \vee (E \wedge L)$, $(\in S)$,
3. $(B \wedge C) \vee (E \wedge L)$ (from (1) and (2) by IR),
4. $\neg B \vee \neg C$, $(\in S)$,
5. $\neg E \vee \neg L$, $(\in S)$,
6. $\emptyset$ (from (3), (4), and (5) by IR)

Using Theorem 1 or Theorem 2, we obtain the unsatisfiability of M.

We draw your attention to the fact that in these examples the inference with IR is shorter than the inference with CR, which concerns both the number of c-disjuncts and the number of inference rule applications in the inferences. This demonstrates that calculi with CR and IR, in spite of having similar definitions, behave very differently in some specific situations, which was tracked in [14].

The calculi with CR and IR admit many of the (sound and complete) strategies that take place for the usual class-resolution such as, for example, binary resolution, linear resolution, and positive and negative hyper-resolution (see [12] and [13]).

For the case of classical logic with equality (denoted by $\simeq$) we can incorporate the usual paramodulation rule in the calculi with CR and IR.

Paramodulation rule (PP). Suppose we have two c-clauses D and $C \vee (K \wedge s \simeq t)$, where C is a c-clause and K conjunct (possibly, empty). If there exists mgsu σ of the

set of terms $\{s, u\}$, where u is a term occurring in D at a selected position, then the c-clause $C \cdot \sigma \vee (D \cdot \sigma)[t \cdot \sigma]$ is said to be deduced from the given c-clauses by the PP-*rule*, where $(D \cdot \sigma)[t \cdot \sigma]$ denotes the result of the replacement of the term $u \cdot \sigma$ being at the selected position in $D \cdot \sigma$ by $t \cdot \sigma$.

The addition of the PP-rule to the calculi with CR and IR leads to their complete paramodulation extensions in all the cases where such extensions are complete for the usual clash-resolution, binary resolution, and their strategies, thus avoiding the need for functionally reflexive axioms where it is possible to do so, in the case of the usual clash-resolution. The proof of these facts can be made, for example, on the basis of the ideas used in [13, 14] for the proof of the completeness of CR and IR for sets of c-clauses without equality.

2.1.2 Maslov's Inverse Method

In 1964, Sergei Maslov (the USSR) proposed his inverse method (MIM) for the establishment of the deducibility of sequents of the form $\rightarrow D_1, \ldots, D_n$ in first-order classical logic (without equality), where $D_1, \ldots, D_n$ are formulas presented in disjunctive normal form, that is they are d-conjuncts under the condition that each of their variables is bounded by an existential quantifier. The method was defined as a special calculus of so-called favorable assortments [8]. The description of his method was made in terms that did not correspond to traditional logical terminology applied at that time (including resolution terminology). That is why the problem of the interpretation of the scheme of MIM operation in resolution terms or other commonly-used terms was raised.

In [15], S. Maslov himself gave a MIM interpretation in the resolution notions for a restricted case. Later, the first author of this paper proposed a MIM interpretation in the terms of the c-disjunct calculus with the IR-rule [13]. In 1989, V. Lifschitz, independently introducing the notion of a c-disjunct under the name of super-clause, improved such interpretation [16]. An interpretation of MIM closer to its definition in [8] was made in [17] for the case where a set of clauses (disjuncts in our terminology), maybe, with equality is taken as "input" for MIM.

In what follows, MIM is treated as a special calculus of so-called *favorable sequents* having the most adequate interpretation of MIM in comparison with all the other interpretations of MIM in the form from [8]. (Reasoning is made by validity i.e. deducibility.)

Suppose D is a d-conjunct $C_1 \wedge \ldots \wedge C_n$, where $C_1, \ldots, C_n$ are disjuncts, $x_{i,1}, \ldots, x_{i,m_i}$ are all the variables from C_i, and P_{C_i} is a new m_i-arity predicate symbol $(1 \leqslant i \leqslant n)$. Then $P_{C_i}(x_{i,1}, \ldots, x_{i,m_i})$ is called an *abbreviation* (*name* in the terminology of [17]) for C_i.

Let D_1 and D_2 be d-conjuncts and P_{C_1} and P_{C_2} be abbreviations for C_1 from D_1 and C_1 from D_1 respectively (D_1 and D_2 can coincide). Suppose there exists the mgsu σ of a set $\{L_1, \tilde{L}_2\}$, where L_1 is a literal from C_1 and L_2 is a literal from C_2. Then $P_{C_1} \cdot \sigma \vee P_{C_2} \cdot \sigma$ is called a *favorable disjunct* for D_1 and D_2. In the case of the coincidence of D_1 and D_2, $P_{C_1} \cdot \sigma$ is a *favorable disjunct* for D_1.

Obviously, there is only a finite set of favorable disjuncts for any d-conjuncts D_1 and D_2. The same concerns a finite set of d-conjuncts.

We determine the *favorable sequent calculus* in the following way (as in the case of [8], this calculus has two inference rules: **A** and **B**).

We are interested in establishing of the validity of a set $\{D_1, \ldots, D_n\}$ (i.e. in the deducibility of the sequent $\to D_1, \ldots, D_n$), where $D_1, \ldots, D_n$ are d-conjuncts not having pairwise common variables.

Rule **A** (for generating a starting favorable sequent). Let $\{D_1, \ldots, D_n\}$ be a set of d-conjuncts being investigated for validity and D_i be of the form $C_{i,1} \wedge \ldots \wedge C_{i,m_i}$, where $C_{i,1}, \ldots, C_{i,m_i}$ are disjuncts ($i = 1, \ldots, n$). If $P_{C_{1,1}}$, ..., $P_{C_{n,m_n}}$ are abbreviations of $C_{i,1}, \ldots, C_{n,m_n}$ respectively and Γ is the multiset of all favorable disjuncts constructed by using $D_1, \ldots, D_n$, then $\Gamma \to P_{C_{1,1}} \wedge \ldots \wedge P_{C_{1,m_1}}, \ldots, P_{C_{n,1}} \wedge \ldots \wedge P_{C_{n,m_n}}$ is called a *starting favorable sequent* deduced from $D_1, \ldots, D_n$ by the rule **A**.

Remark. A starting sequent does not contain the negation symbol. Moreover, its antecedent contains favorable disjuncts with only one or two literals and its succedent contains only conjuncts of atomic formulas.

Rule **B**. Let $\Gamma \to \Delta$ be a favorable sequent (i.e. it is already deduced by the rule **A** or **B**). Suppose there are disjuncts $B_1 \vee L_1, \ldots, B_n \vee L_n$ in Γ, where $B_1, \ldots, B_n$ are disjuncts and $L_1, \ldots, L_n$ are literals, and Δ contains a conjunct $P_{D_1} \wedge \ldots \wedge P_{D_n}$, such that there exists the mgsu σ of the set $\{P_{D_1}, L_1\}, \ldots, \{P_{D_n}, L_n\}$. Then, for any variant D of the disjunct $B_1 \cdot \sigma \vee \ldots \vee B_n \cdot \sigma$, the sequent $\Gamma, D \to \Delta$ is called a *favorable sequent* deduced from $\Gamma \to \Delta$ by the rule **B**.

Any process of inference search in the favorable sequent calculus starts with the generation of a starting favorable sequent S by the rule **A**. Subsequent favorable sequents can be deduced only by the rule **B**. The inference process is considered to be successful, if an sequent containing $\emptyset$ in its antecedent is deduced. Such a sequent is called a *final favorable sequent*.

Theorem 3. *A set of d-conjuncts $\{D_1, ..., D_k\}$ is valid in first-order classical logic if, and only if, a final favorable sequent is deducible from the starting favorable sequent for $\{D_1, ..., D_k\}$ in the favorable sequent calculus.*

Example 2. Let us return to Example 1 and establish the unsatisfiability of M transforming M into its "dual" image M' containing d-conjuncts: $M' = \{(\neg A \vee A) \wedge (\neg B \vee \neg C) \wedge (\neg E \vee \neg L),\ B \wedge C,\ E \wedge L\}$.

Let us introduce the following abbreviations: P_1 for $\neg A \vee A$, P_2 for $\neg B \vee \neg C$, P_3 for $\neg E \vee \neg L$, G_1 for B, G_2 for C, G_3 for E, and G_4 for L.

We can construct the following inference in the favorable sequent calculus (we underline objects participating in applications of the rule **B**) :

1. $P_1,\ \underline{P_2 \vee G_1},\ \underline{P_2 \vee G_2},\ P_3 \vee G_3,\ P_3 \vee G_4 \to P_1 \wedge P_2 \wedge P_3,\ \underline{G_1} \wedge \underline{G_2}, G_3 \wedge G_4$
   (by **A**, a starting favorable sequent),
2. $P_1,\ P_2 \vee G_1,\ P_2 \vee G_2,\ \underline{P_3 \vee G_3},\ \underline{P_3 \vee G_4},\ P_2 \to P_1 \wedge P_2 \wedge P_3,\ G_1 \wedge G_2, \underline{G_3} \wedge \underline{G_4}$
   (from (1) by **B**),
3. $\underline{P_1},\ P_2 \vee G_1,\ P_2 \vee G_2,\ P_3 \vee G_3,\ P_3 \vee G_4,\ \underline{P_2}, \underline{P_3} \to \underline{P_1} \wedge \underline{P_2} \wedge \underline{P_3},\ G_1 \wedge G_2, G_3 \wedge G_4$
   (from (2) by **B**),
4. $P_1,\ P_2 \vee G_1,\ P_2 \vee G_2,\ P_3 \vee G_3,\ P_3 \vee G_4,\ P_2, P_3, \emptyset \to P_1 \wedge P_2 \wedge P_3,\ G_1 \wedge G_2, G_3 \wedge G_4$
   (from (3) by **B**, a final favorable sequent; it contains $\emptyset$).

In accordance with Theorem 3, M' is a valid set. (Hence, M is an unsatisfiable one.)

The problem of the building-in of a equality-handling technique into MIM in the form from [8] remains open. It is expected that the favorable sequent calculus will be able to give a key to its solution.

2.2 Sequent Inference Search in EA-style

From the very beginning of its appearance, the EA program has paid great attention to developing machine proof search methods suitable for the various fields of mathematics and reflecting (informal) human reasoning techniques. The first attempt in this direction was made in 1963, when the problem of automated theorem proving in the group theory was formulated by V. Glushkov. To develop it, texts on the group theory were exposed to the careful analysis. As a result, a machine procedure for proof search in the group theory was constructed in the middle of the 1960s [18].

Later, that procedure was generalized to the case of classical first-order restricted predicate calculus without equality [19, 20]. The procedure admitted its interpretation as a specific, sound and complete, sequent calculus later called the AGS (Auxiliary Goals Search) calculus [21]. The distinctive features of the primary AGS were: (1) goal-orientation (i.e. at every moment of time the succedent of any sequent under consideration had no more than one formula-goal) and (2) specific handling of quantifiers, being actually the independent repetition of Kanger's idea [22] about using of so-called "dummies" and "parameters" in quantifier rules as special variables with subsequent replacing "dummies" by "admissible" terms at a certain time.

Further development of the ASG calculus led to its improvement in the direction of optimizing quantifier handling, separating equality processing from deduction, and making goal-driven proof search.

The optimization of quantifier handling was achieved by the introduction of an original notion of admissible substitutions distinguished from Kanger's.

The equality separation was oriented to the development of special methods for equality processing and equation solving. (Later algebra systems and problem solvers were suggested to use for this purpose.)

The goal-driven search was based on driving the process of an auxiliary goal generation by taking into account a formula (goal) under consideration.

All these investigations led to the construction of an original sequent calculus (with the new notion of admissibility) published in [23] in 1981 and implemented in the (Russian) system for automated deduction SAD. This implementation demonstrated the usefulness of deductive research towards constructing of such calculi.

Since then, the investigations on inference search in EA-style were stopped until 1998, when the first author began to participate in the Intas project 96-0760 "Rewriting Techniques and Efficient Theorem Proving" (1998-2000). The project gave a reason for the modification of the calculus from [23] in several directions (see, for example, [24] for the classical case) and later led to a wide spectrum of interesting results on efficient inference search in the EA-style not only for classical logic, but for intuitionistic one and their modal extensions [25].

Note that the (new) notion of admissibility is not enough for the construction of sound calculi in the intuitionistic case. This situation can be corrected by using the notion of compatibility proposed in [26] for the construction of the sound (and complete) tableau calculus with free variables for intuitionistic logic (some historical details can be found in [27]).

Besides, the usage of the new admissibility led to the possibility of constructing a wide class of Herbrand theorems for classical logic in a form not requiring preliminary skolemization. The exhaustive research on this topic was published in [28], where the sequent approach was used for obtaining the main results.

The sequent approach developed in [29] also was used in [30] for the construction of new complete resolution strategies containing a special resolution and factorization.

The paramodulation rule can be incorporated into these strategies for the case of equality, but the completeness of such extensions is observed only in the case of adding functional reflexivity axioms. The research of the paper [30] also demonstrates the possibility of applying a special paramodulation technique in the model elimination method [31] (with additional functional reflexivity axioms to provide completeness).

As for SAD, its logical "engine" can be considered a modification of a connection tableaux calculus based on the model elimination method. Its further development is aimed at the construction of tools for inference search in non-classical logics.

3 Program-oriented Algebras and Logics

Logical investigations at the Department of Theory and Technology of Programming of the Faculty of Cybernetics were driven by the necessity to develop formal program semantics and methods for program reasoning. These tasks were initiated by V. Glushkov who in his paper [32] proposed an algebra-based formalism for program formalization called *systems of algorithmic algebras*. Later these ideas were elaborated by V. Redko [33, 34] who combined them with ideas developed by V. Korolyuk and E. Yushchenko in their works on *address programming* [35].

An approach proposed by V. Redko was called *composition programming* and aimed at constructing formal models of programming languages.

The main semantic notions of composition programming are *data*, *functions*, and *compositions*. In the simplest case, data are considered as *named sets* of the form $\{(v_1, d_1), \ldots, (v_n, d_n)\}$, where $v_1, \ldots, v_n$ are names, and $d_1, \ldots, d_n$ are their values respectively. *Hierarchical named data* were also introduced. Program semantics is represented by functions over named data (program functions); compositions are understood as program construction means (operations over program functions).

Therefore, the main attention was paid to the definition of functions and compositions oriented on processing of named data (see details in [34, 43]). Formal models of different programming languages and database query languages were constructed and investigated within composition programming [36].

Composition-nominative approach (CNA) [40] aimed to enhance composition programming in the following directions. First, program data were presented as *intensionalized data* [41]. Such data can be considered as certain objects with prescribed intensions. This idea is similar to the notion of typed data, but the latter is usually understood in the extensional sense while CNA aimed to emphasize intensional features of data.

For such kind of data a special form of *abstract computability* was defined, its properties were investigated, and complete classes of computable functions were constructed [42].

Second, special program algebras called *composition-nominative program algebras* (CNPA) were constructed and investigated.

Third, special logics called *composition-nominative program logics* (CNPL) were defined and investigated.

Now we will formally define a special case of program algebras called *quasiary program algebras*.

3.1 Quasiary Program Algebras

For such algebras, data are treated as nominative sets (nominative data in the general case). *Nominative sets* are defined as partial mappings from a set of names (variables)

to a set of basic values. Such mappings do not have fixed arity and are called *quasiary*. Nominative sets can be also treated as states of program variables (see, for example, [37]). More complex case of hierarchical nominative data is not considered in this paper. Compositionality means that complex functions and predicates are built from simpler ones using compositions. Compositions are the operations of the respective algebras used in defining semantics. Also all mappings are partial (can be undefined on some data).

We start with definitions of nominative sets, quasiary predicates and functions.

The arrows $\xrightarrow{p}$ and $\xrightarrow{t}$ specify the sets of partial and total mappings respectively. Also for an arbitrary partial mapping $f : D \xrightarrow{p} D'$:

- $f(d)\downarrow$ is used to denote that f is defined on data $d \in D$;
- $f(d)\downarrow = d'$ is used to denote that f is defined on data $d \in D$ with a value $d' \in D'$;
- $f(d)\uparrow$ is used to denote that f is undefined on data $d \in D$.

Let V be a set of names (variables). Let A be a set of basic values. Then the class ${}^{V}A$ of *nominative sets* is defined as the class of all partial mappings from the set of names V to the set of basic values A. Thus, ${}^{V}A = V \xrightarrow{p} A$.

Set-like notation for nominative sets is more convenient in some cases. We will use the following notation: $[v_i \mapsto a_i \mid i \in I]$ to describe a nominative set where variables v_i have values a_i respectively. Expression $v_i \mapsto a_i \in_n d$ denotes that $d(v_i)\downarrow = a_i$ or in other words that the value of variable v_i in nominative set d is a_i $(i \in I)$.

One of the main operations is a binary total *overriding* operation that joins two nominative sets taking into account names of the variables, and is defined in the following way:

$$d_1 \nabla d_2 = [v \mapsto a \mid v \mapsto a \in_n d_2 \vee (v \mapsto a \in_n d_1 \wedge \neg \exists a'(v \mapsto a' \in_n d_2))].$$

Informally, this means that all name-value pairs from d_2 and those pairs from d_1 whose names are not defined in d_2 are present in the resulting nominative set.

Let $Bool = \{F, T\}$ be the set of Boolean values. Let $Pr^{V,A} = {}^{V}A \xrightarrow{p} Bool$ be the set of all partial predicates over ${}^{V}A$. Such predicates are called *partial quasiary predicates*. They represent different conditions in programs.

Let $Fn^{V,A} = {}^{V}A \xrightarrow{p} A$ be the set of all partial functions from ${}^{V}A$ to A. Such functions are called *partial ordinary quasiary functions*. They represent different expressions in programs. The term 'ordinary' is used to distinguish ordinary functions from program functions (bi-quasiary functions) that represent programs. This term will usually be omitted.

Let $FPrg^{V,A} = {}^{V}A \xrightarrow{p} {}^{V}A$ be the set of all partial functions from ${}^{V}A$ to ${}^{V}A$. Such functions are called *bi-quasiary functions*. They represent semantics of programs.

Algebras with three presented sets (partial quasiary predicates, partial ordinary quasiary functions, and partial bi-quasiary functions) as algebra carriers can be used to define semantics of programming languages. We distinguish the following three algebras:

- pure quasiary predicate algebras with one sort $Pr^{V,A}$ of quasiary predicates;
- quasiary predicate-function algebras with two sorts: $Pr^{V,A}$ of quasiary predicates and $Fn^{V,A}$ of quasiary functions;
- quasiary program algebras with three sorts: $Pr^{V,A}$ of quasiary predicates, $Fn^{V,A}$ of quasiary functions, and $FPrg^{V,A}$ of bi-quasiary functions.

For pure quasiary predicate algebras the set of compositions includes disjunction $\vee : Pr^{V,A} \times Pr^{V,A} \xrightarrow{t} Pr^{V,A}$ and negation $\neg : Pr^{V,A} \xrightarrow{t} Pr^{V,A}$. Also parametric quantification composition $\exists x : Pr^{V,A} \xrightarrow{t} Pr^{V,A}$ and renomination composition

$R^{\bar{v}}_{\bar{x}} : Pr^{V,A} \xrightarrow{t} Pr^{V,A}$ [44, 45] should be included to the set of compositions (here $\bar{v}$ stands for $v_1, \ldots, v_n$ and $\bar{x}$ for $x_1, \ldots, x_n$). Renomination is a specific new composition for quasiary predicates. Informally, while evaluating $R^{\bar{v}}_{\bar{x}}(p)(d)$ we construct a new nominative set changing in d values of names from $\bar{v}$ with values of corresponding names from $\bar{x}$; then p is evaluated on the obtained nominative set.

For quasiary predicate-function algebras we add ordinary quasiary functions to the scope. Compositions of the pure predicate algebras are extended with parametric compositions of superposition for functions $S^{\bar{x}}_F : (Fn^{V,A})^{n+1} \xrightarrow{t} Fn^{V,A}$ and predicates $S^{\bar{x}}_P : Pr^{V,A} \times (Fn^{V,A})^n \xrightarrow{t} Pr^{V,A}$ [44, 45]. Another composition that has to be added is null-ary parametric composition of denomination $'x : Fn^{V,A}$. Renomination composition can be given as a combination of superposition and denomination compositions, thus $R^{\bar{v}}_{\bar{x}}(p) = S^{\bar{v}}_P(p, 'x_1, \ldots, 'x_n)$. So, renomination compositions can be omitted.

For bi-quasiary functions there are many possible ways to define compositions that provide means to construct complex functions from simpler ones. We have chosen the following compositions [37]: the parametric assignment composition $AS^x : Fn^{V,A} \xrightarrow{t} FPrg^{V,A}$, which corresponds to the assignment operator := ; the identity composition (function) $id : FPrg^{V,A}$, which corresponds to the *skip* operator; the composition of sequential execution $\bullet : FPrg^{V,A} \times FPrg^{V,A} \xrightarrow{t} FPrg^{V,A}$; the conditional composition $IF : Pr^{V,A} \times FPrg^{V,A} \times FPrg^{V,A} \xrightarrow{t} FPrg^{V,A}$, which corresponds to the operator *if_ then_ else*; the cyclic composition (loop) $WH : Pr^{V,A} \times FPrg^{V,A} \xrightarrow{t} FPrg^{V,A}$, which corresponds to the operator *while_ do*.

We also need compositions that could provide possibility to construct predicates describing properties of programs. The main conventional operations of this kind are *Glushkov prediction operation* [32] and a ternary *operation induced by Floyd-Hoare assertions*. The prediction operation is related to the weakest precondition defined by Dijkstra and possibility/necessity operations of dynamic logic [38]. We need modifications of these operations oriented on partial predicates. Obtained compositions are called *preimage predicate transformer composition* $PC : FPrg^{V,A} \times Pr^{V,A} \xrightarrow{t} Pr^{V,A}$ (simply referred to as *preimage composition* [39]) and Floyd-Hoare composition for partial predicates $FH : Pr^{V,A} \times FPrg^{V,A} \times Pr^{V,A} \xrightarrow{t} Pr^{V,A}$ (see detailed explanation in [39]).

The latter composition takes a precondition, a postcondition, and a program as inputs and yields a predicate that represents the respective Floyd-Hoare assertion.

Now we give formal definitions of the compositions of the considered algebras. In the following definitions $p, q, r \in Pr^{V,A}$, $f, g_1, \ldots, g_n \in Fn^{V,A}$, $pr_1, pr_2, pr \in FPrg^{V,A}$, $x \in V$, $\bar{x} = (x_1, \ldots, x_n) \in V^n$, $d \in {}^V A$.

$$(p \vee q)(d) = \begin{cases} T, \text{ if } p(d) \downarrow = T \text{ or } q(d) \downarrow = T, \\ F, \text{ if } p(d) \downarrow = F \text{ and } q(d) \downarrow = F, \\ \text{undefined in other cases.} \end{cases}$$

$$(\neg p)(d) = \begin{cases} T, \text{ if } p(d) \downarrow = F, \\ F, \text{ if } p(d) \downarrow = T, \\ \text{undefined in other cases.} \end{cases}$$

$$(\exists x p)(d) = \begin{cases} T, \text{ if } p(d\nabla x \mapsto a) \downarrow = T \text{ for some } a \in A, \\ F, \text{ if } p(d\nabla x \mapsto a) \downarrow = F \text{ for each } a \in A, \\ \text{undefined in other cases.} \end{cases}$$

$$S^{\bar{x}}_P(p, g_1, \ldots, g_n)(d) = p(d\nabla[x_1 \mapsto g_1(d), \ldots, x_n \mapsto g_n(d)]),$$

$$S_F^{\bar{x}}(f, g_1, \ldots, g_n)(d) = f(d\nabla[x_1 \mapsto g_1(d), \ldots, x_n \mapsto g_n(d)]).$$

$$'x(d) = d(x).$$

$$AS^x(f)(d) = d\nabla[x \mapsto f(d)].$$

$$id(d) = d.$$

$$pr_1 \bullet pr_2(d) = pr_2(pr_1(d)).$$

$$IF(r, pr_1, pr_2)(d) = \begin{cases} pr_1(d), \text{ if } r(d) \downarrow = T \text{ and } pr_1(d) \downarrow, \\ pr_2(d), \text{ if } r(d) \downarrow = F \text{ and } pr_2(d) \downarrow, \\ \text{undefined in other cases.} \end{cases}$$

$$WH(r, pr)(d) = d_n, \text{ if } r(d) \downarrow = T, f(d) \downarrow = d_1, \quad r(d_1) \downarrow = T, f(d_1) \downarrow = d_2, \ldots,$$
$$f(d_{n-1}) \downarrow = d_n, r(d_n) \downarrow = F.$$

$$PC(pr, q)(d) = \begin{cases} T, \text{ if } pr(d) \downarrow \text{ and } q(pr(d)) \downarrow = T, \\ F, \text{ if } pr(d) \downarrow \text{ and } q(pr(d)) \downarrow = F, \\ \text{undefined in other cases.} \end{cases}$$

$$FH(p, pr, q)(d) = \begin{cases} T, \text{ if } p(d) \downarrow = F \text{ or } q(pr(d)) \downarrow = T, \\ F, \text{ if } p(d) \downarrow = T \text{ and } q(pr(d)) \downarrow = F, \\ \text{undefined in other cases.} \end{cases}$$

Floyd-Hoare composition was proved to be monotone and continuous [39]. It is derivable from other compositions by formula $FH(p, pr, q) = p \to PC(pr, q)$, therefore this composition can be omitted from the list of algebra compositions.

Thus, the following quasiary program algebras (for different A) were defined:

$$QPA(V, A) =< Pr^{V,A}, Fn^{V,A}, FPrg^{V,A}; \vee, \neg, \exists x, S_P^{\bar{x}}, S_F^{\bar{x}}, ' x, AS^x, \ id, \bullet, IF, WH, PC > .$$

These program algebras form a semantic basis for the corresponding program logics.

3.2 Quasiary Program Logics

Program logics are defined in a sematic-syntactic style. It means that we first define logic semantics as a class of CNPA of the form $QPA(V, A)$ for different A; then we define a logic language as the set of terms of such algebras over sets Fs of function symbols, Ps of predicate symbols, and Prs of program function symbols; finally, we define formula interpretation mappings. These mappings are based on interpretational mappings for function, predicate, and program function symbols $I^{Fs}{:}Fs \xrightarrow{t} Fn^{V,A}$, $I^{Ps}{:}Ps \xrightarrow{t} Pr^{V,A}$, and $I^{Prs}{:}Prs \xrightarrow{t} Prg^{V,A}$ respectively.

Then we can compositionally construct the interpretational mapping for algebra terms. A quadruple $(QPA(V, A), I^{Fs}, I^{Ps}, I^{Prs})$ is called *a logic model*. A model is determined by a tuple $J^{Fs,Ps,Prs}=(V, A, I^{Fs}, I^{Ps}, I^{Prs})$ called *an interpretation*. In simplified form, interpretations will be denoted as J. Algebra terms over the class of

predicates are called formulas. For an interpretation J and a formula Φ the meaning of Φ in J is denoted Φ_J.

A formula Φ is called *satisfiable in an interpretation* J if there is $d \in^V A$ such that $\Phi_J\ (d)\downarrow = T$. A formula Φ is called *satisfiable* if there exists an interpretation J in which Φ is satisfiable. We call formulas Φ and Ψ *equisatisfiable* if they are either both satisfiable or both not satisfiable (i.e. unsatisfiable).

The problem under investigation is to check whether CNPL formula Φ is satisfiable. Our main aim is to transform Φ to an equisatisfiable formula of the classical first-order logic with equality so that we can use existing methods for solving this problem developed for classical logic.

In this paper we consider formulas with loop-free programs only. For programs with loops, their loop-free approximations may be often considered. This is possible because program compositions are continuous.

The required transformation is made in two steps. First, for formulas with loop-free programs we inductively eliminate the preimage composition PC using the following reductions:

$$PC(id, q) = q,$$
$$PC(AS^x(h), q) = S_P^x(q, h),$$
$$PC(pr_1 \bullet pr_2, q) = PC(pr_1, PC(pr_2, q)),$$
$$PC(IF(r, pr_1, pr_2), q) = (r \to PC(pr_1, q)) \wedge (\neg r \to PC(pr_2, q)) \wedge (r \to r).$$

Due to these reductions we obtain formula Ψ of composition-nominative quasiary predicate logic that is equisatisfiable with Φ.

Second, we transform the formula Ψ to a special normal form, which is translated to its classical counterpart Ψ_{CL} afterwards. This reduction we describe in more detail.

As there is no distributivity of existential quantifier with superposition compositions [45] we need an infinite set of unessential variables in order to be able to carry out equivalent transformations of formulas. We assume that a set U of unessential variables is an infinite subset of V ($U \subseteq V$). Informally speaking, this restricts the class of possible interpretations but does not affect the satisfiability problem [45].

A formula Ψ is said to be in *unified superposition normal form* (USNF) if the following requirements are met:

- for every sub-formula of the form $S_P^{\bar v}(\Psi', \bar t)$ we have that $\Psi' \in Ps$;
- for every sub-formula of the form $S_F^{\bar v}(t, \bar t)$ we have that $t \in Fs$;
- all instances of superposition compositions have the same list of variables $\bar w$;
- for every quantifier $\exists y$ occurring in Ψ we have that y should occur in $\bar w$ (see the previous rule).

Consider transformation rules T1–T12 of the form $\Psi_l \mapsto \Psi_r$. These rules are equivalent transformations [44].

T1) $S_P^{\bar v}(\Psi \vee \Psi, \bar t) \mapsto S_P^{\bar v}(\Psi, \bar t) \vee S_P^{\bar v}(\Psi, \bar t)$.

T2) $S_P^{\bar v}(\neg\Psi, \bar t) \mapsto \neg S_P^{\bar v}(\Psi, \bar t)$.

T3) $S_P^{\bar v}(\exists x\ \Psi,\ \bar t) \mapsto \exists u\ S_P^{\bar v}(S_P^x(\Psi, {}'u),\ \bar t)$, u is an unessential variable that does not occur in $S_P^{\bar v}(\exists x\ \Psi,\ \bar t)$, $u \in U$.

T4) $S_P^{\bar u, \bar x}(S_P^{\bar x, \bar v}(\Psi,\ \bar r, \bar s),\ \bar t, \bar w) \mapsto S_P^{\bar u, \bar x, \bar v}(\Psi,\ \bar t,\ S_F^{\bar u, \bar x}(r_1, \bar t,\ \bar w),\ \dots\ ,\ S_F^{\bar u, \bar x}\ (r_k\ ,\ \bar t,\ \bar w),\ S_F^{\bar u, \bar x}(s_1,\ \bar t,\ \bar w),\ \dots\ ,\ S_F^{\bar u, \bar x}(s_m,\ \bar t,\ \bar w))$, here (and in T5) $\bar u = u_1,\dots,u_n$; $\bar t = t_1,\dots,t_n$; $\bar x = x_1,\dots,x_k$; $\bar r = r_1,\dots,r_k$; $\bar w = w_1,\dots,w_k$; $\bar v = v_1,\dots,v_m$; $\bar s = s_1,\dots,s_m$, $u_i \neq v_j$, $i = 1, \dots, n, j = 1, \dots, m$.

T5) $S_F^{\bar u, \bar x}(S_F^{\bar x, \bar v}(t, \bar r, \bar s), \bar t, \bar w) \mapsto S_F^{\bar u, \bar x, \bar v}\ (t, \bar t, S_F^{\bar u, \bar x}\ (r_1, \bar t, \bar w),\ \dots\ , S_F^{\bar u, \bar x}(r_k\ , \bar t, \bar w), S_F^{\bar u, \bar x}\ (s_1,\ \bar t,\ \bar w),\ \dots\ ,\ S_F^{\bar u, \bar x}(s_m,\ \bar t,\ \bar w))$.

T6) $S^{\bar v}_P(r = s, \bar t) \quad \mapsto S^{\bar v}_F(r, \bar t) = S^{\bar v}_F(s, \bar t)$.
T7) $S^{\bar v}_P(\Psi, \bar t) \quad \mapsto S^{x,\bar v}_P(\Psi, {}'x, \bar t)$, x does not occur in $\bar v$.
In particular $\Psi \mapsto S^{x}_P(\Psi, {}'x)$.
T8) $S^{\bar v}_F(t, \bar t) \quad \mapsto S^{x,\bar v}_F(t, {}'x, \bar t)$, x does not occur in $\bar v$.
In particular, $t \mapsto S^{x}_F(t, {}'x)$.
T9) $S^{\bar u,x,\bar v}_P(\Psi, \bar q, r, \bar s) \mapsto S^{x,\bar u,\bar v}_P(\Psi, r, \bar q, \bar s)$.
T10) $S^{\bar u,x,\bar v}_F(t, \bar q, r, \bar s) \mapsto S^{x,\bar u,\bar v}_F(t, r, \bar q, \bar s)$.
T11) $S^{x,\bar v}_F({}'x, t, \bar r) \mapsto t$.
T12) $S^{\bar v}_F({}'x, \bar r) \mapsto {}' x$, x does not occur in $\bar v$.

Rules T3 and T7 permit to assume w.l.o.g. that all quantified variables in the initial formula are different.

Given an arbitrary formula Ψ we can construct (non-deterministically) its unified superposition normal form $usnf[\Psi]$ by applying rules T1–T12. This transformation is satisfiability-preserving.

In order to reduce the satisfiability problem in the composition-nominative predicate logic to the satisfiability problem for classical logics, we consider algebras defined over extended data set $A^V_\varepsilon = V \xrightarrow{t} A \cup \{\varepsilon\}$. Informally, the additional value ε represents undefined components of nominative data. So, by changing data classes from ${}^V A$ to A^V_ε we make our algebras closer to their classical counterparts and simplify the required proofs.

We formalize the syntactical reduction *clf* of terms and formulas in unified superposition normal form to formulas of classical logic inductively as follows:

1. $clf[{}'x] \mapsto x$.
2. $clf[S^{w_1,\dots,w_n}_F(F, t_1, ..., t_n)] \mapsto F(clf[t_1], ..., clf[t_n])$, $F \in Fs$.
3. $clf[(\Psi_1 \vee \Psi_2)] \mapsto (\, clf[\Psi_1] \ \vee clf[\Psi_2])$.
4. $clf[\neg\Psi] \mapsto \neg clf[\Psi]$.
5. $clf[S^{w_1,\dots,w_n}_P(P, t_1, ..., t_n)] \mapsto P(clf[t_1], ..., clf[t_n])$, $n \geq 0$.
6. $clf[\exists x \Psi] \mapsto \exists x(x \neq e \& clf[\Psi])$, $e \in U$, e is a predefined variable.
7. $clf[t_1 = t_2] \mapsto \ clf(t_1) = clf(t_2)$.

Note that all applications of the 6-th rule introduce the same variable e which is a predefined variable from U in the sense that it does not occur in USNF. In interpretations the value of this variable is treated as ε.

These reductions transform formulas (with loop-free programs) of composition-nominative program logics to formulas of classical predicate logic preserving their satisfiability.

We can also use direct methods to check satisfiability/validity in the defined logics. To do this, different calculi of sequent type were constructed [44, 46].

4 Conclusion

In the 1960s, V. Glushkov initiated a series of investigations on automated reasoning, among which the main attention was paid to the issues connected with deduction constructions in both logic and programming as well as their semantic grounding.

Carried out first at the Institute of Cybernetics of NASU, in 1987 these investigations moved to the Faculty of Cybernetics of the Taras Shevchenko National University of Kyiv, where now they are developing in the logical and semantic directions. As a result, sufficiently deep studies have been made in their frameworks.

The following results have been obtained for the first direction: (1) new calculi and methods including resolution- and paramodulation-type techniques as well as modifi-

cations of Maslov's Inverse Method have been constructed; (2) research on inference search in various first-order classical and non-classical sequent logics oriented to their implementation in automated reasoning systems, in particular, in the English SAD system has been made; (3) as a result, the logical "engine" of the English SAD system has been revised and modified.

The following results have been obtained for the second direction: (1) a hierarchy of algebra-based program models have been constructed; (2) new program-oriented logics of partial predicates and functions based on principles of compositionality and nominativity have been developed; (3) sound and complete sequent calculi have been constructed for these logics; (4) algorithms for reduction of the satisfiability problem in these logics to the satisfiability problem in classical logic have been developed; (5) a quasiary program logic which can be considered as an extension of Floyd-Hoare logic on partial predicates has been defined and investigated.

It is expected that the obtained results will be able to serve as a basis for extending automated reasoning systems with more expressive logical languages and more powerful reasoning tools.

References

1. P. C. Gilmore. A program for the production of proofs for theorems derivable within the first-order predicate calculus from axioms. *Inter. Conf. on Information Processing*, Paris, France; June, 1959.
2. J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 2001.
3. V. M. Glushkov. Some problems in the theories of automata and artificial intelligence. *Cybernetics and System Analysis*, 6(2), pp. 17–27, Springer, New York, 1970.
4. A. Lyaletski and K. Vershinine. *Evidence Algorithm and System for Automated Deduction: A retrospective view.* Proceedings of the 10th International Conference "Intelligent Computer Mathematics", pp. 411–426, 2010.
5. A. Lyaletski, K. Verchinine, A. Degtyarev, and A. Paskevich. System for Automated Deduction (SAD): Linguistic and deductive peculiarities. In M. A. Klopotek, S. T. Wierzchon, and M. Michalewicz, editors, *Intelligent Information Systems, 11th International Symposium, IIS 2002*, Advances in Soft Computing, pp. 413–422, Sopot, Poland, June 2002. Physica-Verlag.
6. A. Lyaletski, Evidence Algorithm and inference search in first-order logics. To appear in a special issue of the Journal of Automated Reasoning devoted to the 40 years of the Mizar project.
7. J. Robinson, A machine-oriented logic based on resolution principle. *J. of the ACM*, 12(1), pp. 23-41, 1965.
8. S. Yu. Maslov. The Inverse method for establishing the deducibility in the classical predicate calculus. *DAN SSSR*, 159(1), pp. 17–20, 1964. In Russian.
9. Mints G.: Herbrand theorem. Mathematical Theory of Logical Inference, pp. 311–350, Nauka, Moscow (1967). In Russian.
10. Ch. Lee and R. Ch. Chang. Symbolic logic and mechanical theorem proving. Academic Press, New York, 331 pp. (1973).
11. J. A. Robinson. An overview of mechanical theorem proving. Lecture Notes in Operations Research and Mathematical Systems, 28, pp. 2–20 (1970).

12. A. V. Lyaletski and A. I. Malashonok. A calculus of c-clauses based on the clash-resolution rule. In *Mathematical Issues of Intellectual Machines Theory*, pp. 3–33, GIC AS UkrSSR, Kiev, 1975. In Russian.
13. A. V. Lyaletski. On a calculus of c-clauses. In *Mathematical Issues of Intellectual Machines Theory*, pp. 34–48, GIC AS UkrSSR, Kiev, 1975. In Russian.
14. A. V. Lyaletski. On minimal inferences in the calculi of c-clauses. Issues of the Theory of Robots and Artificial Intelligence, pp. 34–48, GIC AS UkrSSR, Kiev, 1975. In Russian.
15. S. Yu. Maslov. Proof-search strategies. Machine Intelligence, No. 6, American Elsevier, New York, 1971, pp. 77–90, 1971
16. V. Lifschitz. What is the inverse method? Journal of Automated Reasoning, 5(1), pp. 1–23, 1989.
17. A. Degtyarev and A. Voronkov. Equality Elimination for the Inverse Method and Extension Procedures, Technical report 92, Uppsala University, Computing Science Department, 1994.
18. F. V. Anufriev, V. V. Fedyurko, A. A. Letichevskii, Z. M. Asel'derov, and I. I. Didukh. An algorithm for proving theorems in group theory. *Cybernetics and System Analysis*, 2(1), pp. 20–25, Springer, New York, 1966.
19. F. V. Anufriev. An algorithm for proving theorems in set theory. In *The Theory of Automata*, vol. 4, pp. 3–24. GIC AS UkrSSR, Kiev, 1967. In Russian.
20. F. V. Anufriyev. An algorithm of search for proofs of theorems in logical calculi. In *The Theory of Automata*, pp. 3–26. GIC AS UkrSSR, Kiev, 1969. In Russian.
21. A. I. Malashonok. The soundness and completeness of the Evidence Algorithm. In *Computer-Aided Theorem Proving in Mathematics*, pp. 75–95, GIC AS UkrSSR, Kiev, 1974. In Russian.
22. S. Kanger. Simplified proof method for elementary logic. *Comp. Program. and Form. Sys.*, pp. 87–93, 1963.
23. A. I. Degtyarev and A. V. Lyaletski. Logical inference in the system of automated proving, SAD. In *Mathematical Foundations of Artificial Intelligence Systems*, pp. 3–11, GIC AS UkrSSR, Kiev, 1981. In Russian.
24. A. Degtyarev, A. Lyaletski, and M. Morokhovets. Evidence algorithm and sequent logical inference search. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Logic for Programming and Automated Reasoning (LPAR'99)*, *Lecture Notes in Computer Science*, 1705, pp. 44–61. Springer, 1999.
25. A. Lyaletski. On efficient inference search in first-order cut-free modal sequent calculi. In *Pre-Proceedings of the 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 25–34, Timisoara, Romania, September 2008.
26. B. Konev and A. Lyaletski. Tableau method with free variables for intuitionistic logic. In M. Klopotek, S. Wierzchon, and K. Trojanowski, editors, *Proceedings of the International IIS:IIPWM'06 conference*, Intelligent Information Processing and Web Mining, pages 293–305, Ustron, Poland, June 2006. Springer.
27. A. Lyaletski. Sequent formalism and inference search in the Evidence Algorithm style. *Proceedings of the International conference "Modern Informatics: Problems, Achievements, and Prospects of Development"* (devoted to the 90th anniversary of academician V. M. Glushkov), pages 54-56. Kyiv, Ukraine, September 2013.
28. A. Lyaletski. Sequent forms of Herbrand theorem and their applications. *Annals of Mathematics and Artificial Intelligence*, 46(1-2), pages 191–230, 2006.
29. A. Lyaletski. Computer-oriented calculus of sequent trees. In *Foundations of Information and Knowledge Systems, 3rd International Symposium, FoIKS 2004*,

volume 2942 of *Lecture Notes in Computer Science*, pp. 213–230, Austria, February 2004.
30. A. Lyaletski, A. Letichevsky, and O. Kalinovskyy. Literal trees and resolution technique. In M. Klopotek, S. Wierzchon, and K. Trojanowski, editors, *Proceedings of the International IIS:IIPWM'05*, Intelligent Information Processing and Web Mining, pp. 97–106, Zakopane, Poland, June 2005. Springer.
31. D. W. Loveland. Mechanical theorem proving by model elimination. *Journal of the ACM*, 16(3), pp. 349–363, 1968.
32. V. M. Glushkov. Automata theory and formal transformations of microprograms. *Cybernetics*, No. 5, pp. 3–10, 1965. In Russian.
33. V. N. Redko. Compositions of programs and composition programming. *Programming*, No. 5, pp. 3–24, 1978. In Russian.
34. V. N. Redko. Semantic structures of programs. *Programming*, No. 1, pp. 3–14, 1981. In Russian.
35. B. V. Gnedenko, V. S. Korolyuk, and E. L. Yushchenko. Elements of Programming. *Fizmatgiz, Moscow*, 1961. In Russian.
36. I. A. Basarab, N. S. Nikitchenko, V. N. Redko. Composition Databases. *Lybid, Kiev*, 1992. In Russian.
37. H. R. Nielson H.R. and F. Nielson. Semantics with applications: A Formal Introduction. *John Wiley & Sons Inc.*, 1992.
38. D. Harel, D. Kozen, J. Tiuryn. Dynamic Logic. *MIT Press, Cambridge, MA*, 2000.
39. A. Kryvolap, M. Nikitchenko, W. Schreiner. Extending Floyd-Hoare logic for partial pre- and postconditions. *CCIS*, Springer, Heidelberg, vol. 412, pp. 355–378, 2013.
40. N. S. Nikitchenko. Composition-nominative approach to program semantics. *Techn. Rep. TR: 1998-020, Technical University of Denmark*, 1998.
41. M. S. Nikitchenko. Gnoseology-based approach to foundations of informatics. *Proc. of the 7-th Int. Conf. ICTERI-2011, Kherson, Ukraine, May 4-7, 2011*, CEUR-WS.org/Vol-716, pp. 27–40, 2011.
42. N. S. Nikitchenko. Abstract computability of non-deterministic programs over various data structures. *Perspectives of System Informatics, LNCS*, vol. 2244, pp. 471–484, 2001.
43. M. S. Nikitchenko. Composition-nominative aspects of address programming. *Cybernetics and System Analysis*, No. 6: 24–35, 2009. In Russian.
44. M. S. Nikitchenko and S. S. Shkilnyak. Mathematical logic and theory of algorithms. *Publishing house of Taras Shevchenko National University of Kyiv, Kyiv*, 2008. In Ukrainian.
45. M. S. Nikitchenko and V. G. Tymofieiev. Satisfiability in composition-nominative logics. *Central European Journal of Computer Science*, 2(3), pp. 194–213, 2012.
46. S. S. Shkilnyak. First-order logics of quasiary predicates. *Cybernetics and System Analysis*, No. 6, pp. 21–50, 2010. In Russian.
47. Hao Wang. Towards mechanical mathematics. *IBM Journal of Research and Development*, 4, pp. 2–22, 1960.

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Towards Standard Environments for Formalizing Mathematics

Adam Grabowski[1] and Christoph Schwarzweller[2]

[1] Institute of Informatics, University of Białystok
ul. Akademicka 2, 15-267 Białystok, Poland
adam@math.uwb.edu.pl
[2] Department of Computer Science, University of Gdańsk
ul. Wita Stwosza 57, 80-952 Gdańsk, Poland
schwarzw@inf.univ.gda.pl

Abstract. Though more and more advanced theorems have been formalized in proof systems their presentation still lacks the elegance of mathematical writing. The reason is that proof systems have to state much more details – a large number of which is usually omitted by mathematicians. In this paper we argue that proof languages should be improved into this direction to make proof systems more attractive and usable – the ultimate goal of course being a like-on-paper presentation. We show that using advanced Mizar typing techniques we already have the ability of formalizing pretty close to mathematical paper style. Consequently users of proof systems should be supplied with environments providing and automating these techniques, so that they can easily benefit from these.

1 Introduction

Interactive reasoning aims at developing methods and systems to be used to formalize – state and prove – mathematical theorems in a comfortable way. The ultimate dream is a system containing all mathematical knowledge in which mathematicians develop and prove new theories and theorems. Though more and more advanced pieces of mathematical knowledge are being formalized, we are still far from this dream – in particular few mathematicians even notice proof systems.[3] Formalizing mathematics more or less still is a matter of computer scientists.

In our opinion the main reason is the clash between how mathematicians and proof systems work: Any proof system by nature is based on logical rigour to ensure correctness of formalization. Consequently such systems state theorems more or less as logical formulae and use a logical calculus doing inferences to prove them. Mathematicians, however, do not use logic or logical symbols in the strong sense of proof systems. They argue rather intuitively assuming that their arguments can be easily transformed into such a logical reasoning. From this stem reservations against using proof systems like "Theorems are hard to read", "Too much obvious facts has to be explicitly considered", or "Applying theorems is too elaborated".

[3] The most prominent exception is Thomas Hales' Flyspeck project [7].

To illustrate the above we consider the term $n-1$ with $n \in \mathbb{N}$ as an easy example. In some situations – to apply a theorem or to use $n-1$ as an argument of a function – $n-1$ has to be a natural number. This is of course obvious if $n \geqslant 1$ (or $n \neq 0$), so mathematicians do not care about. Proof systems have to be much more formal: One has to prove that in this particular situation $n-1 \in \mathbb{N}$. This is of course pretty easy and can for example be done by changing the type of $n-1$ to $\mathbb{N}$, using the monus function $n \dot{-} 1$ or by generating some proof obligation. Somewhat more involved examples would be $(p-1)/2 \in \mathbb{N}$, if $p \neq 2$ is a prime or that $(-1)^n = -1$, if n is odd.

Though there have been efforts to overcome these shortcomings, we claim that in proof systems this kind of mathematical obviousness should be more strengthened: Proofs as those in the above example must be invisible for users, that is automatically identified and conducted. In this paper we show that Mizar's attributed types [14], [2] can be used to do so: Providing a number of so-called registrations and redefinitions – stored and made available to users in a special environment – automates reasoning as sketched in the above example and therefore allows for a much more mathematicians-like handling of mathematical knowledge. More concrete, we present examples from number theory – which in particular includes theorems as mentioned above – and deal with instantiation of algebraic structures.

2 Pocklington's Theorem

Pocklington's criterium is a number theoretical result providing a sufficient condition for (large) numbers to be prime. It may be worth mentioning that in the original work [15] there is no precisely stated theorem. In the literature one therefore finds a number of different variants. One version (from [3]) reads as follows.

> Let s be a positive divisor of $n-1$, $s > \sqrt{n}$. Suppose there is an integer a satisfying:
>
> $$a^{n-1} \equiv 1 \pmod{n}$$
> $$\gcd(a^{(n-1)/q} - 1, n) = 1$$
>
> for each prime q dividing s. Then n is prime.

One can find several formalizations of Pocklington's criterium none of which, however, resembles completely the mathematical formulation. In Mizar [16] we find for example the following.

```
for n,f,d,n1,a,q being Element of NAT
st n-1 = q|^n1 * d & q|^n1 > d & d > 0 & q is prime &
   a|^(n-'1) mod n = 1 & (a|^((n-'1) div q)-'1) gcd n = 1
holds n is prime;
```

As we see the minus function `-'` and division with remainder though not being part of the theorem are used. Furthermore besides `mod` function and `prime` the formalization does not use no number theoretical notation, even divisibility is expressed implicitly.

The formalization found in [4] adds `coprime` as a number theoretical notation in this way substituting the `gcd` function. Note, however, that divisibility is expressed once explicitly using predicate | and once implicitly.

$$n \geqslant 2 \wedge n - 1 = q \cdot r \wedge n \leqslant q^2 \wedge a^{n-1} \equiv 1 \ (mod\ n)$$
$$\wedge(\forall p.\texttt{prime}\ p \wedge p|q \longrightarrow \texttt{coprime}(a^{\frac{n-1}{q}} - 1)\ n) \longrightarrow \texttt{prime}\ n.$$

In Coq [6] a somewhat different version has been formalized using a partial factorization of $n - 1$. Therefore lists of natural numbers have been used. Congruence of numbers is here expressed using the `Mod`-function, and `S` and `pred` denote $n + 1$ and $n \dot{-} 1$, respectively.

```
∀(n q m : nat) (a : Z) (qlist : natlist),
n > 1 →
n = S (q × m) →
q = product qlist →
allPrime qlist →
Mod (Exp a (pred n)) 1 n →
allLinCombMod a n m qlist → n ⩽ q × q → Prime n.
```

Though of course correct – and also more or less well-readable – all these formalizations rather present themselves as an expression having been proved in a formal system than as a well-formulated mathematical theorem. In the rest of this section we will show how preparing a number theoretic environment allows for the following Mizar formulation of Pocklington's theorem.

```
for n being 2_greater natural number,
    s being non trivial Divisor of n-1 st s > sqrt(n) &
  ex a being natural number
  st a|^(n-1),1 are_congruent_mod n &
     for q being PrimeDivisor of s holds a|^((n-1)/q) - 1 gcd n = 1
holds n is prime;
```

2.1 Preparing Proper Mathematical Notation

The first step is obvious. We have to introduce definitions resembling the mathematical objects of concern: `Divisor`, `PrimeDivisor`, `are_congruent_mod`, and so on. Some of them were already available in the Mizar Mathematical Library, some we had to introduce by ourselves. How to do this has been described for example in [8] and is therefore omitted. However, we also introduced the $<$-relation as a Mizar adjective `_greater` by the following attribute definition.

```
definition
  let n,x be natural number;
  attr x is n_greater means x > n;
end;
```

This at first sight seems to be an unnecessary repetition. However, Mizar adjectives can be used in so-called cluster registrations to automatically extend and enrich objects' types. For example, the fact that primes $p \neq 2$ are odd can now be formulated – and proved – not only as a theorem, but also as a cluster registration:

```
registration
  cluster 2_greater -> odd for PrimeNumber;
end;
```

As a consequence having `p` of type `2_greater PrimeNumber` Mizar automatically adds the adjective `odd` to the type of `p` in this way adding hidden information about mathematical objects – that mathematicians use implicitly. In section 2.2 from this then will – also automatically – follow that `(p-1)/2` for such `p` is actually a natural number. To give another example here, the existence of an arbitrary large prime number can be guaranteed by the following registration.

```
registration
  let n be natural number;
  cluster n_greater for PrimeNumber;
end;
```

Now, if necessary, the user can declare an arbitrary large prime number by just writing `let p be 12345_greater PrimeNumber;` or even more generally by `let n be natural number; let p be n_greater PrimeNumber;`. Its existence is guaranteed by the above registration and the fact that `p` is greater than `12345` or `n` respectively can be used in the following without any proving or referencing.

2.2 Automatically Adapting Types

The cluster mechanism of adding adjectives to an object's type from the last subsection can be used to automatically adapt types in particular situations. In this way users – like mathematicians – do not have to deal explicitly with changing and adapting types to apply functors or theorems.

To deal with the easy example from the introduction first, if $n \in \mathbb{N}$ is not equal to 0 the type of $n-1$ of course can be changed to natural number. To do this automatically, we identify properties – given by adjectives – ensuring that $n - 1 \in \mathbb{N}$ and formulate corresponding registrations, such as for example

```
registration
  let n be non zero natural number;
  cluster n-1 -> natural;
end;

registration
  let m be natural number;
  cluster m_greater -> non zero for natural number;
end;
```

Note here that registrations do not stand alone, but are applied in a iterative matter.[4] As a consequence the type of $n - 1$ now happens to be `natural number` not only if n is `non zero`, but also if n is `m_greater` for an arbitrary natural number m.

We end this section by illustrating how the use of adjectives and cluster registrations allows to avoid additional helper functions such as minus and division with remainder to formulate Pocklington's theorem. Having the following registration

[4] Actually Mizar rounds up an object's type by adding all adjectives from clusters available in the environment, see [2].

```
registration
  let n be odd natural number;
  cluster (n-1)/2 -> natural;
end;
```

then, if `p` is of type `2_greater PrimeNumber` the type of `(p-1)/2` is not just `real number` as given by the type of the division functor `/`. Together with the registrations from section 2.1 both adjectives `odd` and then `natural` are added to the type of `(p-1)/2`. Hence its type in particular is `natural number` and `(p-1)/2` is therefore accepted as the argument of a function requiring natural numbers. Note that once the registration has been introduced, no proof obligation for the user shows up, all that's necessary has – and must have – been proved in the cluster registration. Using the earlier introduced type `Divisor` the following

```
registration
  let n be natural number;
  let q be Divisor of n;
  cluster n/q -> natural;
end;
```

now is an easy generalization of the former case – `q = 2` – changing the type of a quotient to natural number, as necessary in the formulation of Pocklington's theorem. Note again that the type of `n/q` is automatically enriched with adjective `natural`, if `n` and `q` have the attributed types mentioned in the registration.

3 Abstract Mathematical Structures and Instantiations

Another main topic is moving between mathematical structures: Mathematical proofs receive their elegance from noting that a given domain constitutes a special structure and applying theorems from it. Here both jumping to completely different structures as well as inheriting from more general structures is of concern. In proof systems, however, this goes along with a type coercion. The type of an element of a ring is different from the one of a real number, of an element of a group or a topological space. Much effort has been spent to ease users of proof systems to move between and to apply theorems from different structures, see e.g. [9], [1], [17], [18].

Here we deal with another topic connected with inheriting from general structures: Functions and properties defined in a general structure are to be refined or extended in a more concrete one. As a running example we consider greatest common divisors in different domains. The greatest common divisor and a number of its basic properties can be defined for arbitrary gcd domains. Note, however, that one cannot define a gcd function, just because in general the gcd is not unique. In gcd domains we therefore end up with a type `a_gcd`:[5]

```
definition
  let L be non empty multMagma;
  let x,y,z be Element of L;
  attr z is x,y-gcd means
```

[5] One can of course also define the set of gcds for given x and y, but we found it more convenient to use Mizar types here.

```
    z divides x & z divides y &
    for r being Element of L
                st r divides x & r divides y holds r divides z;
end;

definition
  let L be gcdDomain;
  let x,y be Element of L;
  mode a_gcd of x,y is x,y-gcd Element of L;
end;
```

In more concrete gcd domains – so-called instantiations – such as the ring of integers or polynomial rings the notion of a gcd now is adopted – actually changed into a gcd function – by just saying that the gcd is greater than 0 or the gcd is monic, respectively. However, these additional properties apply only to objects of the more concrete type – `Integer` and `Polynomial` – whereas `a_gcd` expects arguments of the more general type `Element of L`, where `L` is a `gcdDomain`. To easily adopt mathematical refining techniques we need a way to – automatically – identify these types.

3.1 Preparing the Instantiation

Instantiations of abstract structures are defined by gluing together the corresponding objects and operations in the appropriate structure, in our example `doubleLoopStr`. So the ring of polynomials with coefficients from a structure `L` can be defined by

```
definition
  let L be Ring;
  func Polynom-Ring L -> strict non empty doubleLoopStr equals
    doubleLoopStr(#POLYS,polyadd(L),polymult(L),1_.(L),0_.(L)#);
end;
```

where `POLYS` is the set of objects with type `Polynomial of L`. So we are left with two different types `Polynomial of L` and `Element of the carrier of Polynom-Ring L`, the latter one being the type of the – abstract – ring elements. As a consequence special properties of polynomials can be only defined for the concrete type `Polynomial of L`, such as for example an adjective `monic` . Even after its definition, `monic` is not available for objects of type `Element of the carrier of Polynom-Ring L`:

```
definition
  let L be Ring,
      p be Polynomial of L;
  attr p is monic means Leading-Coefficient p = 1.L:
end;

now let L be Ring;
  let p be Element of the carrier of Polynom-Ring L;
  p is monic;
::>    *106: Unknown attribute
    ...
end;
```

This is unsatisfying not only because it is obvious that `p` in this example is actually a polynomial, but also because it prevents the combination of `monic` with the former defined type `a_gcd`. The solution is to automatically cast abstract types into concrete ones, here `Element of the carrier of Polynom-Ring L` into `Polynomial of L`. Again attributes – and a so-called redefinition – allow both to describe such situations and to enhance Mizar type checking: First an attribute `polynomial-membered` describes sets containing only elements of type `Polynomials of L`. Then for such sets the type of its elements can be redefined into `Polynmial of L` – because the attribute `polynomial-membered` ensures that this cast is possible.[6]

```
definition
  let L be non empty ZeroStr;
  let X be set;
  attr X is L-polynomial-membered means
    for p be set st p in X holds p is Polynomial of L;
end;

definition
  let L be Ring;
  let X be non empty L-polynomial-membered set;
  redefine mode Element of X -> Polynomial of L;
end;
```

All that remains now is stating – and proving – a cluster saying that the type `the carrier of Polynom-Ring L` can automatically be enriched with the adjective `polynomial-membered`

```
registration
  let L be Ring;
  cluster the carrier of Polynom-Ring L -> L-polynomial-membered;
end;
```

and objects of type `Element of the carrier of Polynom-Ring L` are automatically identified as objects also having the concrete type `Polynomial of L`. Therefore notions defined for `Polynomial of L` – such as for example `monic` – are also available for objects of type `Element of the carrier of Polynom-Ring L`.

3.2 Extending Definitions in the Instantiation

Working in an environment containing the clusters and redefinitions from the last section users can now extend and combine properties defined for different types – abstract ring elements and concrete polynomials – according to their needs. First – if not already provided in the standard environment – one has to ensure that the instantiation establishes the abstract structure, here that `Polynom-Ring L` is a gcd domain.

```
registration
  let L be Field;
  cluster Polynom-Ring L -> Euclidian;
end;
```

[6] In fact exactly this has to be proved in the redefinition.

Then all is set: Both abstract notions from `gcdDomain` and concrete ones for `Polynomials` are available and can be easily used. The definition for polynomial gcd function, for example, is now just combining notions `a_gcd` and `monic`.

```
definition
  let L be Field;
  let p,q be Element of the carrier of Polynom-Ring L;
  func p gcd q -> Element of the carrier of Polynom-Ring L means
    it is a_gcd of p,q & it is monic;
end;
```

Please note again, that notion `monic` has been introduced for objects of type `Polynomial of L`, whereas notion `a_gcd` for objects of type `Element of the carrier of Polynom-Ring L`. To nicely complete the development of polynomial gcds one should in addition provide the following registration enriching the type of the just defined gcd function – to make available properties of polynomial gcds without the need of referencing its definition.[7]

```
registration
  let L be Field;
  let p,q be Element of the carrier of Polynom-Ring L;
  cluster p gcd q -> monic p,q-gcd;
end;
```

4 Conclusion and Further Development

We have seen how thorough preparation of Mizar environments using registrations and redefinitions to manipulate mathematical objects' types not only improves working in a special mathematical theory, but also enables automatic use of hidden information – information that is used implicitly but is not stated by mathematicians. It is this kind of obvious knowledge and inferences that proof systems must enable in order to attract more users. We claim that further development of the presented techniques will lead to both more convenience in formalizing mathematics and more recognition of proof systems by mathematicians.

In this context we want to discuss briefly two further items that seem important to us. Firstly, a lot of theorems mathematicians do not mention in proofs can actually be presented as term reductions, just because they describe equalities, such as for example $(-1)^n = -1$, if n is odd, $v - v = 0$, $a \wedge b = a$, if $a \leqslant b$ or $x \cup \emptyset = x$. Of course it depends on the proof assistant to which extent such equalities/reductions are automatically applied. Mizar, however, provides a language construct similar to clustering that enables users enriching the proof process by particular reductions [13], so for example

```
registration
  let n be odd natural number;
  reduce (-1)|^n to -1;
end;
```

[7] There is a relatively new Mizar environment directive – `expansion` – that also serves for automatically applying definitions.

After such a registration the Mizar prover automatically identifies the left term with the term on the right side, so that – even depending on the object's type – equalities are automatically performed and accepted.

```
now let n be odd natural number;
  5 * (-1)|^n = -5;
  ...
end;
```

Unfortunately not all examples from above can be registered this way, because at the moment reductions must be to proper subterms.

The second point concerns the use of ellipses, a constituent mathematicians use to eliminate logical formulae describing finite sequences. In [5] we find, for example, Pocklington's theorem as follows:

> Let $n \in \mathbb{N}$, $n > 1$ with $n - 1 = q \cdot m$ such that $q = q_1 \cdots q_t$ for certain primes $q_1, \ldots q_t$. Suppose that $a \in \mathbb{Z}$ satisfies $a^{n-1} = 1 \pmod{n}$ and $\gcd(a^{\frac{n-1}{q_i}} - 1, n) = 1$ for all $i = 1, \ldots t$. If $q > \sqrt{n}$, then n is a prime.

The Coq version of Pocklington's theorem mentioned in section 2 actually formalizes this theorem and therefore uses lists of natural numbers. Mizar already offers the use of ellipses, but only if the underlying formula is existential [12], like for example in the following theorem.

```
  for n being non zero natural number,
      x being integer number holds
  x,0 are_congruent_mod n or ... or x,(n-1) are_congruent_mod n;
```

This, however, unfortunately does not allow to formulate Pocklington's theorem with the use of ellipses. In particular the use of ellipses for indexed variables is necessary here.

Summarizing, proof assistants should be capable of automatically identifying and performing obvious mathematical arguments and shortcuts, that is arguments left out by working mathematicians. In our opinion the way to do so is not strengthening the proof assistant's inference system, but making its proof languages more flexible and adaptable by using language constructs like those presented in the paper: They allow to explicitly state theorems behind mathematicians' obvious arguments and then enrich the proof process by automatically applying them. In this way users – or developers by providing standard environments for formalizing mathematics – can adapt the proof process according to their particular needs.

References

1. C. Ballarin, Locales: A Module System for Mathematical Theories; Journal of Automated Reasoning, 52(2), pp. 123–153, 2014.
2. G. Bancerek, On the Structure of Mizar Types; Electronic Notes in Theoretical Computer Science, 85(7), pp. 69–85, 2003.
3. J. Buchmann, V. Müller, Primality Testing; available at `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.1952`

4. A. Chaieb, Automated Methods for Formal Proofs in Simple Arithmetics and Algebra; Dissertation, TU München, Germany, 2008.
5. O. Caprotti, M. Oostdijk, Formal and Efficient Primality Proofs by Use of Computer Algebra Oracles; Journal of Symbolic Computation, 32(1/2), pp. 55–70, 2001.
6. The Coq Proof Assistant; `http://coq.inria.fr`.
7. The Flyspeck Project; `http://code.google.com/p/flyspeck`.
8. A. Grabowski, A. Korniłowicz, A. Naumowicz, Mizar in a Nutshell; Journal of Formalized Reasoning, 3(2), pp. 153–245, 2010.
9. H. Geuvers, R. Pollack, F. Wiedijk, J. Zwanenburg, A Constructive Algebraic Hierachy in Coq; Journal of Symbolic Computation, 34(4), pp. 271–286, 2002.
10. A. Grabowski, C. Schwarzweller, On Duplication in Mathematical Repositories; in S. Autexier et al. (eds.), Intelligent Computer Mathematics, LNCS 6167, pp. 427–439, 2010.
11. A. Korniłowicz, How to define Terms in Mizar Effectively?; Studies in Logic, Grammar and Rhetoric, 18(31), pp. 67–77, 2009.
12. A. Korniłowicz, Tentative Experiments with Ellipsis in Mizar; in J. Jeuring et al. (eds.), Intelligent Computer Mathematics, LNCS 7362, pp. 453–457, 2012.
13. A. Korniłowicz, On Rewriting Rules in Mizar; Journal of Automated Reasoning, 50(2), pp. 203–210, 2013.
14. The Mizar Home Page; `http://mizar.org`.
15. H.C. Pocklington, The Determination of the Prime or Composite Nature of Large Numbers by Fermat's Theorem; Proc. Camb. Phil. Soc., vol. 18, pp. 29–30, 1914.
16. M. Riccardi, Pocklington's Theorem and Bertrand's Postulate; Formalized Mathematics, 14(2), pp. 47–52, 2006.
17. P. Rudnicki, A. Trybulec, C. Schwarzweller, Commutative Algebra in the Mizar System; Journal of Symbolic Computation, 32(1/2), pp. 143–169, 2001.
18. C. Schwarzweller, Mizar Attributes: A Technique to Encode Mathematical Knowledge into Type Systems; Studies in Logic, Grammar and Rhetoric, 10(23), pp. 387–400, 2007.

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Parallelizing Mizar

Josef Urban*

Radboud University, Nijmegen

Abstract. This paper surveys and describes the implementation of parallelization of the Mizar proof checking and of related Mizar utilities. The implementation makes use of Mizar's compiler-like division into several relatively independent passes, with typically quite different processing speeds. The information produced in earlier (typically much faster) passes can be used to parallelize the later (typically much slower) passes. The parallelization now works by splitting the formalization into a suitable number of pieces that are processed in parallel, assembling from them together the required results. The implementation is evaluated on examples from the Mizar library, and future extensions are discussed.

1 Introduction and Motivation

While in the 90-ies the processing speed of a single CPU has grown quickly, in the last decade this growth has considerably slowed down, or even stopped. The main advances in processing power of computers have been recently done by packing multiple cores into a single CPU, and related technologies like hyperthreading. A low-range dual-CPU (Intel Xeon 2.27 GHz) MathWiki server of the Foundations Group at the Radboud University bought in 2010 has eight hyperthreading cores, so the highest raw performance is obtained by running sixteen processes in parallel. The server of the Mizar group at University of Bialystok has similar characteristics, and the Mizar server at University of Alberta has twelve hyperthreading cores. Packing of CPU cores together is happenning not only on servers, but increasingly also on desktops and notebooks, making the advantages of parallelization attractive to many applications.

To take advantage of this development, reasonable ways of parallelizing time-consuming computer tasks have to be introduced. This paper discusses the various ways of parallelization of proof checking with the Mizar formal proof verifier, and parallelization of the related Mizar utilities. Several parallelization methods suitable for different scenarios and use-cases are introduced, implemented, and evaluated.

The paper is organized as follows: Section 2 describes the main tasks done today by the Mizar [7, 17] verifier and related utilities, and the ways how they are performed. Section 3 explores the various possible ways and granularity levels in which suitable parallelization of the Mizar processing could be done, and their advantages and disadvantages for various use scenarious. Section 4 describes and evaluates parallelization of the processing of the whole Mizar library and Mizar wiki done on the coarsest level of granularity, i.e. on the article level. Section 5 then describes the recent parallelization done on sub-article levels of granularity, i.e. useful for the speedup of processing of a single Mizar article. Both the verification and various other utilities have been parallelized this way, and evaluation on hundreds of Mizar articles is done. Section 7 names possible future directions, and concludes.

* Supported by the NWO project MathWiki.

2 Mizar Processing

2.1 Article Workflow

The term *Mizar Processing* can in the broad sense refer to several things. Mizar consists of a large library of formal matheamatical articles, on top of which new articles are written, formally verified by the Mizar verifier, possibly also checked by various (proof improving) utilities during or after the writing, possibly HTML-ized for better understanding during and after the writing, and usually translated to TeX after they are written. During the verification a number of tools can be used, ranging from tools for library searching, tools for creating proof skeletons, to tools for ATP or AI based proof advice.

After a new article is written, it is typically submitted to the library, possibly causing some refactoring of the library and itself, and the whole new version of the library is re-verified (sometimes many times during the refactoring process), and again possibly some more utilities can be then applied (again typically requiring further re-verification) before the library reaches the final state. The new library is then HTML-ized and publicly released. The library also lives in the experimental Mizar wiki based on the git distributed version control system [28, 2]. There, collaborative re-factoring of the whole library is the main goal, requiring fast real-time re-verification and HTML linking.

2.2 Basic Mizar Verification

In more detail, the basic verification of an article starts by selecting the necessary items from the library (so called *accommodation*) and creating an article-specific local environment (set of files) in which the article is then verified without further need to access the large library. The verification and other Mizar utilities then proceeds in several compiler-like passes that typically vary quite a lot in their processing times. The first *Parser* pass tokenizes the article and does a fast syntactic analysis of the symbols and a rough recognition of the main structures (proof blocks, formulas, etc.).

The second *Analyzer* pass then does the complete type computation and disambiguation of the overloading for terms and formulas, and checks the structural correctness of the natural deduction steps, and computes new goals after each such step. These processes typically take much longer than the parsing stage, especially when a relatively large portion of the library is used by the article, containing a large amount of type automations and overloaded constructs. The main product of this pass is a detailed XML file containing the disambiguated form of the article with a number of added semantic information [23]. This file serves as the main input for the final *Checker* pass, and also for the number of other Mizar proof improving utilities (e.g., the *Relprem*[1] utility mentioned in Table 1), for the HTML-ization, and also for the various ATP and AI based proof advice tools.

The final *Checker* pass takes as its main input the XML file with the fully disambiguated constructs, and uses them to run the limited Mizar refutational theorem prover for each of the (typically many) atomic (*by*) justification steps. Even though this checker is continuosly optimised to provide a reasonable combination of strength, speed, and "human obviousness", this is typically the slowest of the verifier passes. Similar situation is with the various utilities for improving (already correct) Mizar

[1] Irrelevant Premises Detector

proofs. Such utilities also typically start with the disambiguated XML file as an input, and typically try to merge some of the atomic proof steps or remove some redundant assumptions from them. This may involve running the limited Mizar theorem prover several times for each of the atomic proof steps, making such utilities even slower than the *Checker* pass.

2.3 Other Tools

All the processes described so far are implemented using the Mizar code base written in object-oriented extension of Pascal. The disambiguated XML file is also used as an input for creation of the HTML representation of the article, done purely by XSL processing. XSL processing is also used for translation of the article to an ATP format, serving as an input for preparing ATP problems (solvable by ATP systems) corresponding to the problems in the Mizar article, and also for preparing data for other proof advice systems (MML Query, Mizar Proof Advisor). The XSL processing is usually done in two stages. The first stage (called *absolutization*) is common for all these utilities, it basically translates the disambiguated constructs living in the local article's environment into the global world of the whole Mizar library. The second stage is then the actual XSL translation done for a particular application. The XSL processing can take very different times depending on its complexity. Generally, XSL processors are not as much speed-optimized as, e.g., the Pascal compilers, so complex XSL processing can take more time than analogous processing programmed in Pascal.

Finally, there are a number of proof advice tools, typically taking as input the suitably translated XML file, and providing all kinds of proof advice using external processing. Let us mention at least the Automated Reasoning for Mizar [32, 30, 9] system, linking Mizar through its Emacs authoring environment and through a HTML interface to ATP systems (particulary a custom version [29] of the Vampire-SInE system [15] and a customized [27] version of E [19]) usable for finding and completing proofs automatically, for explaining the Mizar atomic justifications, and for ATP-based cross-verification of Mizar. This processing adds (at least) two more stages: (i) It uses the MPTP system [22, 25] to produce the ATP problems corresponding to the Mizar formulation, and (ii) it uses various ATP/AI systems and metasystems to solve such problems. Attached to such functions is typically various pre/post-processing done in Emacs Lisp and/or as CGI functions.

See Figure 1 for the overall structure of Mizar and related processing for one article. Table 1 gives timings of the various parts of Mizar processing for the more involved Mizar article `fdiff_1` about real function differentiability[2] [14], and for the less involved Mizar article `abian` about Abian's fixed point theorem[3] [16] run on recent Intel Atom 1.66 GHz notebook[4].

[2] `http://mws.cs.ru.nl/~mptp/mml/mml/fdiff_1.miz`

[3] `http://mws.cs.ru.nl/~mptp/mml/mml/abian.miz`

[4] This small measurement is intentionally done on a standard low-end notebook, while the rest of global measurements in this paper are done on the above mentioned server of the Foundations Group. This is in order to compare the effect of parallelized server-based verification with standard notebook work in Section 5.

Fig. 1. Structure of the Mizar processing for one article

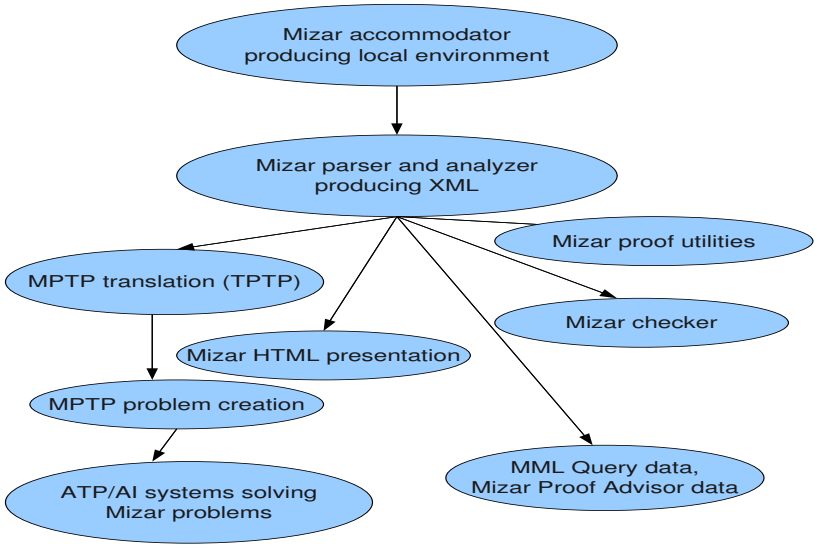


Table 1. Speed of various parts of the Mizar processing on articles fdiff_1 and abian in seconds - real time and user time

Processing (language)	real - fdiff_1	user - fdiff_1	real - abian	user - abian
Accommodation (Pascal)	1.800	1.597	1.291	1.100
Parser (Pascal)	0.396	0.337	0.244	0.183
Analyzer (Pascal)	28.455	26.155	4.182	4.076
Checker (Pascal)	39.213	36.631	10.628	10.543
Relprem (Pascal)	101.947	99.385	48.493	47.683
Absolutizer (XSL)	17.203	13.579	9.624	7.886
HTML-izer (XSL)	27.699	24.498	11.582	11.323
MPTP-izer (XSL)	70.153	68.919	47.271	45.410

3 Survey of Mizar Parallelization Possibilities

There are several ways how to parallelize Mizar and related utilities, and several possible levels of granularity. Note that for any of these Mizar parallelization methods the main issue is speed, not the memory consumption. This is because Pascal does not have garbage collection, and Mizar is very memory efficient, taking typically less than 30MB RAM for verifying an article. The reason for this extreme care is mainly historical, i.e., the codebase goes back to times when memory was very expensive. Methods used for this range from exhaustive sharing of data structures, to using only the part of the library that is really necessary (see *accommodation* in 2.2).

The simplest method of parallelization which is useful for the Mizar wiki users, developers, and library maintainers is article-level parallelization of the whole library verification, and parallization of various other utilities applied to the whole Mizar library. There are about 1100 Mizar articles in the recent library, and with this number the parallelization on the article level is already very useful and can bring a lot of speed-ups, especially useful in the real-time wiki setting, and for the more time consuming utilities like the above mentioned *Relprem*.

A typical user is however mainly interested in working with one (his own) article. For that, finer (sub-article) levels of parallelization are needed. A closer look at the Table 1 indicates that the *Parser* pass of the verification is very fast, while the *Analyzer* and especially the *Checker* passes are the bottlenecks (see also the global statistics for the whole MML processing done with article-level parallelization in Table 2).

3.1 Checker parallelization

There are several basic options to parallelizing the most costly verification operation - the *Checker* pass, they are explained in more detail below:

1. Running several *Checker* passes in parallel as separate executables, each checking only a part of the atomic steps conducted in the article
2. Running one *Checker* pass as only one executable, with multithreading code used for parallelizing the main checking procedure
3. Running one *Checker* pass as only one executable, with multithreading code used inside the main checking procedure
4. Combinations of above

As mentioned above, the input for the *Checker* pass is a fully disambiguated article, where only the atomic justification steps need to be checked, i.e. proved by the Mizar's limited theorem prover. The number of such atomic justification steps in one article is typically high, about every second to third line in a formal Mizar text is justified in such a way. The result of one such theorem proving attempt is completely independent of others, and it is just a boolean value (true or false)[5]. All of these theorem proving attempts however share a lot of data-structures that are basically read-only for them, for example information about the types of all the ground terms appearing up to the particular point in the formal text, and information about the equalities holding about ground terms at particular points of the formal text.

The first method suggested above - running several *Checker* passes in parallel as separate executables, each checking only a part of the atomic steps conducted in the article - is relatively "low-tech", however it has some good properties. First, in the

[5] Note that this is not generally true for nonclassical systems like Coq, where the proof might not be an opaque object.

methods based on multithreading, the relatively large amount of the shared data has to be cloned in memory each time a new thread is created for a new justification step. This is not the case when several executables are running from the beginning to the end, each with its own memory space. Second, the implementation can be relatively simple, and does not require use of any multithreading libraries, and related refactoring of the existing single-threaded code.

The second and third method require the use of a multithreading library (this is possible for the Free Pascal Compiler used for Mizar, with the *MTProcs* unit), and related code refactoring. There are several places where the multithreading can be introduced relatively easily, let us name at least the most obvious two: (i) the main entry to the refutational proof checker, and (ii) within the refutational proof checker, separately disproving each of the disjuncts in the toplevel disjunctive normal form created in the initial normalization phase. The advantage of such implementation in comparison with running several executables would probably be more balanced load, and in the latter case, possibly being able to use more extreme parallelization possibilities (e.g., if 1000 cores are available, but the article has only 500 atomic justifications).

3.2 Type Analysis and Caching: Why not use fine multithreading

Caching vs. Multithreading For the also relatively costly *Analyzer* pass, the methods based on fine multithreading however seem to be either relatively complicated or of relatively little value. The problem is following: A major and increasing amount of work done in *Analyzer* consists in computing the full types of terms. This is because the Mizar mechanisms for working with adjectives are being used more and more, and are being made stronger and stronger, recently to a level that could be compared to having arbitrary Prolog programs working over a finite domain (a finite set of ground terms). The method that then very considerably improves the *Analyzer* efficiency in the singlethreaded case is simple caching of terms' types. With a simple multithreaded implementation, when the newly computed types are forgotten once the thread computing them exits, this large caching advantage is practically lost. Implementation where each thread updates the commonly used cache of terms' types is probably possible, but significantly more involved, because the access to the shared datastructures is then not just read-only (like in the *Checker* case), and the updates are likely to be very frequent.

Suitable Parallelization for Tree-like Documents Above is the reason why in the *Analyzer* case, it makes much more sense to rather have several "long-term-running" threads or processes, each developing and remembering its own cache of terms' types. The main problem is then to determine a proper level of granularity for dividing *Analyzer*'s work into such larger parts. Unlike in the *Checker* pass, *Analyzer* is not a large set of independent theorem proving runs returning just a boolean result. Analysing each term depends on the analysis of its subterms, and similarly, analysing the natural deduction structure of the proofs (another main task of this pass) depends on the results of the analysis of the proof's components (formulas, and natural deduction steps and subproofs). Thus, the finer the blocks used for parallelization, the larger the part that needs to be repeated by several threads (all of them having to analyse all the necessary parts of the nested proof, formula, and term levels leading to the fine parallelized part). To put this more visually, the formal text (proof, theory) is basically a tree (or forest) of various dependencies. The closer to the leaves the parallelization happens, the

more common work has to be repeated by multiple threads or processes when descending down the branches to the parallelization points on those branches. Obviously, the best solution is then to parallelize not on the finest possible level, but on the coarsest possible level, i.e., as soon as there are enough branches for the parallelization.

Toplevel Proofs as Suitable Parallelization Entry Points To this requirement reasonably corresponds the choice of toplevel proofs in a given formal text as the entry points for parallelization. There are typically tens to hundreds of toplevel proofs in one article, and with some exceptions (very short articles, or articles consisting of one very involved proof) these toplevel proofs can usually be divided into the necessary number of groups with roughly the same overall length. Mizar (unlike e.g. Coq) never needs the proofs for anything, only the proved theorem can be used in later proofs. Thanks to this, a simple directive (`@proof`) was introduced in the Mizar language long time ago, in order to omit verification of the (possibly long) proofs that have already been proved, and would only slow-down the verification of the current proof. This directive basically tells to the *Parser* to skip all text until the end of the proof is found, only asserting the particular proposition proved by this skipped proof. Due to the file-based communication between the passes, the whole skipped proof therefore never appears in the *Analyzer*'s input, and consequently is never analyzed. This feature can be used for file-based parallelization of the *Analyzer*, described in more detail in Section 5. It also parallelizes the *Checker*, and also can be used for easy parallelization of the subsequent HTML-ization.

3.3 HTML-ization parallelization

As mentioned above, HTML-ization of Mizar texts is based on the disambiguated article described in the XML file produced by the *Analyzer*. HTML-ization is done completely separately from the Mizar codebase written in Pascal, by XSL processing. Even though XSL is a pure lazily evaluated functional language[6], as of January 2011, the author is not aware of a XSL processor implementing multithreading. The remaining choice is then again file-based parallelization, which actually corresponds nicely to the file-based parallelization usable for skipping whole proof blocks in the *Analyzer*. During the XSL processing, it is easy to put the HTML-ized toplevel proofs each into a separate file[7], and then either to load the proofs into a browser on-demand by AJAX calls, or to merge the separate files with HTML-ized proofs created by the parallelization by a simple postprocessing into one big HTML file.

3.4 Parallelization of Related Mizar Processing

Remaining Mizar refactoring utilities (like *Relprem*) are typically implemented by modifying or extending the *Checker* or *Analyzer* passes, and thus the above discussion and solutions apply to them too. Creation of data for MML Query, Mizar Proof Advisor, and similar systems is done purely by XSL, and the file-based approach can again

[6] Thanks to being implemented in all major browsers, XSL is today probably by far the most widely used and spread purely functional language.

[7] This functionality actually already exists independently for some time, in order to decrease the size of the HTML code loaded into browser, loading the toplevel proofs from the separate files by AJAX calls.

be applied analogously to HTML-ization. The same holds for translating the article to the MPTP format (extended TPTP), again done completely in XSL. A relatively important part used for the automated reasoning functions available for Mizar is the generation of ATP problems corresponding to the Mizar problems. This is done by the MPTP system implemented in Prolog. The problem generating code is probably quite easily parallelizable in multithreaded Prologs (Prolog is by design one of the most simply parallelizable languages), however the easiest way is again just to run several instances of MPTP in parallel, each instructed to create just a part of all the article's ATP problems. The recent Emacs authoring interface for Mizar implements the functions for communicating with ATP servers asynchronously [18], thus allowing to solve as many ATP-translated problems in parallel as the user wants (and the possible remote MPTP/ATP server allows). The asynchronously provided ATP solutions then (in parallel with other editing operations) update the authored article using Emacs Lisp callbacks.[8]

As for the parallelization of the ATP solving of Mizar problems, this is a field where a lot of previous research exists [20, 21], and in some systems (e.g. Waldmeister, recent versions of Vampire used for the Mizar ATP service) this functionality is readily available. Other options include running several instances of the ATPs with different strategies, different numbers of most relevant axioms, etc. The MaLARea [26, 33, 10] metasystem for solving problems in large Mizar-like theories explores this number of choices in a controlled way, and it already has some parallelization options implemented.

4 Parallelization of the MML Processing on the Article Level

A strong motivation for fast processing of large parts of the library comes with the need for collaborative refactoring. As the library grows, it seems that the number of submissions make it more and more difficult for the small core team of the library maintainers and developers to keep the library compact, and well organized and integrated together. The solution that seems to work for Wikipedia is to outsource the process of library maintanance and refactoring to a large number of interested (or addicted) users, through a web interface to the whole library. In order for this to work in the formal case, it is however important to be able to quickly re-verify the parts of the library dependent on the refactored articles, and notify the users about the results, possibly re-generating the HTML presentation, etc.

The implementation of article-level parallelization is as follows. Instead of the old way of using shell (or equivalent MS Windows tools) for processing the whole library one article after another, a Makefile has been written, using the files produced by the various verification passes and other tools as targets, possibly introducing artificial (typically empty file) targets when there is no clear target of a certain utility. The easiest option once the various dependencies have been reasonably stated in the Makefile, is just to use the internal parallelization implemented in the GNU *make* utility. This parallelization is capable of using a pre-specified number of processes (via the `-j` option), and to analyse the Makefile dependencies so that the parallelization is only done when the dependencies allow that. The Makefile now contains dependencies for all the main processing parts mentioned above, and is regularly used by the author to process

[8] See, e.g.,the AMS 2011 system demonstration at `http://mws.cs.ru.nl/~urban/ams11/out4.ogv`

the whole MML and generate HTML and data for various other tools and utilities. In Table 2 the benefits of running `make -j64` on the recently acquired eight-core hyperthreading Intel Xeon 2.27 GHz server are summarized. The whole library verification and HTML-ization process that with the sequential processing can take half a day (or much more on older hardware), can be done in less than an hour when using this parallelization. See [28] for further details and challenges related to using this technique in the git-based formal Mizar wiki backend to provide reasonably fast-yet-verified library refactoring.

Table 2. Speed of various parts of the Mizar processing on the MML (1080 articles) with 64 process parallelization run on an 8-core hyperthreading machine, in seconds - real time and user time, total and averages for the whole MML.

Stage (language)	real times total	user times total	real times avrg	user times avrg
Parser (Pascal)	14	91	0.01	0.08
Analyzer (Pascal)	330	4903	0.30	4.53
Checker (Pascal)	1290	18853	1.19	17.46
Absolutizer (XSL)	368	4431	0.34	4.10
HTML-izer (XSL)	700	8980	0.65	8.31

Similar Makefile-based parallelization technology is also used by the MaLARea system when trying to solve the ca. fifty thousand Mizar theorem by ATPs, and producing a database of their solutions that is used for subsequent better proof advice and improved ATP solving using machine learning techniques. One possible (and probably very useful) extension for purposes of such fast real-time library re-verification is to extract finer dependencies from the articles (e.g. how theorems depend on other theorems and definitions - this is already to a large extent done e.g. by the MPTP system), and further speed up such re-verification by checking only certain parts of the dependent articles, see [3] for detailed analysis. This is actually also one of the motivations for the parallelization done by splitting articles into independently verified pieces, described in the next section.

5 Parallelization of Single Article Processing

While parallelization of the whole (or large part of) library processing is useful, and as mentioned above it is likely going to become even more used, the main use-case of Mizar processing is when a user is authoring a single article, verifying it quite often. In the case of a formal mathematical wiki, the corresponding use-case could be a relatively limited refactoring of a single proof in a larger article, without changing any of the exported items (theorems, definitons, etc.), and thus not influencing any other proofs in any other article. The need in both cases is then to (re-)verify the article as quickly as possible, in the case of wiki also quickly re-generating the HTML presentation, giving the user a real-time experience and feedback.

5.1 Toplevel Parallelization

As described in Section 3, there are typically several ways how to parallelize various parts of the processing, however it is also explained there that the one which suits best

the *Analyzer* and HTML-ization is a file-based parallelization over the toplevel proofs. This is what was also used in the initial implementation of the Mizar parallelizer[9]. This section describes this implementation (using Perl and LibXML) in more detail.

As can be seen from Table 1 and Table 2, the *Parser* pass is very fast. The total user time for the whole MML in Table 2 is 91.160 seconds, which means that the average speed on a MML article is about 0.1 second. This pass identifies the symbols and the keywords in the text, and the overall block structure, and produces a file that is an input for the much more expensive *Analyzer* pass. Parsing a Mizar article by external tools is (due to the intended closeness to mathematical texts) very hard [5], so in order to easily identify the necessary parts (toplevel proofs in our case) of the formal text, the output of the *Parser* pass is now also printed in an XML format, already containing a lot of information about the proof structure and particular proof positions[10]

The Parallelizer's processing therefore starts by this fast Parser run, putting the necessary information in the XML file. This XML file is then (inside Perl) read by the LibXML functions, and the toplevel proof positions are extracted by simple XPath queries from it. This is also very fast, and adds very little overhead. These proof positions are an input to a (greedy) algorithm, which takes as another input parameter the desired number of processes (N) run in parallel (for compatibility with GNU make, also passed as the `-j` option to the parallelizer). This algorithm then tries to divide the toplevel proofs into N similarly hard groups. While there are various options how to estimate the expected verification hardness of a proof, the simplest and reasonably working one is the number of lines of the proof. Once the toplevel proofs are divided into the N groups, the parallelizer calls Unix `fork()` on itself with each proof group, spawning N child instances.

Each instance creates its own subdirectory (symbolically linking there the neccessary auxiliary files from the main directory), and creates its own version of the verified article, by replacing the keyword `proof` with the keyword `@proof` for all toplevel proofs that do not belong to the proofs processed by this particular child instance. The *Parser* pass is then repeated on such modified input by the child instance, the `@proof` directives producing input for *Analyzer* that contains only the desired toplevel proofs. The costly subsequent passes like the *Analyzer*, *Checker*,and HTML-ization can then be run by the child instance on the modified input, effectively processing only the required toplevel proofs, which results in large speedups. Note that the Parser's work is to some extent repeated in the children, however its work in the skipped proofs is very easy (just counting brackets that open and close proofs), and this pass is in comparison with others very fast and thus negligible. The parallel instances of the Analyzer, Checker, and HTML-ization passes also overlap on the pieces of the formal text that are not inside the toplevel proofs (typically the stated theorems and definitions have to be at least analyzed), however this is again usually just a negligible share of the formal text in comparison with the full text with all proofs.

The speedup measured for the verification (Parser, Analyzer, Checker) passes on the above mentioned article `fdiff_1` run with eight parallel processes `-j8` is given in the Table 3 below. While the total user time obviously grows with the number of parallel processes used, the real verification time is in this case decreased nearly four

[9] `http://github.com/JUrban/MPTP2/raw/master/MizAR/cgi-bin/bin/mizp.pl`

[10] Note that the measurement of Parser speed in the above tables was done after the XMLization of the Parser pass, so the usual objection that printing a larger XML file slows down verification is (as usual) completely misguided, especially in the larger picture of costly operations done in the Analyzer and the Checker.

times. Additionally, in comparison with the notebook processing mentioned in the initial Table 1, the overall real-time benefit of remote parallelized server processing is a speedup factor of 20. This is a strong motivation for the server-based remote verification (and other) services for Mizar implemented in Emacs and through web interface decribed in [32]. The overall statistics done across all (395) MML articles that take in the normal mode more than ten seconds to verify is computed for parallelization with one, two, four, and eight processes, and compared in Table 4. The greatest real-time speedup is obviously achieved by running with eight processes, however, already using two processes helps significantly, while the overhead (in terms of user time ratios) is very low. When all the child instances finish their jobs, the parent parallelizer post-

Table 3. Comparison of the verification speed on article `fdiff_1` run in the normal mode and in the parallel mode, with eight parallel processes (`-j8`)

Article	real (normal)	user (normal)	real (-j8)	user (-j8)
fdiff_1	13.11	12.99	3.54	21.20

Table 4. Comparison of the verification speeds on 395 slow MML articles run with one, two, four, and eight parallel processes

	-j1	-j2	-j4	-j8
Sum of user times (s)	12561.07	13289.41	15937.42	21697.71
Sum of real times (s)	13272.22	7667.37	5165.9	4277.12
Ratio of user time to -j1	1	1.06	1.27	1.73
Ratio of real time to -j1	1	0.58	0.39	0.32

processes their results. In the case of running just verification (Analyzer and Checker), the overall result is simply a file containing the error messages and positions. This file is created just by (uniquely) sorting together the error files produced by the child instances. Merging the HTML-ization results of the child instances is very simple thanks to the mechanisms described in Section 3.3. The `-ajax-proofs` option is used to place the HTMLized proofs into separate files, and depending on the required HTML output, either just bound to AJAX calls in the toplevel HTML-ization, inserting them on-demand, or postprocessing the toplevel HTML in Perl by the direct inclusion of the HTML-ized toplevel proofs into it (creating one big HTML file).

5.2 Finer Parallelization

The probably biggest practical disadvantage of the parallelization based on toplevel proofs is that in some cases, the articles really may consist of proofs with very uneven size, in extreme cases of just one very large proof. In such cases, the division of the toplevel proofs into groups of similar size is going to fail, and the largest chunk is going to take much more time in verification and HTML-ization than the rest. One option is in such cases to recurse, and inspect the sub-proof structure of the very long proofs, again, trying to parallelize there. This was not done yet, and instead, the Checker-based parallelization was implemented, providing speedup just for the most expensive Checker pass, but on the other hand, typically providing a very large parallelization possibility. This is now implemented quite similarly to the toplevel proof parallelization, by modifying the intermediate XML file passed from the Analyzer to the Checker. As with the `@proof` user-provided directive, there is a similar internal directive usable

in the XML file, telling the Checker to skip the verification of a particular atomic inference. This is the used very similarly to `@proof`: The parallelizer divides the atomic inferences into equally sized groups, and spawns N children, each of them modifying the intermediate XML file, and thus checking only the inferences assigned to the particular child. The errors are then again merged by the parent process, once all the child instances have finished.

The overall evaluation of this mode done again across all (395) MML articles that take in the normal mode more than ten seconds to verify is shown in Table 5 for (checker-only) -j8, and compared with the (toplevel) -j8 from Table 4 where the toplevel parallelization mode is used. The data confirm the general conjecture from Section 3.2: A lot of Mizar's work is done in the type analysis module, and the opportunity to parallelize that is missed in the Checker-only parallelization. This results in lower overall user time (less work repetition in analysis), however higher real time (time perceived by the user). This parallelization is in some sense orthogonal to the toplevel

Table 5. Comparison of the toplevel and checker-only verification speeds on 395 slow MML articles run with one and eight parallel processes

	-j1	-j8 (toplevel)	-j8 (checker-only)
Sum of user times (s)	12561.07	21697.71	18927.91
Sum of real times (s)	13272.22	4277.12	5664.1
Ratio of user time to -j1	1	1.73	1.51
Ratio of real time to -j1	1	0.32	0.43

proof parallelization, and it can be used to complement the toplevel proof parallelization in cases when there are for instance only two major toplevel proofs in the article, but the user wants to parallelize more. I.e., it is no problem to recurse the parallelizer, using the Checker-based parallelization for some of the child instances doing toplevel-proof parallelization.

6 Related Work

As already mentioned, sophisticated parallelization and strategy scheduling have been around in some ATP systems for several years now, an advanced example is the infrastructure in the Waldmeister system [8]. The Large Theory Batch (LTB) division of the CADE ATP System Competition has started to encourage such development by allowing parallelization on multicore competition machines. This development suits particularly well the ATP/LTB tasks generated in proof assistance mode for Mizar. Recent parallelization of the Isabelle proof assistant and its implementation language are reported in [12] and in [34], focusing on fitting parallelism within the LCF approach. This probably makes the setting quite different: [34] states that *there is no magical way to add the "aspect of parallelism" automatically*, which does not seem to be the case with the relatively straightforward approaches suggested and used here for multiple parts of Mizar and related processing. As always, there seems to be a trade-off between (in this case LCF-like) safety aspirations, and efficiency, usability, and implementation concerns. Advanced ITP systems are today much more than just simple slow proof checkers, facing similar "safety" vs. "efficiency" issues as ATP systems [13]. The Mizar philosophy favors (sometimes perhaps too much) the latter, arguing that there are always enough ways how to increase certainty, for example, by cross-verification

as in [31], which has been recently suggested as a useful check even for the currently safest LCF-like system in [1]. Needless to say, in the particular case of parallelization a possible error in the parallelization code is hardly an issue for any proof assistant (LCF or not) focused on building large libraries. As already mentioned in Section 2, at least in case of Mizar the whole library is typically re-factored and re-verified many times, for which the safe file-based parallelization is superior to internal parallelization also in terms of efficiency, and this effectively serves as overredundant automated cross-verification of the internal parallelization code.

7 Future Work and Conclusions

The parallelizer has been integrated in the Mizar mode for Emacs [24] and can be used instead of the standard verification process, provided that Perl and LibXML are installed, and also in the remote server verification mode, provided Internet is available. The speedups resulting from combination of these two techniques are very significant. As mentioned above, other Mizar utilities than just the standard verifier can be parallelized in exactly the same way, and the Emacs environment allows this too. The solutions described in this paper might be quite Mizar-specific, and possibly hard to port e.g., to systems with non-opaque proofs like Coq, and the LCF-based provers, that do not use similar technique of compilation-like passes. Other, more mathematician-oriented Mizar-like systems consisting of separate linguistic passes like SAD/ForThel [11] and Naproche [6] might be able to re-use this approach more easily.

As mentioned above, another motivation for this work comes from the work on a wiki for formal mathematics, and for that mode of work it would be good to have finer dependencies between the various items introduced and proved in the articles. Once that is available, the methods developed here for file-based parallelization will be also usable in a similar way for minimalistic checking of only the selected parts of the articles that have to be quickly re-checked due to some change in their dependencies. This mode of work thus seems to be useful to have not just for Mizar, but for any proof assistant that would like to have its library available, editable, and real-time verifiable in an online web repository.

References

1. Mark Adams. Introducing HOL Zero – (extended abstract). In Komei Fukuda, Joris van der Hoeven, Michael Joswig, and Nobuki Takayama, editors, *ICMS*, volume 6327 of *LNCS*, pages 142–143. Springer, 2010.
2. Jesse Alama, Kasper Brink, Lionel Mamane, and Josef Urban. Large formal wikis: Issues and solutions. In James H. Davenport, William M. Farmer, Josef Urban, and Florian Rabe, editors, *Calculemus/MKM*, volume 6824 of *LNCS*, pages 133–148. Springer, 2011.
3. Jesse Alama, Lionel Mamane, and Josef Urban. Dependencies in formal mathematics: Applications and extraction for Coq and Mizar. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *AISC/MKM/Calculemus*, volume 7362 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012.
4. Serge Autexier, Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, editors. *Intelligent Computer Mathematics, 10th International Conference, AISC 2010, 17th Symposium, Calculemus*

2010, and 9th International Conference, MKM 2010, Paris, France, July 5-10, 2010. Proceedings, volume 6167 of *LNCS*. Springer, 2010.
5. Paul A. Cairns and Jeremy Gow. Using and parsing the Mizar language. *Electr. Notes Theor. Comput. Sci.*, 93:60–69, 2004.
6. Marcos Cramer, Bernhard Fisseni, Peter Koepke, Daniel Kühlwein, Bernhard Schröder, and Jip Veldman. The Naproche Project: Controlled Natural Language Proof Checking of Mathematical Texts. In Norbert E. Fuchs, editor, *CNL*, volume 5972 of *LNCS*, pages 170–186. Springer, 2009.
7. Adam Grabowski, Artur Korniłowicz, and Adam Naumowicz. Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.
8. Thomas Hillenbrand. Citius altius fortius: Lessons learned from the theorem prover WALDMEISTER. *Electr. Notes Theor. Comput. Sci.*, 86(1):9–21, 2003.
9. Cezary Kaliszyk and Josef Urban. MizAR 40 for Mizar 40. *CoRR*, abs/1310.2805, 2013.
10. Cezary Kaliszyk, Josef Urban, and Jiri Vyskocil. Machine learner for automated reasoning 0.4 and 0.5. *CoRR*, abs/1402.2359, 2014. Accepted to PAAR'14.
11. Alexander V. Lyaletski and Konstantin Verchinine. Evidence algorithm and system for automated deduction: A retrospective view. In Autexier et al. [4], pages 411–426.
12. David C. J. Matthews and Makarius Wenzel. Efficient parallel programming in poly/ml and isabelle/ml. In Leaf Petersen and Enrico Pontelli, editors, *DAMP*, pages 53–62. ACM, 2010.
13. William McCune and Olga Shumsky. Ivy: A preprocessor and proof checker for first-order logic. In Matt Kaufmann, Panagiotis Manolios, and J Strother Moore, editors, *Computer-Aided Reasoning: ACL2 Case Studies*, pages 265–281. Kluwer Academic, June 2000.
14. Konrad Raczkowski and Paweł Sadowski. Real function differentiability. *Formalized Mathematics*, 1(**4**):797–801, 1990.
15. Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *Journal of AI Communications*, 15(2-3):91–110, 2002.
16. Piotr Rudnicki and Andrzej Trybulec. Abian's fixed point theorem. *Formalized Mathematics*, 6(**3**):335–338, 1997.
17. Piotr Rudnicki and Andrzej Trybulec. On equivalents of well-foundedness. *J. Autom. Reasoning*, 23(3-4):197–234, 1999.
18. Piotr Rudnicki and Josef Urban. Escape to ATP for Mizar. In Pascal Fontaine and Aaron Stump, editors, *First International Workshop on Proof eXchange for Theorem Proving - PxTP 2011*, Wroclaw, Poland, 2011.
19. Stephan Schulz. E - A Brainiac Theorem Prover. *AI Commun.*, 15(2-3):111–126, 2002.
20. G. Sutcliffe and D. Seyfang. Smart Selective Competition Parallelism ATP. In A. Kumar and I. Russell, editors, *Proceedings of the 12th International FLAIRS Conference*, pages 341–345. AAAI Press, 1999.
21. Geoff Sutcliffe. The Design and Implementation of a Compositional Competition-Cooperation Parallel ATP System. In H. de Nivelle and S. Schulz, editors, *Proceedings of the 2nd International Workshop on the Implementation of Logics*, number MPI-I-2001-2-006 in Max-Planck-Institut für Informatik, Research Report, pages 92–102, 2001.
22. Josef Urban. MPTP – Motivation, Implementation, First Experiments. *Journal of Automated Reasoning*, 33(3-4):319–339, 2004.

23. Josef Urban. XML-izing Mizar: Making semantic processing and presentation of MML easy. In Michael Kohlhase, editor, *MKM*, volume 3863 of *Lecture Notes in Computer Science*, pages 346–360. Springer, 2005.
24. Josef Urban. MizarMode – an integrated proof assistance tool for the Mizar way of formalizing mathematics. *Journal of Applied Logic*, 4(4):414–427, 2006.
25. Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *J. Autom. Reasoning*, 37(1-2):21–43, 2006.
26. Josef Urban. MaLARea: a metasystem for automated reasoning in large theories. In Geoff Sutcliffe, Josef Urban, and Stephan Schulz, editors, *ESARLT: Empirically Successful Automated Reasoning in Large Theories*, volume 257 of *CEUR Workshop Proceedings*, pages 45–58. CEUR, 2007.
27. Josef Urban. BliStr: The Blind Strategymaker. *CoRR*, abs/1301.2683, 2014. Accepted to PAAR'14.
28. Josef Urban, Jesse Alama, Piotr Rudnicki, and Herman Geuvers. A wiki for mizar: Motivation, considerations, and initial prototype. In Autexier et al. [4], pages 455–469.
29. Josef Urban, Krystof Hoder, and Andrei Voronkov. Evaluation of automated theorem proving on the Mizar Mathematical Library. In *ICMS*, pages 155–166, 2010.
30. Josef Urban, Piotr Rudnicki, and Geoff Sutcliffe. ATP and presentation service for Mizar formalizations. *J. Autom. Reasoning*, 50:229–241, 2013.
31. Josef Urban and Geoff Sutcliffe. ATP-based cross-verification of Mizar proofs: Method, systems, and first experiments. *Mathematics in Computer Science*, 2(2):231–251, 2008.
32. Josef Urban and Geoff Sutcliffe. Automated reasoning and presentation support for formalizing mathematics in Mizar. In Autexier et al. [4], pages 132–146.
33. Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jirí Vyskocil. Malarea sg1- machine learner for automated reasoning with semantic guidance. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2008.
34. Makarius Wenzel. Parallel proof checking in Isabelle/Isar. The ACM SIGSAM 2009 PLMMS Workshop.

Part III

Optimization Techniques and Decision-making Processes

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Optimization Problems and Methods Applicable in Intelligent Tourist Travel Planners

Jolanta Koszelew

Faculty of Computer Science
Białystok University of Technology
15-351 Białystok, Wiejska 45 a
j.koszelew@pb.edu.pl

Abstract. The paper describes the project of innovative software component – LOGTRAVEL which can be applied as a logistic toolbox for e-tourism systems called Tourist Travel Planners (TTP). Functionalities of LOGTRAVEL supports planning and organization of cheap and attractive touristic travels which tourists can use in TTP system. The component includes solutions of many variants and extensions of Orienteering Problem which enable a generation of an optimal trip satisfying variety traveler preferences. The paper has the survey character and is to the definition of the problems and their application and does not present solutions for them.

Keywords: Tourist Travel Planners, Orienteering Problem, Point of Interest

1 Introduction

According to the actual trends on the market, expected conditions in the development of e-services in the next five years, include social networks and thematic hobby that can grow despite the huge competition and general social networks portals in the area of tourism and travel. The increase in the number of users of an e-tourism or web-based systems on the theme of tourism, is already noticeable and only in Poland is about 10 percents per year over the past two years.

The trends described above fits perfectly the need for a social networks site for fans of expeditions organized yourself. There are many travel portals for individual tourists. The largest one in Poland has less than 25 thousand users, which appears to be the result of the weak against the statistics showing the number of Poles going as a tourist and not using the travel agency services. These people were in Poland, almost 3.3 million in 2009, including nearly 2 million followers it yourself organized trip. What is the reason for this relatively little interest in Polish travel portals? In the opinion of many users of these sites expressed in online forums, they lack the functionality to support the planning, organization and monitoring of travel.

In all Polish web portals is implemented the same idea of gathering unspecified information in the form of relationships with travel, photos and videos. The user can only select a location or region that wants to visit, read reviews or memories, view photos and videos added by people who visited this place during his departure. Sometimes

the user who presented their relations with the expedition is also possible to issue the scoring visited tourist objects.

This paper presents a proposal for the travel of innovative functionalities, supporting logistics processes occurring during route planning tourist trips organized independently. The implementation of these functions is the subject of research conducted at the Faculty of Computer Science, Bialystok University of Technology, which will result in the software component (library) called LOGTRAVEL.

2 LOGTRAVEL functionalities

The result of the application functionality is implemented by LOGTRAVEL path (or set of paths) with the highest possible degree of tourist attractiveness, fulfilling a fixed set of constraints and preferences set by the user. It is important that the method of generating routes were efficient in time, both when they are applied to the area of the type of the continent, as well as when they are run for smaller territorial regions such as the state or city. The implementation of component functionality LOGTRAVEL requires the solution of many optimization problems on graphs. These problems are variations of a computationally difficult task called Orienteering Problem (OP) [9], also known under the Selective Travelling Salesman Problem with Profits ($STSPwP$) [2]. Vertices of the graph that models the problem of the OP are tourist attractions (Points of Interest, POI in brief) with a certain degree of attractiveness and geographical location. Types of tourist attractions can be very different from the monuments, museums or starting viewpoints, and in hotels, guest houses or clubs ending entertainment. In addition to the degree of attractiveness and geographical location of the attraction may be related to other information, such as ticket prices or accommodation or hours away. The edges of the graph correspond to connections between attractions (ie road, air transport, walking, public transport, etc.). The edges are assigned weights indicating the length of the call, expressing the distance and /or time of the merger. If the connection is implemented using public transport timetables, the value of connections depend on the time of departure. LOGTRAVEL realize the following functionalities:

- $SelectingandRouting(SaR)$: Generate routes for the highest possible total degree of attractiveness of the objects contained in the route. Length of the route may not exceed the set limit (time or length of communication links) [7].
- $PossibilitiesofReturns$ (POR): The generated route may further comprise recurring attractions , but on subsequent presentations of the same objects degree of attractiveness is zero [2]. Functionality useful in networks communication links , which are modeled part-graph , i.e. one in which there may be no direct connection between at least one pair of objects.
- $ObligatoryPOIs$ (OPs): Generate routes, containing additional mandatory attractions of the status of "must see". The user defines a set of activities that wants mandatory visit in route [1], [16].
- $OpeningHours$ (OH): Generate a route taking into account the additional access time of interest for inclusion in the route [12].
- $ScenicRoutes$ (SR): Generating route, which also has the highest possible value of viewing dialed communication, if links in the graph are assigned values of the attractiveness of the observation [11].
- $PublicTransportation$ (PT): Generating route, which also may be fully or partially developed using selected measures of public transport (ie train, plane, bus, etc.) with specific timetables [6].

- *BudgetLimitations* (BL): Generate route, which also satisfies the budget allocated for the route [1].
- *DynamicRecalculation* (DR): functionality needed in the event of unforeseen events earlier (eg the delay of the aircraft) and the modification is required originally planned route in "real-time" [12].
- $Max - nType$ (MnT): Reducing the number of points of interest in a particular type of route (eg route can occur up to three museums) [1].
- *ObligatoryTypes* (OT): Specify the type of activities that must occur at least once in the itinerary, such as the route must occur at least one historic church [1].

Currently LOGTRAVEL includes enforcing functions *SelectingandRouting* and *PossibilitiesofReturns*. Other functionality of the library will be implemented and tested soon. Fig. 1 shows the screenshot of the application testing component functionality LOGTRAVEL realized. Tests were carried out on real data from the region of Podlasie. Solutions to problems implementing the functionality SaR and times are based on the use of artificial intelligence methods (AI in brief), namely genetic algorithms [3]. Testing time complexity and accuracy of the resulting slopes confirm the high quality of the solutions used. A test network consisted of nearly six tourist attractions. Execution time of a single query about the route attractive to tourists (with drawn results on a map, geographic interface) does not exceed a few seconds, regardless of the route length limit.

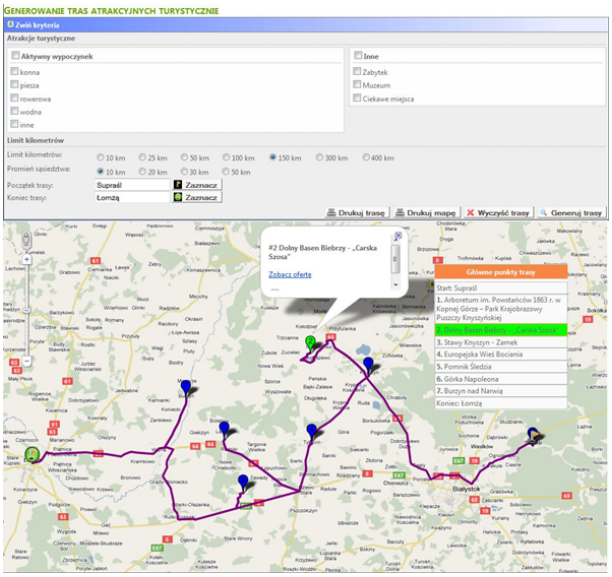


Fig. 1. Generating attractive tourist routes

None of Polish travel portals does not implement even one of the above functionality. In the literature you can find information about many solutions in systems with attractive tourist route engine. Table.1 lists works which relate to the discussed issues in this paper and noted that the consideration of functionality are included in them.

Unfortunately, most of the work has not been completed and their effect are only prototypes of the software. Logistical support "globetrotters" can be realized not only by tools supporting planning travel routes, but also the software that enables the creation of (organizing) group of travel and its continuous contact at the stage of determining the organizational details of travel. IT support is also necessary to monitor the travel in the course of the route, in terms of compliance with the planned trip parameters actually implemented. However, it seems that the functionality associated with travel arrangements and its monitoring are not as complex research problem as the process of planning a route.

2.1 Tourist Tour Generating Problem

In this section we define the most complicated problem which we can defined for TTP systems. We will called this problem Tourist Tour Generating Problem (TTGP). TTGP is a significant generalization of the Team Orienteering Problem with Time Windows (TOPTW) [19], which, in turn, is an extension of the Team Orienteering Problem (TOP) [17]. TTGP is an original new problem with an empty set of solutions now.

authors solutions	SaR	PoR	OPs	OH	SR	PT	BL	DR	MnT	OT
V. W. Soo i S. H. Liang [14]	2001	X								
Y. Suna i L. Lee [15]	2004	X								
A. Maruyama et al [9]	2004	X			X					
K. Hagen et al [4]	2005	X			X					
T. Shiraishi et al [13]	2005	X			X		X	X		
T. Kinoshita et al [5]	2005	X			X					
M. Nagata et al [11]	2006	X			X					
L. Castillo et al [1]	2008	X	X	X		X	X		X	
A.S. Niaraki i K. Kim [11]	2009					X				
C. S. Lee et al [7]	2009	X			X					

Table 1. Overview of the functionality of planning travel routes

Vansteenwegen et al [19] conducted a comprehensive analysis of solutions for OP and its extensions. Tang [18] analysed the published approaches to solutions of the OP family, which attracted the attention of many research teams in the last two years due to practical applications and computation difficulty [17].

Since OP is a NP-hard problem, it is necessary to develop approximate solutions also for all the extensions of the problem. Although many efficient hybrid approaches solving TOP [19] have been presented in literature, those approaches do not have efficient applications in solutions for $TOPTW$ and $TTGP$. Until now, only few algorithms solving $TOPTW$ and only one algorithm for $TTGP$ have been published; besides, they are only for small networks (up to 200 nodes). The best approaches to $TOPTW$ solutions are used by: the $AntColonySystem$ (ACS)proposed by Montemanniand Gambardella [18], local search heuristics ($IteratedLocalSearch$ – ILS)] and the method of $VariableNeighborhoodSearch$ (VNS) [2].

The application of $TTGP$ in particular allows for generating a route that is attractive from the tourist point of view (with the highest total of profits/scores of the

objects) once the user has determined a system of constraints and route preferences, i.e. tour duration or budget. If the route takes more than one day, the system also determines optimal accommodation places. It is worth noting that all the test results for varieties of OP available in literature only refer to small networks. Benchmarks used for tests do not exceed 200 nodes for the problems of OP, TOP and $TOPTW$ and 85 nodes for $TTGP$. On the other hand, applications of solutions to OP [17] problems are a strong motivation for working on time-efficient solutions to those problems for much larger networks, even up to 600 nodes (e.g. the $ITRS$ system for Podlasie region necessitates including about 600 POIs and accommodation places in the network).

Therefore: For further development of $TTGP$ applications, it is essential to develop efficient solutions to this problem dedicated to large networks. In the opinion of the applicants, it is possible to create new efficient and universal algorithms solving $TTGP$, both for small and large networks, i.e. ones including up to 600 nodes.

The formal definition of $TTGP$ is following:
Let $G = (V, E)$ be a connected graph with kinds of vertices $V = M \cup N$. $R = \{1, \ldots, m\}$ represents red ($POIs$ in touristic planners) nodes,
$B = \{m + 1, \ldots, m + n\}$ represents black nodes (hotels int touristic planners),
$E = \{(i, j) : i, j \in V\}$ represents edges.
For ease of exposition, we assume that only one red node is selected for the entire tour. However, our formulation and method can be extended to more general cases in which more than one red node may be selected.
We define:
C – the total budget available for the tour,
d – the number of days available for the tour, and let $D = 1, \ldots, d$,
T_k – the tour time available during the k-th part ($k \in D$),
i – the tour time that are spending at vertex i, $i \in V$, ($i = 0$ for $i \in R$),
(a_i, b_i) – the time period during is allowed to visit vertex $i \in V$, and we assume that it is the same every part of the tour,
c_i – the cost associated with visiting (or staying at) vertex $i \in V$,
e_{ij} – the cost associated with the edge between site i and j , $i, j \in V$,
U_i – the profit of visiting (or staying at) vertex $i \in V$,
t_{ij} – the edge time between vertices i and j , $i, j \in V$.
TPP is to construct a d-part tour that maximizes the total profit of all nodes visited while satisfying the following constraints:
-– the number of part for the tour is d;
-– for each part, the tour starts and ends at the red node;
-– each black node is visited no more than once;
-– the arrival and the departure time at each black node is restricted by its permitted visiting time $[a_i, b_i]$;
-– the cumulative tour time of the kth part does not exceed the available tour time T_k;
-– the total cost of the d-part tour does not exceed the budget C.
We now introduce the following decision variables:

$$x_{ijk} = \begin{cases} 1 \text{ if the edge } (i,j) \text{ belongs to the } k\text{th part of the tour} \\ 0 \text{ otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 \text{ if the node } (i) \text{ belongs to the tour} \\ 0 \text{ otherwise} \end{cases}$$

$$z_i = \begin{cases} 1 \text{ if the red node } (i) \text{ belongs to the tour} \\ 0 \text{ otherwise} \end{cases}$$

t_i = the time epoch at which the node i is start visit,
t^a_{ik} = the time epoch at which ends visit at the red node ion the $k$part of the tour,
t^b_{ik} = the time epoch at which the tourist starts visit at red node i on the $k$part of the tour.
TPP can be formulated as the following mixed integer linear programming (MILP):

$$\text{max of the value } \sum_{i\in N} U_i y_i + \sum_{i\in M} U_i z_i \tag{1}$$

$$\text{such that } \sum_{i\in V} x_{irk} = \sum_{j\in V} x_{rjk} \qquad r \in N, k \in D \tag{2}$$

$$\sum_{i\in N} x_{irk} = \sum_{j\in N} x_{rjk} \qquad r \in M, k \in D \tag{3}$$

$$y_i \in \sum_{k\in D} \sum_{j\in V} x_{ijk} \qquad i \in N \tag{4}$$

$$z_i = \frac{1}{d} \sum_{k\in D} \sum_{j\in N} x_{ijk} \qquad i \in M \tag{5}$$

$$\sum_{i\in M} z_i = 1 \tag{6}$$

$$t_i + \tau_i + t_{ij} - (1 - x_{ijk})T_k \leqslant t_j \qquad i, j \in N, k \in D \tag{7}$$

$$t^a_{ik} + t_{ij} - (1 - x_{ijk})T_k \leqslant t_j \qquad i \in M, j \in N, k \in D \tag{8}$$

$$t_i + \tau_i + t_{ij} - (1 - x_{ijk})T_k \leqslant t^b_{jk} \qquad i \in N, j \in M, k \in D \tag{9}$$

$$a_i \leqslant t_i \leqslant b_i \qquad i \in N \tag{10}$$

$$a_k \leqslant t^a_{ik} \leqslant t^b_{ik} \leqslant b_k \qquad i \in M, k \in D \tag{11}$$

$$\sum_{i\in N}(c_i y_i + \sum_{k\in D} \sum_{j\in V} e_{ij} x_{ijk}) + \sum_{i\in M}(dc_i z_i + \sum_{k\in D} \sum_{j\in N} e_{ij} x_{ijk}) \leqslant C \tag{12}$$

$$x_{ijk} \in \{0, 1\} \qquad i, j \in V, k \in D \tag{13}$$

$$y_i \in \{0, 1\} \qquad i \in N \tag{14}$$

$$z_i \in \{0, 1\} \qquad i \in M \tag{15}$$

In the above $MILP$, the objective function (1) maximizes the total utility of all sites visited by the tourist and the constraints are:
- Constraint (2) is the balance equation, which ensures that if the node is visited, it is left,
- Constraint (3) ensures that each part of tour starts and ends at the same red node;
- Constraints (4) and (5) state the relationship between x_{ijk}, y_i and z_i;
- Constraint (6) ensures that only one red node is selected for the entire tour;
- Constraints (7)–(11) are the time constraints;
- Constraint (12) is the budget constraint.

We note that $TTGP$ is related to the classical Vehicle Routing Problem (VRP). In VRP, the objective is to design the routes for a fleet of vehicles to service all customers with minimum total cost, subject to vehicle capacity and time constraints. The cost in VRP can be either the number of vehicles used or the total length of the routes. In graphical theory terminology, VRP is to find a set of cycles with minimum total

cost to cover all the vertices. By comparison, $TTGP$ is to generate a fixed number of cycles such that the total utility value of the vertices covered by these cycles is maximized. 4. Conclusions and Future Work The set of problems which are applied in TTP systems includes many versions of OP. Each of these versions is NP-hard, of course and we need to develop approximate solutions for them, in particular for big networks. The future work is planning as follows: 1) Development of algorithms solving $TTGP$ Generally, two methods will be developed: an evolution algorithm and a hybrid algorithm. The applicants have successfully used the evolutionary approach to the solution of a specific version of TPP, i.e. OP. Although TPP has many additional constraints in comparison to TOP and $TOPTW$, the use of the ideologically similar evolutionary approach seems highly proper to the applicants. The other -– hybrid -– approach results from the fact that the most efficient solutions to simplifications of $TTPG$ do not use a popularization strategy but actions aiming at changing an individual copy of solutions. In the next phase,versions of evolution and hybrid algorithm efficient for particular cases of $TTGP$, i.e. TOP and $TOPTW$, will be developed. 2) Studying the efficiency of the developed algorithms The solutions will be efficiently implemented and then thoroughly tested regarding the total profit of the route profile and execution time. The tests will be conducted on two large networks (300 and 600 nodes) and benchmarks (up to 100 nodes) on which the solution of Zhu's team and solutions for specific $TTGP$ cases were tested [20].

3 Conclusions and Future Work

The set of problems which are applied in TTP systems includes many versions of OP. Each of these versions is NP-hard, of course and we need to develop approximate solutions for them, in particular for big networks. The future work is planning as follows:

1. Development of algorithms solving $TTGP$ Generally, two methods will be developed: an evolution algorithm and a hybrid algorithm. The applicants have successfully used the evolutionary approach to the solution of a specific version of TPP, i.e. OP [17]. Although TPP has many additional constraints in comparison to TOP and $TOPTW$, the use of the ideologically similar evolutionary approach seems highly proper to the applicants. The other – hybrid – approach results from the fact that the most efficient solutions to simplifications of $TTPG$ do not use a popularization strategy but actions aiming at changing an individual copy of solutions. In the next phase,versions of evolution and hybrid algorithm efficient for particular cases of $TTGP$, i.e. TOP and $TOPTW$, will be developed.
2. Studying the efficiency of the developed algorithms The solutions will be efficiently implemented and then thoroughly tested regarding the total profit of the route profile and execution time. The tests will be conducted on two large networks (300 and 600 nodes) and benchmarks (up to 100 nodes) on which the solution of Zhu's team and solutions for specific $TTGP$ cases were tested [20].

References

1. Castillo L., Armengol E., Onainda E., Sebastia L., Gonzalez-Boticario J.: Rodriguez A., Fernandez S., Arias J. D., Borrajo D.: An user-oriented adaptive system for planning tourist visits, Expert Systems with Applications, Vol. 34, 2008, pp. 1318–1332.

2. Feillet D., Dejax P., Gendreau M.: Traveling Salesman Problems with Profits, Transportation Science, Vol 39(2), 2005, pp. 188–205.
3. Goldberg D. E.: Algorytmy genetyczne i ich zastosowania, Warszawa: WNT, 1998.
4. Hagen K., Kramer R., Hermkes M., Schumann B., Mueller P.: Semantic matching and heuristic search for a dynamic tour guide. Information and Communication Technologies in Tourism, Springer, 2005.
5. Kinoshita T., Nagata M., Shibata N., Murata Y., Yasumoto K., Ito M.: A personal navigation system for sightseeing across multiple days. W Proc. of the 3rd Int'l. Conf. on Mobile Computing and Ubiquitous Networking (ICMU2006), 2006, pp. 254–259.
6. Koszelew J., Piwonska A.: A New Evolutionary Algorithm for Routes Generation with Optimal Time of Realization in Public Transport Network, Journal of Applied Computer Science, Vol 18(2), 2010, pp. 7–23.
7. Lee C. S., Chang Y. C., Wang M. H.: Ontological recommendation multi-agent for tainan city travel, Expert Systems with Applications, Vol 36, 2009, pp. 6740–6753.
8. Li W. H., Zhenping L., Wang R. S., Zhou W.: Models and Algorithms for the Constrained Orienteering ProblemThe Ninth International Symposium on Operations Research and Its Applications, (ISORA'10), Chengdu-Jiuzhaigou, China, 2010, pp. 89–97.
9. Nagata N., Murata Y., Shibata N., Yasumoto K.: Simulated Evolution and Learning, LNCS, 4247, A Method to Plan Group Tours with Joining and Forking, Springer Berlin-Heidelberg, 2006, pp. 881–888.
10. Shiraishi T., Nagata M., Shibata N., Murata Y., Yasumoto K., Ito M.: A personal navigation system with a schedule planning facility based on multi-objective criteria. In: Proceedings of 2nd International Conference on Mobile Computing and Ubiquitous Networking, 2005, pp.104–109.
11. Shiraishi T., Nagata M., Shibata N., Murata Y., Yasumoto K., Ito M.: A personal navigation system with functions to compose tour schedules based on multiple connecting criteria, IPSJ Digital Courier, nr 1, 2005, pp. 528–536.
12. Suna Y., Lee L.: Agent-based personalized tourist route advice system, In ISPRS Congress Istanbul 2004, Proceedings of Commission II, 2004, pp. 319–324.
13. Archetti C., Hertz A., Speranza M.: Metaheuristics for the team orienteering problem, Journal of Heuristics, 13, 2007, pp. 49–76.
14. Buhalis D.: eTourism: information technology for strategic tourism management, London: Prentice Hall, 2003.
15. Buhalis D., Law R.: Twenty years on and TEN years after the internet: the state of eTourismresearch, Tourism Management, 29(4), 2008, pp. 609–623.
16. Chao I., Golden B., Wasil E: The team orienteering problem, European Journal of Operational Research, 88, 1996, pp. 464–474.
17. Souffriau W., Vansteenwegen P., VandenBerghe G., Van Oudheusden D.: A greedy randomized adaptive search procedure for the team orienteering problem, In Proceedings of EU/Meeting 2008, France, 23-24 October 2008.
18. Tang H., Miller-Hooks E.: A tabu search heuristic for the team orienteering problem, Computers and Operations Research 32 (6), 2005, pp. 1379–1407.
19. Vansteenwegen P., Souffriau W., VandenBerghe G., Van Oudheusden D.: The City trip planner: An expert system for tourists, Expert Systems with Applications 38 (6), 2011, pp. 6540–6546.
20. Zhu, C., Hu J.Q., Wang F., YifanXu Y., Cao, R.: On the tour planning problem, Ann Oper Res, Springer Science+Business Media, 192, 2012, pp. 67–95.

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Combining Genetic Algorithm and Path Relinking for Solving Orienteering Problem with Time Windows

Joanna Karbowska-Chilińska and Paweł Zabielski

Białystok University of Technology, Faculty of Computer Science, Poland

Abstract. Path relinking is a search strategy that explores trajectories connecting two solution to generate new best solutions. In this article two options combining path relinking and genetic algorithm are investigated: one introduced a path relinking between selected generations of population, and the other applies path relinking when genetic algorithm solution trapped in a local optimum. These two strategies are applied with the genetic algorithm solving orienteering problem with time windows. Experiments carried out on benchmark instances show that proposed methods obtain better solutions than standard genetic algorithm.

Keywords: orienteering problem with time windows, genetic algorithm, local optimum, path relinking

1 Introduction

The path relinking strategy (PR) was proposed by Glover and Laguna [4] as a method for intensification and diversification tabu search method. Path reliniking starts from selecting initial and guiding solutions to represent the starting and the ending points of the path. Attributes from the guiding solution are gradually introduced into the intermediate solutions, so that these solutions contain less characteristics from the initial solution and more from the guiding solution. As a result, the selective moves between initial and guiding solutions provide the better solutions than the input solutions.

In the literature conjunction PR with different metaheuristics for solving different optimisation problems have been regarded e.g. with greedy randomized adaptive search procedure (GRASP) [2], variable neighbour search (VNS) [14], genetic algorithms (GA) [5], [13] and tabu search (TS) [6].

In this paper a hybrid heuristic is proposed by combining GA and PR and applying this method to orienteering problem with time windows (OPTW). The OPTW is a well-known optimization graph problem, which was first introduced by Kantor [7]. In OPTW a given positive profit (score) and time interval are associated with each graph vertex. The solution to the OPTW finds a route comprising a subset of the vertices, with a fixed limit on length or travel time, that maximises the cumulative score of the vertices visited in the predefined time intervals. Different application for this problem can be found e. g. in logistics for planing optimal routes and delivers [15], in tourism- to plan a most profitable tourist routes in a city [18], taking into account opening and closing time points of interests. The OPTW is variant of the widely studied more general

orienteering problem (OP) in which vertices could be visited in any time interval [17], [8]. Another extension of OP is Team Orienteering Problem (TOP) in which m optimal routes are build, each one satisfied a given cost limit [1]. Furthermore in the literature, another variant with time windows so-called TOPTW is dealt [12], [11]. All these problems are NP-hard [19] so exact solutions are time-consuming and not applicable in practice. Therefore meta-heuristic approaches are usually used e. g. genetic algorithms [9], local search methods [19], tabu search [16], ant colony optimisation approach [12], variable neighbour search [11].

In our previous work we used path relinking instead a crossover in each iteration of a genetic algorithm solving OPTW [10]. The proposed modification gave high quality results for benchmark instances in comparison to the other heuristic methods and the genetic algorithm with crossover. In this article we develop this idea. Two methods are proposed: one introduced a path relinking between selected iterations of the GA, the other method applies path relinking when GA trapped in a local optimum. The numerous experiments are carried out on the benchmark instances and the results are compared to the previous methods.

The remainder of the paper is organised as follows. Section 2 provides the problem definition and the example. Section 3 describes the concept of combining genetic algorithm and path relinking. Section 4 presents the algorithm details. The results of computational experiments run on benchmark datasets are discussed in Section 5. Finally, section 6 concludes that article with closing remarks and plans for future work.

2 Formulation of the OPTW

The OPTW examined in this paper is defined as follows. Let G be a graph with n vertices, each vertex i has a profit p_i and a service time T_i. Moreover for each vertex a time window $[O_i, C_i]$ is assigned, where O_i and C_i denote the opening and closing times of a vertex i. The early arrivals lead to waiting for service, whereas late arrivals cause infeasibility. Let t_{ij} be a fixed cost associated to the edge between vertices i and j. This value is interpreted as the time or length needed to travel between vertices. The OPTW consists of determining a single route, from a starting point s to a fixed ending point e, that visits some of the vertices within the fixed time windows and maximises the total profit. In addition, the total time related to visiting vertices and edges on a route must be less than the given limit t_{max} and each vertex on the route is visited only once. The mathematical formulation of the OPTW problem can be found in [7]. To illustrate the problem a simple example of the weighted graph is showed in Figure 1. The profit value p_i and time windows $[O_i, C_i]$ are marked next to each vertex. The t_{ij} value is assigned on each edges. A visiting time for each vertex is given in the following vector T=[0, 12, 18, 12, 6, 12, 3]. The value of t_{max} is 96. Let vertex 1 be the starting point of the route as well as this vertex be the ending point. The best solution of the OPTW is the route 1-7-2-3-5-1, with the total profit 70 and travel time equal to 81.

3 Combining genetic algorithm and path relinking

The presented approach is based on genetic algorithm (GA) for OPTW proposed by us in [8]. The previous GA started from a population of randomly generated routes. A route is coded as a sequence of vertices. In each generation, the fitness of every route

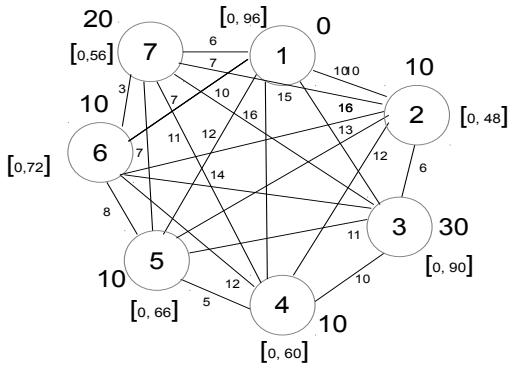


Fig. 1. Graph example to illustrate the OPTW problem.

was evaluated. In each iteration of the algorithm the best routes were selected by a tournament selection from the current population and the random routes were modified by a mutation and crossover. The algorithm terminated when maximum number of generations was produced or earlier if it coverages. In the previous version of the algorithm called GAPR [10], the PR method instead the crossover operator was applied to GA. In the PR two routes were selected randomly. Next the vertices not presented in one route are inserted in the best positions to the other. Moreover if shift resulting from the new insertion exceed the time windows for vertices located after the newly inserted then old vertices were removed to restore the possibility of inserting new vertex. The process was continued until there were vertices that could be inserted. As a result, this led to a more diverse route than in the crossover.

In this article the idea of the path relinking in combination with the GA is developed. Let B denotes the fixed number of the GA generations. In first approach, called GAPR_B, the PR between the route with highest profit value from the population and a new generated route is performed every B generations. In the second method, called GAPR_ELO, the PR strategy is used when the GA solution is trapped in a local optimum. When the GA solution does not improve for A generations, PR between random routes is performed to escape from the local optimum. Let N_g denotes a number of generations, P_{size} describe the number of routes in the initial population and $best\_profit(gen)$ denotes the value of best total profit in the generation gen. The basic structure of GAPR_ELO is given as follows:

```
compute initial P_size routes; iter=0; not_imp=0;
while  iter< Ng do
    iter++;
    evaluate the fitness value of each route;
    make a tournament grouping selection;
    perform crossover;
    perform mutation;
```

```
    ------------------------------------------------------------
   | if best_profit(gen)= best_profit(gen-1) then not_imp++;  |
   |   else not_imp=0;                                        |
   |if not_imp > A then PR;                                   |
   ------------------------------------------------------------
  od
return the route with the highest profit value;
```

Structure of GAPR_B is similar to above but in the GAPR_B instead of instructions in the box there are the following:

```
if iter mod B ==0  then
    generate a new route R1;
    perform PR between R1 and route with the highest profit;
if no improvements in last 100 iterations then break;
```

4 Algorithms details

4.1 Initial population

Each route is coded as a sequence of vertices. An initial population of P_{size} routes is generated in the following way. First the route is initialized by the s and e vertices. Then the following values are assigned sequentially to the initialized vertices: $arrive_i$ – arrival time at vertex i, $wait_i$ – waiting time, if the arrival at the vertex i is before opening time, $start_i$ and end_i – starting and ending service time at vertex i. Moreover, the maximum time the service of a visit i can be delayed without making other visits infeasible is calculated for each location in the route as in [19]: $MaxShift_i = Min(C_i - start_i - T_i, wait_{i+1} + MaxShift_{i+1})$. Let l be the predecessor of vertex e in the route. In the subsequent steps a set of vertices is prepared. Each vertex v from this set is adjacent to vertex l and vertex e and will satisfy the following conditions after insertion: (a) $start_v$ and end_v are within the range $[O_v, C_v]$; (b) the locations after v could be visited in the route; and (c) the current travel length does not exceed the given t_{max} (including consumption time to insert the vertex v between l and e). A random vertex v is selected from this set. The values $arrive_v$, $wait_v$, $start_v$ and end_v are calculated and the vertex v is inserted. After the insertion, the values $arrive_e$, $wait_e$, $start_e$ and end_e are updated. Moreover, for each vertex in the tour (from vertex e to s) the $MaxShift$ value is updated as well. The tour generation is continued for as long as locations that have not been included are present and t_{max} is not exceeded.

4.2 Selection

Tournament grouping selection is used, which yields better adapted routes than standard tournament selection [9]. A set of P_{size} routes is divided into k groups and the tournaments are carried out sequentially in each of groups. t_{size} random routes are removed from the group, the route with the highest value for the fitness function $TotalProfit^3/TravelTime$ is copied to the next population, and the t_{size} previously chosen routes are returned to the old group. Selection from the group currently analysed has been repeated P_{size}/k times, P_{size}/k routes are chosen for a new population.

Finally, when this step has been repeated in each of the remaining groups, a new population is created, containing P_{size} routes.

4.3 Crossover

In the crossover stage, first two random routes are selected. Then we determine all vertices which could be replaced without exceeding the time window conditions and the t_{max} limit. We choose a set of vertices with similar time windows and start and end of service. If there are no similar vertices, crossover is terminated (no changes are applied). Otherwise, a random pair is selected from all similar pairs of vertices. This pair is a point of crossover. Two new routes are created by exchanging routes fragments (from the crossing point to the end of the route) from both parents. Next, for each vertex i from the crossing point to vertex e, the values for $arrival_i$, $wait_i$, $start_i$ and end_i are updated and the new $MaxShift$ values are calculated for each locations from vertex e to s.

4.4 Mutation

In mutation phase a random route is selected from P_{size} routes. Two types of mutation are possible – a gene insertion or gene removal (the probability of each is 0.5). The mutation process is repeated on the selected route N_m times, where N_m is the parameter. During the *insertion mutation*, all possibilities for inclusion of each new vertex (not present in the route) are considered. We check whether the shift resulting from the new insertion exceeds the constraints associated with the previously calculated $wait$ and $MaxShift$ values of the gene located directly after the newly inserted one. The location u with the highest value of $(p_u)^2/TravelTimeIncrease(u)$ is selected for insertion. $TravelTimeIncrease(u)$ is defined as the increased travel time after u is included. This value also takes into account the waiting and visiting time of vertex u. The selected gene u is inserted into the route and the values of $arrival_u$, $wait_u$, $start_u$ and end_u are calculated. For each location after u the arrival time, waiting time, and start and end of service are updated. Starting from the ending point, the $MaxShift$ value is updated for each location in the route.

In the *deletion mutation* we remove a randomly selected vertex (excluding the s and e) in order to shorten the travel length. After the vertex is removed, all locations after the removed vertex are shifted towards the beginning of the route. Furthermore, the locations before and after the removed vertex should be updated as in the insertion mutation.

4.5 Path relinking

Let R_1 and R_2 be the routes selected to the PR process. In GAPR_B route R_1 is a route with the highest profit value and R_2 is a new generated route. In GAPR_ELO R_1 as well R_2 are the random routes. $V_{R_1-R_2}$ denotes the set of vertices present in R_1 and not in R_2. $V_{R_2-R_1}$ denotes the set of vertices present in R_2 and not in R_1. During PR(R_1, R_2) we attempt to insert vertices from $V_{R_2-R_1}$ into R_1 in the best possible positions. The total consumption time associated with inserting a vertex j between vertex i and k is calculated as follows [19]: $Shift_j = t_{ij} + wait_j + T_j + t_{jk} - t_{ik}$. In addition, we check whether the shift resulting from the new insertion exceeds the constraints associated with the previously calculated $wait$ and $MaxShift$

values for the vertices located directly after the newly inserted one. If the shift exceeds the constraints the vertices from $V_{R_1-R_2}$ are removed to restore the possibility of inserting new locations. For each vertex u from this set a ratio is calculated as follows: $RemovalRatio = (p_u)^2/(end_u - arrive_u)$. After this computation the vertex with the smallest $RemovalRatio$ is removed. This removal is repeated until we can insert some vertices into the path. Finally the vertex u with the highest value for $InsertionRatio = (p_u)^2/Shift_u$ and not exceeded the mentioned constrains is selected for insertion. The insertion is performed in the same as in the insertion mutation. The process is repeated for as long as t_{max} is not exceeded and the set $V_{R_2-R_1}$ is not empty. In the GAPR_B the new route, being the result of PR, replace the route with the lowest fitness function from population. Additionally in the GAPR_ELO, PR(R_2, R_1) is performed by inserting vertices from $V_{R_1-R_2}$ into R_2. Two new routes are created as a result of PR(R_1, R_2) and PR(R_2, R_1). If the fitness values of the new routes are higher than the fitness value of R_1 and R_2, they replace them.
Figure 2 shows two routes R_1: 1-3-4-5-1 and R_2: 1-7-4-2-3-1 related to the graph in Figure 1. The set $V_{R_2-R_1}$ is $\{2, 7\}$. At the beginning of the PR(R_1, R_2) vertex 7 is inserted to the route R_1, because the vertex 2 does not meet all constraints. In the next step the vertex 5 which is presented in R_1 and not presented in R_2 is removed from R_1 and the route is 1-7-3-4-1. It is the final result of PR(R_1, R_2).

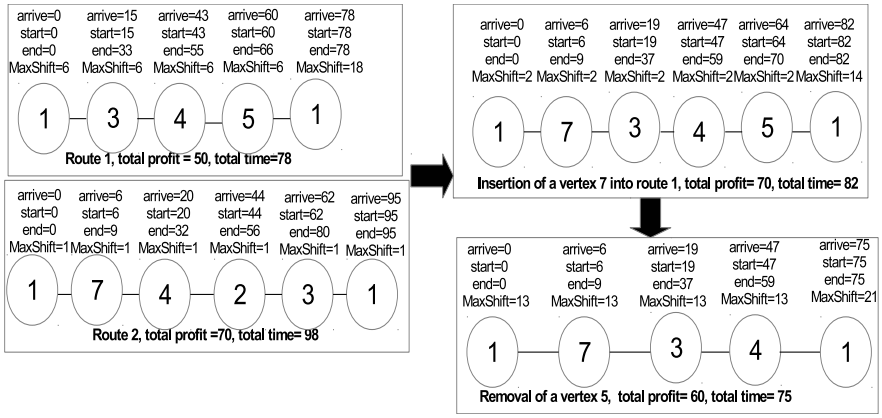


Fig. 2. Example of the insertion and removal in path relinking process

5 Experimental Results

The GAPR_B and GAPR_ELO were coded in C++ and run on an Intel Core i7, 1.73 GHz CPU (turbo boost to 2.93 GHz). The algorithms were tested on Solomon [15] and Cordeau [3] test instances for the OPTW. The number of vertices in the Solomon instances is equal to 100 and different layouts for the vertices are considered: cluster (c), random (r) and random-clustered (rc). The Solomon benchmarks $c\backslash r\backslash rc200$, have the

same coordinates of vertices, profits and visiting times, but they have approximately three times higher values of t_{max} and larger time windows than the $c\backslash r\backslash rc100$ instances. The Cordeau instances vary between 48 and 288 vertices.
The parameter values of GAPR_B and GAPR_ELO which represent the best trade-off between the quality of the solutions and the computational time are determined by tests: P_{size} =150 (initial size of population), t_{size}=3 (number of routes selected from a group in selection), k=15 (number of groups in selection), N_g=500 (maximum number of iteration) and N_m=15 (number of mutations repeated on the selected route). In GAPR_ELO the parameter A is set to 100. Moreover, during the testing of GAPR_B we consider different B =2, 5, 10, 20. The differences between solutions for different values of B are about 1%. We decide to use B=10 for the final comparison.

The solutions of GAPR_B and GAPR_ELO (and the previous version of GA [9] and GAPR [10]) are obtained by performing sixteen runs simultaneously on each benchmark instance (two runs each on the eight processor cores). The OpenMP technology is used for simultaneously tests. Tables 1 – 3 present detailed results of experiments: the average profit and time (in seconds). Moreover they also show the percentage gap between the best know solution values (BK) [11]. The values of BK are given in italic if the optimal value is known. An empty cell denotes a gap equal to 0. Tests for which our solutions improves the best known values are given in bold.

The results presented in Tables 1 – 3 indicate that combining PR and genetic algorithm outperforms solutions of the GA. The average gap between GA and GAPR, GA and GAPR_B, GA and GAPR_ELO is 3.3%, 3.4%, 2.6% respectively. The application of PR instead crossover and the GAPR_B gives similar profit results and the execution time. The GAPR_B gives the best results for benchmark instances with large time windows and large values of t_{max} e.g. the groups r200 and rc200. For $c \setminus r \setminus rc100$ instances (where time windows is narrow) the profit results of GAPR_B and GAPR_ELO are comparable. Using PR when the algorithm trapped in local optimum causes the increase of GAPR_ELO execution time (on average three times longer in $c \setminus r \setminus rc100$, two times in $c \setminus r \setminus rc200$ and pr1-20 classes in comparison to the other methods). Moreover PR causes that the solution space is extended to better results for p11, p13, p15, p17 than the best know values so far.

6 Conclusions and Further Work

In this article two new methods using path relinking with genetic algorithm have been investigated: GAPR_B has introduced the PR every B generations of the genetic algorithm and GAPR_ELO has applied PR when the genetic algorithm trapped in a local optimum. Computer experiments has shown that the proposed methods gives better solutions in comparisons to the previous GA and similar to GAPR solutions (the path relinikng instead of the crossover). To the best our knowledge the other GAs for the considered version of OPTW has never been proposed in the literature. It is worth mentioning that GAPR has outperformed other every fast and effective heuristics e.g. iterated local search [10]. The proposed method can be tailored to solve realistic problem of planning the most valuable routes for tourism (co-called tourist trip design problem [18]). In our future research we intend to add some algorithm improvements and to conduct experiments on the realistic database of points of interests in our city.

Table 1. Results for Solomon's test problems (n=100).

	BK	GA		GAPR		GAPR_B		GAPR_ELO			% gap BK with			
name	score	avg	time	avg	time	avg	time	avg	time		GA	GAPR	GAPR_B	GAPR_ELO
c101	*320*	317	0.2	320	0.2	320	0.2	320	0.6					
c102	*360*	360	0.3	360	0.2	360	0.2	360	0.7					
c103	*400*	389	0.3	400	0.3	390	0.3	400	0.8		2.8		2.5	
c104	*420*	403	0.3	420	0.3	410	0.2	400	0.8		4.2		2.4	4.6
c105	*340*	340	0.2	340	0.2	340	0.2	340	0.6					
106	*340*	340	0.2	340	0.2	340	0.2	340	0.6					
c107	*370*	362	0.2	360	0.2	360	0.2	370	0.7		2.2	2.7	2.7	
c108	*370*	370	0.2	370	0.3	370	0.2	370	0.7					
c109	*380*	380	0.2	380	0.3	380	0.2	380	0.8					
sum:	**3300**	**3260**	**2.1**	**3290**	**2.2**	**3270**	**2.0**	**3280.7**	**6.2**	**avg.:**	**1.2**	**0.3**	**0.9**	**0.6**
r101	*198*	189	0.2	197	0.2	198	0.2	193	0.5		4.8	0.5		2.5
r102	*286*	286	0.3	286	0.2	286	0.2	286	0.6		0.1			
r103	*293*	290	0.2	293	0.3	293	0.3	288	0.7		1.1			1.7
r104	*303*	297	0.3	303	0.3	297	0.2	298	0.8		2.0		2.0	1.7
r105	*247*	244	0.2	247	0.2	247	0.2	247	0.6		1.2			
r106	*293*	292	0.2	293	0.2	293	0.2	293	0.7		0.3			
r107	*299*	292	0.2	299	0.3	299	0.3	297	0.7		2.3			0.7
r108	*308*	300	0.3	308	0.4	304	0.4	308	0.8		2.5		1.4	
r109	*277*	270	0.2	272	0.2	270	0.2	270	0.6		2.5	1.7	2.4	2.5
r110	*284*	277	0.3	283	0.3	281	0.3	281	0.7		2.4	0.4	1.1	1.1
r111	*297*	293	0.4	297	0.3	297	0.2	295	0.7		1.3			0.7
r112	*298*	293	0.2	292	0.2	298	0.2	295	0.7		1.6	2.0		1.0
sum.	**3383**	**3323**	**3.1**	**3370**	**3.0**	**3364**	**3.0**	**3351**	**8.1**	**avg.:**	**1.8**	**0.4**	**0.6**	**0.9**
rc101	*219*	216	0.2	216	0.2	219	0.2	217	0.6		1.4	1.4		0.7
rc102	*266*	261	0.2	266	0.3	266	0.2	266	0.6		1.9		1.4	
rc103	*266*	262	0.2	265	0.3	265	0.2	265	0.6		1.5	0.5	0.4	0.4
rc104	*301*	294	0.2	301	0.2	301	0.2	301	0.7		2.3			
rc105	*244*	236	0.2	241	0.2	239.3	0.2	241	0.6		3.3	1.2	1.9	1.2
rc106	*252*	245	0.2	242	0.2	250	0.2	250	0.6		2.8	3.8	0.8	0.8
rc107	*277*	269	0.2	258	0.2	275.6	0.2	274	0.7		2.9	6.9	0.5	1.1
rc108	*298*	298	0.2	298	0.2	298	0.2	298	0.7		3.0			
sum.	**2123**	**2072**	**1.6**	**2087**	**1.8**	**2113.9**	**1.6**	**2112**	**4.9**	**avg.:**	**2.4**	**1.7**	**0.4**	**0.5**

Table 2. Results for Solomon's test problems, cont. (n=100).

	BK	GA		GAPR		GAPR_B		GAPR_ELO			% gap BK with			
name	score	avg	time	avg	time	avg	time	avg	time		GA	GAPR	GAPR_B	GAPR_ELO
c201	870	848	0.4	860	0.4	860	0.5	860	1.2		2.5	1.1	1.1	1.1
c202	930	901	0.7	920	0.5	922	0.7	908	1.4		3.1	1.1	0.9	2.4
c203	960	931	0.7	957	0.7	960	0.7	960	1.6		3.0			
c204	980	952	0.7	968	0.8	968.7	0.8	970	1.8		2.9	1.3	1.2	1.0
c205	910	893	0.4	900	0.4	900	0.5	900	1.3		1.9	1.1	1.1	1.1
c206	930	905	0.5	919	0.5	916	0.6	912.7	1.4		2.7	1.2	1.5	1.9
c207	930	910	0.4	930	0.7	920	0.5	930	1.4		2.2		1.1	
c208	950	931	0.5	940	0.5	947.3	0.6	950	1.4		2.0	1.1	0.3	1.4
sum:	**7460**	**7271**	**4.3**	**7394**	**4.5**	**7394**	**4.8**	**7390.7**	**11.5**	**avg.:**	**2.5**	**0.9**	**0.9**	**0.9**
r201	797	760	0.7	776	0.6	783.6	0.8	785	1.5		4.6	2.6	1.7	1.5
r202	929	867	0.9	885	1.1	901.7	0.9	883.2	1.8		6.7	4.8	2.9	4.9
r203	1021	954	1.3	969	1.2	979,1	0.9	992	2.0		6.6	5.1	4.1	2.8
r204	1086	1018	1.3	1031	1.0	1048,3	1.0	1059	2.2		6.4	5.1	3.5	2.5
r205	953	876	0.9	925	1.3	895,3	1.0	908.9	1.8		8.1	3.0	6.1	4.6
r206	1029	954	1.0	1007	1.1	982	0.9	980.7	1.9		7.4	2.1	4.6	4.7
r207	1072	986	0.9	1041	1.3	1035	0.9	986.5	2.1		8.0	2.9	3.5	8.0
r208	1112	1042	1.1	1074	1.4	1095	1.5	1077.4	2.2		6.3	3.4	1.5	3.1
r209	950	897	1.1	927	1.0	932	1.4	920	1.9		5.6	2.4	1.9	3.2
r210	987	915	1.2	944	1.0	976.3	0.9	943.6	1.9		7.3	4.4	1.1	4.4
r211	1046	967	1.2	1003	0.8	990.0	0.9	1008.7	2.1		7.5	4.2	5.4	3.6
sum:	**10982**	**10235**	**11.6**	**10581**	**11.9**	**10618.7**	**11.1**	**10545.1**	**21.3**	**avg.:**	**6.8**	**3.7**	**3.3**	**4.0**
rc201	795	768	0.5	789	0.6	784.3	0.9	788.8	1.3		3.4	0.7	0.1	0.8
rc202	936	866	0.7	919	0.7	900	0.7	921.7	1.5		7.5	1.8	3.8	1.5
rc203	1003	866	0.7	956	1.0	975.7	1.1	959.8	1.7		7.1	4.7	2.7	4.3
rc204	1136	1044	1.2	1123	0.9	1080.9	1.0	1091	2.0		8.1	1.1	4.9	4.0
rc205	859	808	0.5	842	0.7	845.9	0.7	822.8	1.4		5.9	2.0	1.5	4.2
rc206	895	850	0.6	859	0.7	884.3	0.8	864.5	1.5		5.0	4.0	1.2	3.4
rc207	983	896	0.6	948	0.9	973,2	1.1	918.5	1.7		8.9	3.6	1.0	6.6
rc208	1053	976	0.9	1005	0.6	1037	0.6	951.2	1.8		7.3	4.6	1.5	9.7
sum:	**7660**	**7140**	**5.6**	**7442**	**6.2**	**7491.3**	**6.8**	**7318.3**	**12.9**	**avg.:**	**6.8**	**2.9**	**2.2**	**4.5**

Table 3. Results for Cordeau's test problems (n from 48 to 288).

	BK	GA		GAPR		GAPR_B		GAPR_ELO			% gap BK with			
name	score	avg	time	avg	time	avg	time	avg	time		GA	GAPR	GAPR_B	GAPR_ELO
48 pr1	*308*	298	0.2	305	0.2	307.8	0.2	308	0.6		3.2	1.0	0.1	
96 pr2	*404*	381	0.4	400	0.4	398.6	0.5	395	2.2		5.7	1.1	1.3	2.2
144 pr3	*394*	373	0.6	393	0.4	383.8	0.7	391.3	1.2		5.3	0.3	2.6	0.7
192 pr4	*489*	449	0.6	487	0.6	470	0.8	459.9	1.7		8.2	0.4	3.9	6.0
240 pr5	*595*	553	1.9	559	2.0	563.2	1.2	582	2.7		7.1	6.1	5.3	2.2
288 pr6	*590*	512	1.0	573	1.6	559	1.8	553.9	2.7		13.2	2.9	5.3	6.1
72 pr7	*298*	282	0.2	288	0.2	289.5	0.3	288	0.7		5.4	3.4	2.8	3.4
144 pr8	*463*	426	0.5	463	0.4	446.0	0.4	452.9	1.3		8.0		3.7	2.2
216 pr9	*493*	448	1.0	451	0.9	464	1.0	440.5	2.0		9.1	8.5	5.9	10.6
288 pr10	594	536	1.1	556	1.8	594	1.8	558	3.0		9.8	6.3		6.1
sum:	**4628**	**4258**	**7.6**	**4475**	**8.6**	**4475.9**	**8.9**	**4429.6**	**16.8**	**avg.:**	**8.0**	**3.3**	**3.3**	**4.3**
48 pr11	330	**332**	0.3	**344**	0.3	**346.0**	0.3	**348**	0.7		-0.6	-4.3	-4.8	-5.5
96 pr12	442	417	0.5	432	0.4	434.5	0.4	431.6	1.0		5.7	2.3	1.8	2.4
144 pr13	461	429	0.6	455	0.9	**464.6**	1.0	446.2	1.5		6.9	1.3	-0.7	3.2
192 pr14	567	489	0.9	533	1.5	531.6	0.9	518	1.0		13.8	5.9	6.2	8.6
240 pr15	685	611	1.8	**700**	2.4	664.5	1.3	662.9	3.0		10.8	-2.1	3.0	3.2
288 pr16	674	564	1.7	607	1.5	603.7	1.7	614.2	3.2		16.3	10.0	10.4	8.9
72 pr17	359	349	0.3	**360**	0.3	356	0.3	353	0.8		2.8	-0.4	0.8	1.7
144 pr18	535	468	0.9	528	0.7	528	0.6	523	1.5		12.5	1.3	1.3	2.2
216 pr19	562	501	1.0	533	1.3	527	1.2	507.9	2.2		10.9	5.1	6.2	9.6
288 pr20	667	590	1.8	611	2.1	623.6	2.3	618	1.5		11.5	8.3	6.5	7.3
sum:	**5282**	**4750**	**9.7**	**5103**	**11.5**	**5078.8**	**9.9**	**5022.8**	**16.2**	**avg.:**	**10.1**	**3.4**	**3.8**	**4.9**

Acknowledgements

The authors gratefully acknowledge support from the Polish Ministry of Science and Higher Education at the Bialystok University of Technology (grant S/WI/1/2014 and W/WI/2/2013).

References

1. Archetti, C., Hertz, A., Speranza, M.G.: Metaheuristics for the team orienteering problem. Journal of Heuristics, 13, 49–76 (2007)
2. Campos, V., Marti, R., Sanchez-Oro, J., Duarte, A.: Grasp with Path Relinking for the Orienteering Problem, Journal of the Operational Research Society (2013)
3. Cordeau, J.F., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems, Networks, 30(2), 105–119 (1997)
4. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers. Boston (1997)
5. Huang, Y. H., Ting, C. K.: Genetic algorithm with Path Relinking for the Multi-Vehicle Selective Pickup and Delivery problem, Evolutionary Computation (CEC), IEEE Congress. 1818–1825 (2011)
6. Jia, S., Hu, Z.H.: Path-relinking Tabu search for the multi-objective flexible job shop scheduling problem. Computers and Operations Research, 47, 11–26 (2014)
7. Kantor, M., Rosenwein, M .: The Orienteering Problem with Time Windows, Journal of the Operational Research Society, 43, 629–635 (1992)
8. Karbowska-Chilinska, J., Zabielski, P.: A Genetic Algorithm vs. Local Search Methods for Solving the Orienteering Problem in Large Networks. Knowledge Engineering, Machine Learning and Lattice Computing with Applications, Lecture Notes in Computer Science. 7828, 11–20 (2013)
9. Karbowska-Chilinska, J., Zabielski, P.: A Genetic Algorithm Solving Orienteering Problem with Time Windows. Advances in Systems Science. Advances in Intelligent Systems and Computing, Springer. 240, 609–619 (2014)
10. Karbowska-Chilinska, J., Zabielski, P.: Genetic Algorithm with Path Relinking for the Orienteering Problem with Time Windows. Concurrency, Specification and Programming: Proceedings of the 22nd International Workshop on Concurrency, Specification and Programming CS&P'2013, 245–258 (2013)
11. Labadie, N., Mansini, R., Melechovsky, J., Wolfler Calvo, R.: The Team Orienteering Problem with Time Windows: An LP-based Granular Variable Neighborhood Search. European Journal of Operational Research. 220(1), 15–27 (2012)
12. Montemanni, R., Gambardella, L.M.: Ant colony system for team orienteering problems with time windows. Foundations of Computing and Decision Sciences. 34, (2009)
13. Oliveira, P.L, Arroyo, J.E.C, Santos, A.G., Goncalves, L.B., Oliveira, A.P.: An evolutionary algorithm with path-relinking for the parallel machine with job splitting, Systems, Man, and Cybernetics, IEEE International Conference, 3153–3158 (2012)
14. Perez, M.P., Rodriguez, F.A., Moreno-Vega, J.M. :A hybrid VNS-path relinking for the p-hub median problem. Journal Management Mathematics. 18(2), 157–171 (2007)
15. Solomon, M.: Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research 35(2) 254-265 (1987)
16. Tang, H., Miller-Hooks, E.: A tabu search heuristic for the team orienteering problem. Computers & Operational Research. 32(6), 1379–1407 (2005)

17. Tsiligirides, T.: Heuristic Methods Applied to Orienteering. Journal of the Operational Research Society. 35(9), 797–809 (1984)
18. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: The City Trip Planner: An expert system for tourists. Expert Systems with Applications. 38(6), 6540–6546 (2011)
19. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: Iterated local search for the team orienteering problem with time windows. Computers O.R. 36, 3281–3290 (2009)

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Comparison of Different Graph Weights Representations Used to Solve the Time-Dependent Orienteering Problem

Krzysztof Ostrowski

Faculty of Computer Science
Białystok University of Technology, Poland
http://www.wi.pb.edu.pl

Abstract. The paper compares three types of network weights representations applied to solve the Time-Dependent Orienteering Problem (TDOP). First variant uses real, time-dependent weights, second variant uses mean weights and the third is a hybrid of the previous two variants. A randomized, local search algorithm was used in this comparison and tests were conducted on real public transport network of Białystok. The results show importance of both detail and general properties of network topology when solving TDOP.

Keywords: time-dependent orienteering problem, public transport network, mean weights, time-dependent weights, hybrid weights, comparison

1 Introduction

The Time-Dependent Orienteering Problem (TDOP) is a variant of well known Orienteering Problem (OP). The problem name comes from the sport game of orienteering. The competitors start and end the game at a specific control point. The task is to collect as much points as possible at given checkpoints and return to the control point within a given period of time. The problem has many practical applications including tourist trip planning [14], [21] and logistics. In classic OP distances (weights) between points are constant whereas in TDOP they vary with time.

Let $G(V, E)$ be a directed, weighted graph where V is the set of vertices and E is the set of time-dependent arcs ($|V| = n$, $|E| = m$). Each vertex i has a non-negative profit p_i and a non-negative visiting time u_i for $i = 1, ..., n$. Each edge $(i,\ j)$ has associated time-dependent weights in a form of function $w_{ij}(t)$ – it represents time of travel from vertex i to vertex j which starts at time t $(i, j = 1, ..., n)$. Given time constraint t_{max}, travel start time t_0 and a pair of vertices $(s,\ e)$ the goal of the TDOP is to determine a path from s to e which starts at time t_0, visits a subset of vertices from V and maximizes total collected profit. In addition its total cost (travel time plus vertex visiting time) cannot exceed t_{max}. Each vertex on this path can be visited only once. TDOP can be defined as a linear integer problem. Let $x_{ij}(t) = 1$ if a path include direct travel from vertex i to vertex j which starts at time t $(i, j = 1, ..., n)$. Otherwise it is equal to zero. Let r_i be a position of a vertex in a path $(i = 1, ..., n)$. This variable is assigned only to vertices visited during the tour. In mathematical terms the goal is to maximize total profit (1) given constraints (2-8):

$$max \sum_{i \in V} \sum_{j \in V} \sum_{t \in T} (p_i \cdot x_{ij}(t)) + p_e \tag{1}$$

$$\sum_{j \in V} \sum_{t \in T} x_{sj}(t) = \sum_{i \in V} \sum_{t \in T} x_{ie}(t) = 1 \tag{2}$$

$$\sum_{j \in V} x_{xj}(t_0 + u_s) = 1 \tag{3}$$

$$\underset{k \in V \setminus \{s,e\}}{\forall} (\sum_{i \in V} \sum_{t \in T} x_{ik}(t) = \sum_{j \in V} \sum_{t \in T} x_{kj}(t) \leqslant 1) \tag{4}$$

$$\underset{i \in V,\ k \in V \setminus \{s,e\},\ t \in T}{\forall} (x_{ik}(t) = 1 \Rightarrow \underset{j}{\exists} (x_{kj}(t + u_k) = 1)) \tag{5}$$

$$\sum_{i \in V} \sum_{j \in V} \sum_{t \in T} ((w_{ij}(t) + u_i) \cdot x_{ij}(t)) + u_e \leqslant t_{max} \tag{6}$$

$$r_s = 1 \tag{7}$$

$$\underset{i \in V,\ j \in V,\ t \in T}{\forall} (x_{ij}(t) = 1 \Rightarrow r_j = r_i + 1) \tag{8}$$

Constraint 2 forces the path to start in vertex s and end in vertex e while constraint 3 means that travel starts at time t_0. Constraint 4 guarantee that every vertex is visited at most once. Constraint 5 states that visit in any vertex k lasts exactly u_k. Constraint 6 forces the travel time not to exceed t_{max}. Constraints 7 and 8 eliminates any subtours from the path.

2 Literature review

Many different techniques were used to deal with the OP. It is an NP-hard problem [7] and its solution space grows very fast with data size increase – therefore most approaches were based on approximate methods. However, exact solutions (branch and cut) were also introduced [3, 6, 12] for networks up to 400 nodes.

One of the first approximate approaches to OP was presented by Tsiligirides [19] – his algorithm is based on Monte Carlo method. Many researchers concentrated on local search methods with greedy procedures of path improving. Golden et al. [7] introduced route construction which uses center of gravity and 2-opt path improvement phase. Chao et al. [2] presented a greedy heuristic exploiting another geometrical feature – search space was limited to vertices lying in an ellipse (t_{max} is a major axis length). Vansteenwegen et al. [20] developed a guided local search (GLS) method which uses improvement procedures (2-opt, insert, replace), center of gravity and penalty function for vertices. Greedy Randomized Adaptive Search Procedure with Path relinking (GRASPwPR) was presented [15] to solve Team Orienteering Problem (TOP) and later it was adopted to solve the OP [1]. It uses greedy-random path construction and improvement. Afterwards path relinking is performed for all pairs of solutions it gradually changes one path into another. It gives very satisfactory results for benchmark instances.

Besides local search methods other successful approaches were introduced. Tasgetiren et al. [17] introduced the first genetic algorithm for OP. He used tournament

selection, crossover and mutation which based on local search procedures (i. e. add, replace, swap). Most of researchers concentrated on small and medium size graphs (32-400 nodes in benchmark networks). Evolutionary algorithms gave satisfactory results also for larger instances [8–10]. Ant-colony optimization [13] and tabu search [16] were also among methods successfully used to deal with the OP.

While there were several approaches to the OP few works concentrated on TDOP related problems. Li [11] introduced dynamic programming algorithm which finds exact solutions for small instances. It solved Time-Dependent Team Orienteering Problem (TDTOP) in which there can be more than one path in solution. Garcia et al. [4] was first to apply and solve the problem for public transport network. The algorithm was tested in San Sebastian. Additional option was inclusion of time windows – intervals in which vertices could be visited (TDTOPTW problem). However the network was small and the algorithm (Iterated Local Search – ILS) used static weights – computed as an average of all daily time-dependent weights. Repair procedure was needed to shorten the result paths (if they exceed tmax). Their second method based on time-dependent weights but with assumptions that public transport service is periodic. However, in real network frequencies of bus arrivals can change often during a day. Recently the first approach which uses real time-dependent weights in public transport network was presented [5]. Time discretization of 1 minute was applied – it gives 1440 states in a day. Tests of two fast heuristics were conducted on network of Athens.

3 TDOP in public transport network

3.1 TDOP model

A good example of time-dependent graph is public transport network where travel times between given points depend on bus/train/tram timetable. TDOP can be easily modelled in such a network – for example in tourism when visiting some points of interests (POI) and using public transport to travel between these points. Here are assumptions I made when applying TDOP problem to public transport and POI network:

- Time is discrete and represent every minute of a day – arithmetic modulo 1440 is used in time calculations. Minute discretization is consistent with bus schedules. For a time-dependent network with n POIs there are $1440 \cdot n^2$ weights.
- Each weight $w_{ij}(t)$ is calculated as the shortest multimodal travel time from POI i to POI j starting at time t.
- In a multimodal path both walk links and public transport links are allowed. Walk links are allowed between POIs, between a POI and a bus stop and between bus stops (as a transfer). However, they have limited lenght.
- Number of bus transfers when travelling between POIs is limited to a fixed number. It is rarely greater than 2-3 in a medium-size city. However it depends on public transport network topology.

Public transport network can be expressed mathematically. Let $G(V, E)$ be a directed, weighted graph where V is the set of vertices and E is the set of arcs ($V = V_1 \cup V_2$, $E = E_1 \cup E_2$). Set of vertices is divided into POIs (V_1) and bus stops (V_2). Set of edges is divided into walk links (E_1) and bus links (E_2). Here is formula for walk link time (in minutes):

$$W_{ij} = 60 \cdot \frac{6378.137 \cdot acos(asin(la_i) \cdot sin(la_j) + cos(la_i) \cdot cos(la_j) \cdot cos(lo_i - lo_j))}{S}$$

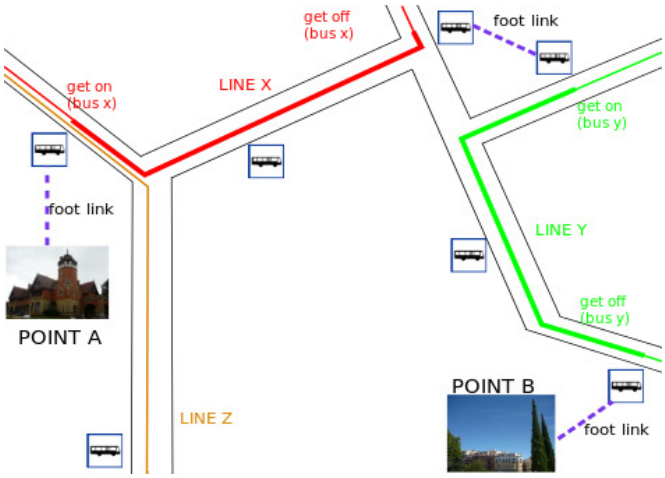


Fig. 1. An exemplary 1-transfer path between two points. It consists of: walk link from point A to bus stop, transport link (line x), walk link between bus stops (transfer), transport link (line y) and walk link from bus stop to point B.

Where i, j are any vertices, S is a walker speed (in km/h) and la_i, la_j, lo_i, lo_j are latitude and longitude of vertices given in radians. Walk time is rounded to the nearest integer. If it is longer than a given walk limit then W_{ij} is infinity.

When there is a direct bus course between any ordered pair of bus stops (i, j) we say that there is a connection from i to j. Between any pair of bus stops there can be a lot of connections during a day (associated with many bus lines and bus courses). Each bus course is a set of connections and public transport schedule is a set of bus courses. Let C be a set of all bus connections. Any connection $c \in C$ has the following properties:

- Starting bus stop ($start(c)$)
- Ending bus stop ($end(c)$)
- Start time ($stime(c)$)
- Travel time ($ttime(c)$)

Bus link time $B_{ij}(t)$ between any ordered pair of bus stops (i, j) at time t is the shortest time needed to get by a direct bus from i to j assuming that arrival time at bus i is t. In order to calculate $B_{ij}(t)$ all direct bus connections from i to j should be considered. Bus link time takes into account both waiting time and time spent in a bus. Thus, $wait(t_1, t_2)$ should be introduced, which is waiting time for a bus arriving at t_2 assuming that a person arrived at a bus stop at time t_1:

$$wait(t_1, t_2) = \begin{cases} t_2 - t_1 & if\ t_2 \geqslant t_1 \\ t_2 - t_1 + 1440 & if\ t_2 < t_1 \end{cases}$$

The above formula comes from the fact that bus arrival times are given in minutes (interval 0-1439) and time arithmetic is performed modulo 1440. Now bus link time $B_{ij}(t)$ can be computed as follows:

$$B_{ij}(t) = min\{T : \exists_{c \in C} (start(c) = i \wedge end(c) = j \wedge T = wait(t, stime(c)) + ttime(c))\}$$

Bus link times are given in minutes (according to bus schedules). In practise the best option (from options above) is usually the first connection (with shortest wait time). If there are no direct connections from i to j then $B_{ij}(t)$ is infinity.

Time-dependent weights between POIs can be computed based on walk links and bus links. The shortest multimodal path from POI i to POI j at time t (weight $w_{ij}(t)$) consisting of at most k bus links is a direct walk link or the shortest (in terms of time) from all paths in the form:

$$POI_i \rightarrow ST_{11} \Rightarrow ST_{12} \rightarrow ST_{21} \Rightarrow ST_{22} \rightarrow ... \rightarrow ST_{z1} \Rightarrow ST_{z2} \rightarrow POI_j$$

Where ST_{ij} is a symbol of a bus stop, $\rightarrow$ is a symbol of a walk link and $\Rightarrow$ is a symbol of a bus link. Number of bus links (z) is not greater than k.

3.2 Precomputation of weights

Before TDOP algorithm is executed, time-dependent weights should be computed. Once determined they can be used in multiple TDOP algorithm executions. Weights are stored in RAM and can be retrieved by the algorithm in O(1) time. It prevents from additional CPU time consumptions associated with calculating shortest multimodal paths between given POIs. Weights precomputation is performed by effective, dynamic programming algorithm. This process in not time consuming – it can determine all $1440n^2$ time-dependent weights for the city of Białystok (141 POIs, 693 bus stops, 37 bus lines) in 5 seconds (Intel Core i7 CPU).

4 TDOP local search algorithm

Once time-dependent weights are determined TDOP algorithm can be executed. A randomized, local search method was applied to deal with the problem. It is based on a random-start, multi-stage hill climbing procedure. After generation and improvement of many paths the algorithm chooses the best one. The local search method was chosen as it can be easily adopted to all weight representation variants and gives satisfactory results within short execution time. Here is the algorithm pseudocode:

```
iteration=0;
resultPath=emptyPath();
while iteration < N do
begin
    currentPath=randomInitialization();
    shorteningPhase(currentPath);
    p=p0;
    while p>0 do
    begin
        improvement=true;
        localOptimum=currentPath;
        while improvement do
        begin
            removalPhase(currentPath);
            insertionPhase(currentPath);
```

```
            if profit(currentPath)>profit(localOptimum) then
            begin
                localOptimum=currentPath;
                improvement=true;
            end
            else
            begin
                currentPath=localOptimum;
                improvement=false;
            end
        end
        p-=dP;
    end
    if profit(currentPath)>profit(resultPath) then
        resultPath=currentPath;
    iteration++;
end
```

Algorithm parameters:
N – number of constructed paths
p – percentage of vertices removed from a path during removal phase
$p0$ – initial percentage p
dP – decrease of percentage p

4.1 Initialization

First, a random path is generated. Starting from vertex s and ending in vertex e the algorithm adds to the path successive, random nodes as long as it is possible without exceeding t_{max} constraint. When number of generated paths is large enough it enables for good exploration of the solution space.

4.2 Path shortening

After initialization the algorithm performs operations which shorten a path without changing its subset of vertices. At this stage operations of *2-opt* and *move* are applied multiply. *2-opt* method changes edge arrangement in order to shorten the path. It checks all pairs of non-adjacent edges ($A \to B$ and $C \to D$) and tries to replace them with new pair of edges ($A \to C$ and $B \to D$) – new edges are formed by vertices exchange. The method choose the modification which shortens the path most. *move* method modifies vertices order. It takes into account all vertices (except s and e) and tries to move them to any other position in a path. From all these options the method chooses the one which shortens the path as much as possible.

4.3 Profit improvement phase

First I init variable p, which is percentage of the path vertices to be removed from the path during removal phase. During removing vertices with smallest profits are chosen. Afterwards insertion phase is executed – it is a hill climbing procedure which adds

new vertices to the path until no more addition is possible without violating t_{max} constraint. During insertion all options are considered (any new vertex can be inserted in any place in the path) and the best of them is chosen according to highest values of heuristic $h = \frac{profitGain^2}{costIncrease}$. Removal and insertion phases are repeated until there is no further improvement of profit. This procedure is executed multiple times for decreasing values of percentage p – changes are smaller as the path is improved.

5 Algorithm variants

Three algorithm variants were tested. The main difference between them was network weights representation.

- Time-dependent (TD) weights version. The local search algorithm uses time-dependent weights $w_{ij}(t)$ during computations.
- Mean weights version. For each pair of vertices one weight w_{ij} is computed as a mean of all 1440 time-dependent weights in a day. The algorithm uses mean weights during computations. After its execution each generated path is checked – its real time-dependent cost is different from the cost obtained using mean weights. If the real cost exceeds t_{max} constraint then the path is shortened according to greedy removal heuristic. Otherwise the path is improved until no more vertices can be added.
- Hybrid version. First the algorithm is executed on mean weights and generated paths are repaired in the same way as in previous version. Afterwards another local search procedure is executed for each path – this time the procedure uses real time-dependent weights. In this version both general network properties (mean weights) and network details (time-dependent weights) can be taken into account.

6 Experiment

The experiment was carried out on a public transport and POI network of BiaĹ‚ystok. The public transport network consists of 693 bus stops and 37 bus lines. The POI network has 141 vertices. During precomputation of shortest paths maximal number of transfers was set to 5 and maximal length of walk links was set to 0.3 km (assuming straight line distance between POIs and average speed of 3 km/h). Fig. 2 shows structure of shortest paths. Shortest paths consisting of more than 3 bus transfers (4 transportation links) are very rare. In fact, most connections include not more than 1 transfer. Due to short walk link, a few POIs are not connected to the public transport network. For the same reason number of direct walk connections is small.

Profits of vertices are random, integer numbers from interval [1, 100] and visit times are random integer numbers from interval [1, 30]. For start point (s) and end point (e) values of profit and visit time were set to 0. In [18] there is the precomputated, time-dependent POI network of Białystok. After description of all 141 POIs (id, label, profit, visit time, latitude and longitude) there are 1440 square matrices (each 141×141) of distances between POIs. The i-th row and the j-th column of the k-th matrix represents weight $w_{ij}(k-1)$. Number 65535 means there is no connection.

Experiment was conducted on Intel Core i7 CPU. Three different algorithm variants were tested. For smallest t_{max} value optimal results (OPT) were also computed. Experiments were conducted for 4 different t_{max} values (180, 360, 720 and 1440 minutes), 4 different t_0 values (0, 360, 720, 1080) and 3 different pairs of start and end

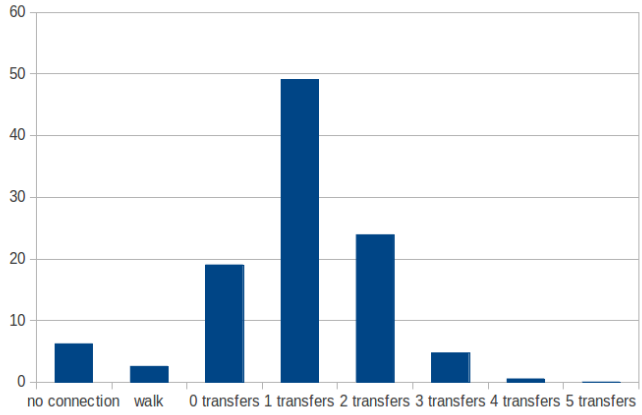


Fig. 2. Occurrence frequency (%) of various forms of shortest paths.

Table 1. Results and execution time comparison ($s = e = 34$)

$\mathbf{t_{max}}$	$\mathbf{t_0}$	TD weights	mean weights	hybrid weights	OPT
180	**0**	793	793	793	831
	360	909	853	879	942
	720	933	858	919	942
	1080	931	858	900	938
	avg. profit	**891.5**	**840.5**	**872.8**	**913.2**
	avg. time	**0.8**	**1**	**1.5**	**11600**
360	**0**	1403	1375	1466	-
	360	1508	1440	1509	-
	720	1500	1507	1550	-
	1080	1494	1486	1505	-
	avg. profit	**1476.3**	**1452**	**1507.5**	**-**
	avg. time	**1.5**	**1.7**	**2.3**	**-**
720	**0**	2449	2495	2454	-
	360	2379	2473	2493	-
	720	2382	2457	2486	-
	1080	2318	2377	2398	-
	avg. profit	**2382**	**2450.5**	**2457.8**	**-**
	avg. time	**3**	**3**	**5.2**	**-**
1440	**0**	3747	3924	3971	-
	360	3751	3894	3912	-
	720	3726	3909	3960	-
	1080	3686	3896	3956	-
	avg. profit	**3727.5**	**3905.8**	**3949.8**	**-**
	avg. time	**8.4**	**8**	**13.1**	**-**

points ($s = e = 34$, $s = 6$ and $e = 91$, $s = 1$ and $e = 51$). Start and and points were chosen according to their location: in the first case ($s = e = 34$) paths start and end in the city center, in the second case ($s = 6$, $e = 91$) paths start in the northern part of the city and end in its southern part and in the third case ($s = 1$, $e = 51$) paths start in the western part of the city and end in its eastern part. Local search algorithm parameters were: N=300, $p0$=30, dP=10. Execution time of algorithms was given in seconds.

In tab. 1 there is a comparison between algorithm variants when both start and end points of paths are 34. For shortest paths (t_{max}=180) time-dependent weights variant gives the best average result while mean weights version is over 5 percent worse. It shows the advantage of detail analysis (TD weights) over general analysis (mean weights) for shorter paths. TD weights variant is on average 2.4 percent worse than the optimal solution. Meanwhile hybrid weights version in terms of results is between other versions with 2 percent loss to TD weights version.

Table 2. Results and execution time comparison ($s = 6, e = 91$).

$\mathbf{t_{max}}$	$\mathbf{t_0}$	**TD weights**	**mean weights**	**hybrid weights**	**OPT**
180	**0**	613	613	613	613
	360	702	684	702	702
	720	734	694	734	753
	1080	769	734	747	775
	avg. profit	**704.5**	**681.3**	**699**	**710.8**
	avg. time	**0.6**	**0.8**	**0.8**	**96**
360	**0**	1253	1367	1369	-
	360	1373	1441	1444	-
	720	1413	1419	1434	-
	1080	1293	1314	1352	-
	avg. profit	**1333**	**1385.3**	**1399.8**	**-**
	avg. time	**1.2**	**1.5**	**2**	**-**
720	**0**	2308	2376	2362	-
	360	2368	2375	2407	-
	720	2246	2330	2341	-
	1080	2216	2267	2314	-
	avg. profit	**2284.5**	**2337**	**2356**	**-**
	avg. time	**2.6**	**2.5**	**4.5**	**-**
1440	**0**	3575	3844	3833	-
	360	3566	3794	3866	-
	720	3679	3929	3953	-
	1080	3546	3890	3911	-
	avg. profit	**3591.5**	**3864.3**	**3890.8**	**-**
	avg. time	**7.3**	**7**	**12.5**	**-**

For longer paths TD weights version is overtaken by other versions. The result difference between TD and hybrid version grows from 2 to over 5 percent. The best results are obtained by hybrid version, which uses both general and detail analysis. In most cases it holds an average advantage of 1–5 percent over mean weight version (except from t_{max}=720 where the difference is minimal). Short paths starting at midnight

(t_0=0) have smallest profits as a result of low rate of bus arrivals. The difference is more pronounced for shortest paths as most of travel takes place at night.

Execution time of the algorithms are acceptable even for very long paths. Hybrid version is more time-consuming than other versions (by 50-70 percent) because of two stage local search. In fig. 3 there is a path generated by the hybrid algorithm version.

Tab. 2. shows results for paths starting in point 6 and ending in point 91. They look similar to those in table 1. TD weights version holds advantage over other versions for shortest paths but gives worse results for longer paths. However differences are smaller and average results are lower than in table 1. This is due to high density of POIs in the city center – paths from tab. 1. start and end in the city center. It should be noted that for $t_{max} = 180$ TD weights version is on average only 0.9 percent worse than optimal solution.

Table 3. Results and execution time comparison ($s = 1, e = 51$).

$\mathbf{t_{max}}$	$\mathbf{t_0}$	**TD weights**	**mean weights**	**hybrid weights**	**OPT**
180	**0**	0	0	0	0
	360	677	659	659	687
	720	747	725	764	771
	1080	730	763	763	763
	avg. profit	**538.5**	**536.8**	**546.5**	**555.2**
	avg. time	**0.6**	**0.8**	**0.8**	**130**
360	**0**	311	311	311	-
	360	1398	1405	1415	-
	720	1382	1352	1394	-
	1080	1334	1353	1353	-
	avg. profit	**1106.3**	**1105.3**	**1118.3**	**-**
	avg. time	**1.2**	**1.5**	**2**	**-**
720	**0**	1707	1650	1699	-
	360	2298	2409	2421	-
	720	2402	2400	2381	-
	1080	2139	2356	2371	-
	avg. profit	**2136.5**	**2203.8**	**2218**	**-**
	avg. time	**2.5**	**2.7**	**4.7**	**-**
1440	**0**	3262	3381	3390	-
	360	3495	3877	3984	-
	720	3592	3830	3902	-
	1080	3628	3862	3937	-
	avg. profit	**3494.3**	**3737.5**	**3803.3**	**-**
	avg. time	**7.3**	**6.5**	**11**	**-**

Tab. 3. shows results for paths starting in point 1 and ending in point 51. In this case the hybrid weights version holds advantage over other versions for all t_{max} values. The average gap between optimal solution and hybrid weights version is 1.6 percent (t_{max}=180).

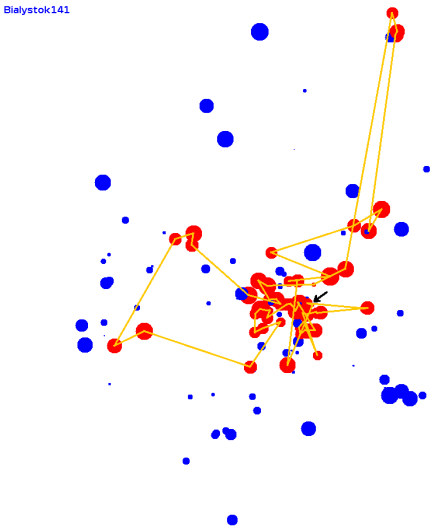


Fig. 3. An exemplary path ($s = e = 34$, $t_{max} = 1440$, $t_0 = 720$). Sizes of vertices are propotional to their profits and visited vertices are red. Yellow lines indicate path edges. Black arrow points to vertex 34.

7 Conclusions and further research

Results show that TDOP local search algorithm which performs calculations only on real time-dependent weights or solely on mean weights is not necessary the best option. For shorter paths (lowest t_{max} values) TD weights version can give the best results as detail analysis is very important when small modifications can result in significant result change. However, for longer paths general analysis (mean weights) is also significant – it can find important network patterns which could be unnoticed by TD weights analysis. As a result hybrid weight analysis seems to be most complete method and will be researched further.

Studies on how maximal number of transfers and walk link length affect network topology and TDOP results are currently performed. Design and tests of an evolutionary algorithm is also planned as such methods were successful in static weights OP. Tests on larger networks (with hundreds of POIs) and adding time windows are also intended.

References

1. Campos, V., Marti, R.: Grasp with Path Relinking for the Orienteering Problem, Technical Raport, 2011, 1–16
2. Chao, I. M., Golden, B. L., Wasil, E. A.: A Fast and effective heuristic for the orienteering, European Journal of Operational Research, 88 (1996), 475–489
3. Fischetti, M., Salazar, J. J., Toth, P.: Solving the Orienteering Problem through Branch-and-Cut, INFORMS Journal on Computing, 10 (1998), 133–148

4. Garcia, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., Linaza, M.T.: Integrating public transportation in personalised electronic tourist guides, Computers and Operations Research, 40 (3) (2013), 758–774
5. Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., Vathis, N.: Efficient Heuristics for the Time Dependent Team Orienteering Problem with Time Windows, Lecture Notes in Computer Science, 8321 (2014), 152–163
6. Gendreau, M., Laporte, G., Semet, F.: A branch-and-cut algorithm for the undirected selective traveling salesman problem, Networks, 32(4) (1998), 263–273
7. Golden, B., Levy, L., Vohra, R.: The orienteering problem, Naval Research Logistics, 34 (1987)
8. Karbowska-Chilińska, J., Koszelew, J., Ostrowski, K., Zabielski, P.: Genetic algorithm solving Orienteering Problem in large networks, Frontiers in Artificial Intelligence and Applications, 243 (2012)
9. Karbowska-Chilińska, J., Zabielski, P.: Genetic Algorithm vs. Local Search Methods Solving Orienteering Problem in Large Networks, Lecture Notes in Computer Science, 7828 (2012), 28–38
10. Koszelew, J., Ostrowski, K.: A genetic algorithm with multiple mutation which solves Orienteering Problem in large networks, Lecture Notes in Computer Science, 8083 (2013), 356–365
11. Li, J.: Model and Algorithm for Time-Dependent Team Orienteering Problem, Communications in Computer and Information Science, 175 (2011), 1–7
12. Ramesh, R., Yoon, Y., Karwan, M.H.: An optimal algorithm for the orienteering tour problem, INFORMS Journal on Computing, 4(2) (1992), 155–165
13. Schilde, M., Doerner, K., Hartl, R., Kiechle, G.: Metaheuristics for the bi-objective orienteering problem, Swarm Intelligence, 3 (2009), 179–201
14. Souffriau, W., Vansteenwegen, P.: Tourist Trip Planning Functionalities: State of the Art and Future, Springer, 6385 (2010), 474–485
15. Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., Van Oudheusden, D.: A path relinking approach for the team orienteering problem, Computers Operational Research, 37 (2010), 1853–1859
16. Tang, H., Miller-Hooks, E.: A tabu search heuristic for the team orienteering problem, Computers and Operation Research, 32(6) (2005), 1379–1407
17. Tasgetiren, M. F., Smith, A.E.: A genetic algorithm for the orienteering problem, Proceed. of the 2000 Congress on Evolutionary Computation, 2 (2000), 1190–1195
18. TDOP Białystok network, `http://p.wi.pb.edu.pl/sites/default/files/krzysztof-ostrowski/files/tdop_bialystok_network.zip`
19. Tsiligirides, T.: Heuristic methods applied to orienteering, Journal of the Operational Research Society, 35(9) (1984), 797–809
20. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: A guided local search metaheuristic for the team orienteering problem, European Journal of Operational Research, 196 (2009), 118–127
21. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: The City Trip Planner: An expert system for tourists, Elsevier, 38 (2011)

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

An Approach to Making Decisions with Metasets

Magdalena Kacprzak and Bartłomiej Starosta

[1] Bialystok University of Technology, Bialystok, Poland
[2] Polish-Japanese Institute of Information Technology, Warsaw, Poland
m.kacprzak@pb.edu.pl, barstar@pjwstk.edu.pl

Abstract. The metaset is a new approach to sets with partial membership relation. Metasets are designed to represent and process vague, imprecise data, similarly to fuzzy sets or rough sets. They make it possible to express fractional certainty of membership, equality, and other relations. In this paper we demonstrate an example of the application of first-order metasets to solving the problem of finding the most appropriate holiday destination for a tourist, taking his preferences into account. The imprecise idea of 'a perfect holiday destination' is represented as a metaset of places whose membership degrees in the metaset are interpreted as their qualities. Client preferences are functions which enable real-number evaluation of the subjective rating of a given destination.

Keywords: metaset, partial membership, set theory

1 Introduction

The metaset is a new attempt at defining the notion of sets with partial membership relation [10]. Metasets enable the representation and processing of vague, imprecise data, similarly to fuzzy sets [16] or rough sets [8]. In addition to fractional certainty of membership, equality or other set-theoretic relations and their negations, metasets admit a hesitancy degree of membership [11, 14, 13], similarly to intuitionistic fuzzy sets [1]. The general idea of the metaset is inspired by the method of forcing in classical set theory [2]. Despite these abstract origins, the definitions of metaset and related notions (i.e. set-theoretic relations or algebraic operations [12]) are directed towards efficient computer implementations and applications [9].

The following paper introduces an example of the application of this new approach. There are many real-life problems which require tools for modelling fractional satisfaction of some properties. They are usually solved by modelling these properties with the 'fuzzy' membership relation of a fuzzy or rough set. Here we try another theory, that of metasets. Since the set of membership values for metasets constitutes a partial order (in fact it is a Boolean algebra), there is great potential here for modelling of imprecise phenomena.

The specific contribution of this paper is to show how metasets can be applied to (software) tools which support decision-making. The problem we have selected is evaluation of the attractiveness of tourist destinations. These may be destinations that the user is considering in a tourist office as the best location for a holiday, but they may also be tourist attractions located in one city, such as monuments, galleries, museums,

etc. In both cases, the problem lies in the selection of one or more sites from among numerous proposals. The input in this problem is a list of sites with the location and a brief description of each. The output has to be a numeric score assigned to each location that allows us to compare them and ultimately select the best one. Difficulties that appear here include the following: (a) to select the most important attributes from the description of the site, (b) to personalize tourist preferences and (c) to assign a score that differentiates the locations in terms of tourist needs. In this paper we show how to use metasets to describe tourist preferences and how this representation helps to compute the degree of membership of a particular object in a set of perfect holiday destinations. It is emphasized that this degree will be different for different types of tourists and will be closely related to their preferences.

The proposed approach can be used in automated personalized tour-planning devices. In particular, it can be used in solving Tourist Trip Design Problems, TTDP (see e.g. [15]). The starting point in the TTDP is the orienteering problem (see e.g. [5, 6]). In this problem a set of nodes is given, each with a score. The goal is to determine a path, limited in length, that visits some nodes and maximizes the sum of the collected scores. However, before the solution to this problem is presented, values must be assigned to the nodes. This is where the algorithm we propose may be helpful. If the nodes represent locations, then by using metasets we can calculate scores which represent the interest of a given tourist.

The remainder of the paper is structured as follows. Section 2 gives the theoretical background, i.e., we briefly recall the main definitions and lemmas concerning metasets. Section 3 presents the problem of assigning to tourist locations an evaluation of their attractiveness and its solution in terms of metasets. Section 4 provides a generalization of the concept introduced. Conclusions are given in Section 5.

2 Metasets

A metaset is a classical crisp set with a specific internal structure which encodes the membership degrees of its members. Therefore, all Zermelo-Fraenkel [3, 7] axioms apply to metasets as well. The membership degrees are expressed as nodes of the binary tree $\mathbb{T}$. All the possible membership values make up a Boolean algebra. They can be evaluated as real numbers as well.

There are several fundamental notions associated with metasets which we recall in this section. These allow the membership relation to be defined and evaluated. We then use it to model the quality of tourist destinations and a client's preferences.

2.1 Basic Definitions

For simplicity, in this paper we deal only with first-order metasets.[3] A metaset of this type is a relation between some set and the set of nodes of the binary tree $\mathbb{T}$. Thus, the structure we use to encode the degrees of membership is based on ordered pairs. The first element of each pair is the member and the second element is a node of the binary tree which contributes to the membership degree of the first element.

Definition 1. *A set which is either the empty set $\emptyset$ or which has the form:*

$$\tau = \{\, \langle \sigma, p \rangle : \sigma \text{ is a set, } p \in \mathbb{T} \,\}$$

[3] See section 4 for the introduction to metasets in general.

is called a first-order metaset.

The class of first-order metasets is denoted by $\mathfrak{M}^1$. The binary tree $\mathbb{T}$ is the set of all finite binary sequences, i.e., functions whose domains are finite ordinals, valued in 2:[4]

$$\mathbb{T} = \bigcup_{n\in\mathbb{N}} 2^n \ . \tag{1}$$

The ordering $\leqslant$ in the tree $\mathbb{T}$ (see Fig. 1) is the reverse inclusion of functions: for $p, q \in \mathbb{T}$ such that $p\colon n \mapsto 2$ and $q\colon m \mapsto 2$, we have $p \leqslant q$ whenever $p \supseteq q$, i.e., $n \geqslant m$ and $p_{\restriction m} = q$. The root $\mathbb{1}$ being the empty function is the largest element of $\mathbb{T}$ in this ordering. It is included in each sequence and for all $p \in \mathbb{T}$ we have $p \leqslant \mathbb{1}$.

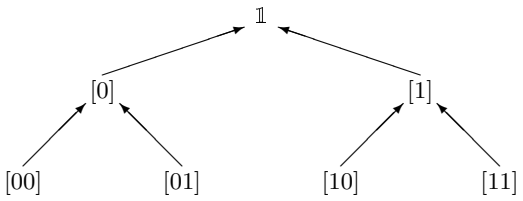


Fig. 1. The levels $\mathbb{T}_0$–$\mathbb{T}_2$ of the binary tree $\mathbb{T}$ and the ordering of nodes. Arrows point at the larger element.

We denote binary sequences which are elements of $\mathbb{T}$ using square brackets, for example: [00], [101]. If $p \in \mathbb{T}$, then we denote its children with $p \cdot 0$ and $p \cdot 1$. A *level* in $\mathbb{T}$ is the set of all finite binary sequences with the same length. The set 2^n consisting of sequences of the length n is the level n, denoted by $\mathbb{T}_n$. The level 0 consists of the empty sequence $\mathbb{1}$ only. A *branch* in $\mathbb{T}$ is an infinite binary sequence, i.e., a function $\mathbb{N} \mapsto 2$. Abusing the notation we will write $p \in \mathcal{C}$ to mark, that the binary sequence $p \in \mathbb{T}$ is a prefix of the branch $\mathcal{C}$. A branch intersects all levels in $\mathbb{T}$, and each of them only once.

Since a metaset is a relation, we may use the following standard notation. For the given $\tau \in \mathfrak{M}^1$, the set $\mathrm{dom}(\tau) = \{\, \sigma\colon \exists_{p\in\mathbb{T}}\ \langle\sigma, p\rangle \in \tau \,\}$ is called the *domain* of the metaset τ, and the set $\mathrm{ran}(\tau) = \left\{\, p\colon \exists_{\sigma\in\mathrm{dom}(\tau)}\ \langle\sigma, p\rangle \in \tau \,\right\}$ is called the *range* of the metaset τ.

A metaset is *finite* when it is finite as a set of ordered pairs. Consequently, its domain and range are finite. The class of finite first-order metasets is denoted by $\mathfrak{MF}^1$. Thus,

$$\tau \in \mathfrak{MF}^1 \quad \text{iff} \quad |\mathrm{dom}(\tau)| < \aleph_0 \wedge |\mathrm{ran}(\tau)| < \aleph_0 \ . \tag{2}$$

This class is particularly important for computer applications where we deal with finite objects exclusively.

2.2 Interpretations

An interpretation of a first-order metaset is a crisp set. It is produced out of a given metaset using a branch of the binary tree. Different branches determine different inter-

[4] For $n \in \mathbb{N}$, let $2^n = \{\, f\colon n \mapsto 2 \,\}$ denote the set of all functions with the domain n and the range $2 = \{\, 0, 1 \,\}$ – they are binary sequences of the length n.

pretations of the metaset. All of them taken together make up a collection of sets with specific internal dependencies, which represents the source metaset by means of its crisp views. Properties of crisp sets which are interpretations of the given first-order metaset determine the properties of the metaset itself. In particular we use interpretations to define set-theoretic relations for metasets.

Definition 2. *Let τ be a first-order metaset and let $\mathcal{C}$ be a branch. The set*

$$\tau_{\mathcal{C}} = \{\, \sigma \in \mathrm{dom}(\tau) \colon\ \langle \sigma, p \rangle \in \tau \wedge p \in \mathcal{C} \,\}$$

is called the interpretation of the first-order metaset τ given by the branch $\mathcal{C}$.

An interpretation of the empty metaset is the empty set, independently of the branch.

The process of producing an interpretation of a first-order metaset consists in two stages. In the first stage we remove all the ordered pairs whose second elements are nodes which do not belong to the branch $\mathcal{C}$. The second stage replaces the remaining pairs – whose second elements lie on the branch $\mathcal{C}$ – with their first elements. As the result we obtain a crisp set contained in the domain of the metaset.

Example 1. Let $p \in \mathbb{T}$ and let $\tau = \{\, \langle \emptyset, p \rangle \,\}$. If $\mathcal{C}$ is a branch, then

$$p \in \mathcal{C} \rightarrow \tau_{\mathcal{C}} = \{\, \emptyset \,\}\ ,$$
$$p \notin \mathcal{C} \rightarrow \tau_{\mathcal{C}} = \emptyset\ .$$

Depending on the branch the metaset τ acquires one of two different interpretations: $\{\, \emptyset \,\}$ or $\emptyset$. Note, that $\mathrm{dom}(\tau) = \{\, \emptyset \,\}$.

As we see, a first-order metaset may have multiple different interpretations – each branch in the tree determines one. Usually, most of them are pairwise equal, so the number of different interpretations is much less than the number of branches. Finite first-order metasets always have a finite number of different interpretations.

2.3 Partial Membership

We use interpretations for transferring set-theoretic relations from crisp sets onto metasets.[5] In this paper we discuss only the partial membership.

Definition 3. *We say that the metaset σ belongs to the metaset τ under the condition $p \in \mathbb{T}$, whenever for each branch $\mathcal{C}$ containing p holds $\sigma_{\mathcal{C}} \in \tau_{\mathcal{C}}$. We use the notation $\sigma\ \epsilon_p\ \tau$.*

Formally, we define an infinite number of membership relations: each $p \in \mathbb{T}$ specifies another relation ϵ_p. Any two metasets may be simultaneously in multiple membership relations qualified by different nodes: $\sigma\ \epsilon_p\ \tau \wedge \sigma\ \epsilon_q\ \tau$. Membership under the root condition $\mathbb{1}$ resembles the full, unconditional membership of crisp sets, since it is independent of branches.

The conditional membership reflects the idea that an element σ belongs to a metaset τ whenever some conditions are fulfilled. The conditions are represented by nodes of $\mathbb{T}$. There are two substantial properties of this technique exposed by the following two lemmas.

[5] For the detailed discussion of the relations or their evaluation the reader is referred to [12] or [14].

Lemma 1. *Let $\tau, \sigma \in \mathfrak{M}^1$ and let $p, q \in \mathbb{T}$. If $\sigma \in_p \tau$ and $q \leqslant p$, then $\sigma \in_q \tau$.*

Proof. If $\mathcal{C}$ is a branch containing q then also $p \in \mathcal{C}$. Therefore $\sigma_{\mathcal{C}} \in \tau_{\mathcal{C}}$.

Lemma 2. *Let $\tau, \sigma \in \mathfrak{M}^1$ and let $p \in \mathbb{T}$. If $\forall_{q<p}\, \sigma \in_q \tau$, then $\sigma \in_p \tau$.*

Proof. If $\mathcal{C} \ni p$, then it also contains some $q < p$. Therefore, $\sigma_{\mathcal{C}} \in \tau_{\mathcal{C}}$.

In other words: $\sigma \in_p \tau$ is equivalent to $\sigma \in_{p\cdot 0} \tau \wedge \sigma \in_{p\cdot 1} \tau$, i.e., being a member under the condition p is equivalent to being a member under both conditions $p\cdot 0$ and $p\cdot 1$, which are the direct descendants of p. Indeed, by lemma 1 we have $\sigma \in_p \tau \;\rightarrow\; \sigma \in_{p\cdot 0} \tau \wedge \sigma \in_{p\cdot 1} \tau$. And if $\sigma \in_{p\cdot 0} \tau$, then again, by lemma 1 we have $\forall_{q\leqslant p\cdot 0}\, \sigma \in_q \tau$, and similarly for $p\cdot 1$. Consequently, we have $\forall_{q<p}\, \sigma \in_q \tau$ and by lemma 2 we obtain $\sigma \in_{p\cdot 0} \tau \wedge \sigma \in_{p\cdot 1} \tau \;\rightarrow\; \sigma \in_p \tau$.

Example 2. Recall, that the ordinal number 1 is the set $\{\,0\,\}$ and 0 is just the empty set $\emptyset$. Let $\tau = \{\,\langle 0, [0]\rangle, \langle 1, [1]\rangle\,\}$ and let $\sigma = \{\,\langle 0, [1]\rangle\,\}$. Let $\mathcal{C}^0 \ni [0]$ and $\mathcal{C}^1 \ni [1]$ be arbitrary branches containing $[0]$ and $[1]$, respectively. Interpretations are: $\tau_{\mathcal{C}^0} = \{\,0\,\}$, $\tau_{\mathcal{C}^1} = \{\,1\,\}$, $\sigma_{\mathcal{C}^0} = 0$ and $\sigma_{\mathcal{C}^1} = \{\,0\,\} = 1$. We see that $\sigma \in_{[0]} \tau$ and $\sigma \in_{[1]} \tau$. Also, $\sigma \in_{\mathbb{1}} \tau$ holds.

Note, that even though interpretations of τ and σ vary depending on the branch, the metaset membership relation is preserved.

2.4 Evaluating Membership

Membership degrees for metasets are expressed as nodes of $\mathbb{T}$. In fact, these nodes determine the basis of the Boolean Algebra of closed-open sets in the Cantor space 2^ω. Indeed, a $p \in \mathbb{T}$ is just a prefix for all infinite binary sequences which form a clopen subset of 2^ω. Thus, the membership relation for metasets is valued in the Boolean algebra. Nonetheless, for the sake of simplicity and in applications we usually refer to the binary tree when talking about membership.

In applications we frequently need a numerical evaluation of membership degrees. To define it first we consider the smallest subset of $\mathbb{T}$ consisting of elements which determine the membership.

Definition 4. *Let $\sigma, \tau \in \mathfrak{M}^1$. The set*

$$\|\sigma \in \tau\| = \max\{\, p \in \mathbb{T} \colon \sigma \in_p \tau \,\}$$

is called the certainty grade for membership of σ in τ.

Note that by definition 3, $\|\sigma \in \tau\| = \max\{\, p \in \mathbb{T} \colon \forall_{\mathcal{C}\ni p}\, \sigma_{\mathcal{C}} \in \tau_{\mathcal{C}} \,\}$. Lemmas 1 and 2 justify definition 4. Indeed, if for $q \in \mathbb{T}$ the membership $\sigma \in_q \tau$ is satisfied, which means that for any branch $\mathcal{C}$ containing q it holds that $\sigma_{\mathcal{C}} \in \tau_{\mathcal{C}}$, then $q \in \{\, p \in \mathbb{T} \colon \forall_{\mathcal{C}\ni p}\, \sigma_{\mathcal{C}} \in \tau_{\mathcal{C}} \,\}$. Therefore, there exists a $p \in \|\sigma \in \tau\|$ such that $q \leqslant p$. And by lemma 1, each such p implies that $\sigma \in_q \tau$, for $q \leqslant p$. This means that all the necessary membership information is contained in $\|\sigma \in \tau\|$. Moreover, no incorrect membership information can be inferred from this set. If for $r \leqslant s$ it is not true that $\sigma \in_r \tau$, then $s \in \|\sigma \in \tau\|$ would contradict lemma 1. Note also that if $\sigma \in_{q\cdot 0} \tau$ and $\sigma \in_{q\cdot 1} \tau$, then consequently for any $r < q$ it holds that $\sigma \in_r \tau$, and therefore by lemma 2 it holds that $\sigma \in_q \tau$. Thus, the set of all $p \in \mathbb{T}$ such that $\sigma \in_p \tau$ consists of subtrees whose roots are in $\|\sigma \in \tau\|$.

We define the numerical evaluation of membership by composing the membership function valued in 2^ω with the natural transformation $2^\omega \mapsto [0, 1]$ as follows.

Definition 5. *Let $\sigma, \tau \in \mathfrak{M}^1$. The following value is called the certainty value of membership of σ in τ:*

$$|\sigma \in \tau| = \sum_{p \in \|\sigma \in \tau\|} \frac{1}{2^{|p|}} \ .$$

Recall that $|p|$ is the length of the binary sequence p, which is equal to the number of the level containing p. One may easily see that $|\sigma \in \tau| \in [0, 1]$.

For the sake of the main topic of the discussion it is worth noticing that in the above definition we treat all the nodes within the same level uniformly, without distinguishing one from another. All nodes on the given level contribute the same factor of $\frac{1}{2^{|p|}}$ to the membership value. This will not be the case for the problem of evaluation of client preferences, where we modify this function.

3 The Problem

In this section we show how use metasets to solve the problem of assigning to tourist locations evaluations of their attractiveness. Here we will use a real data set from the city of Białystok. In the problem that we want to solve, a set of locations is given. We assume, that these are places that can be visited during a one-day trip. At this point we disregard the questions of how many places a tourist may wish to choose, the distance between them, and the means of arrival, focusing only on their attractiveness. The places we choose from can be divided into the following categories: (a) buildings (church, museum, gallery, etc.), (b) sports facilities (playgrounds, skate parks, etc.) and (c) outdoor places (park, forest, river, etc.).

Example 3. We selected attributes that may be important for the selection on the basis of interviews with potential tourists: (a) the possibility of eating something (*Fast food*, *Slow food*), (b) a place to sit and rest (*Physical rest*), (c) sports infrastructure (*Sports*), (d) the chance to explore the city (*Sightseeing*) and (e) a good place for hiking (*Hiking*). Classification of two locations, the Resort of Water Sports in Dojlidy (*RWS*) and Branicki Park & Planty (*PBP*), according to these attributes is shown in Table 1.

	Fast food	*Slow food*	*Physical rest*	*Sports*	*Sightseeing*	*Hiking*
RWS	YES	NO	YES	YES	NO	YES
PBP	YES	YES	YES	NO	YES	YES

Table 1. Classification of locations

3.1 Modelling with Metasets

Our goal is to differentiate these tourist locations according to their attractiveness. To this end we build a binary tree designated by the expression 'a perfect holiday destination'. For most tourists a good place for a holiday is somewhere they can relax and enjoy leisure activities (e.g., hiking, bike riding, sports). Therefore two branches

extend from the root of this tree: *Relaxation* and *Activity* (see Fig. 2). Relaxation requires a convenient location (node: *Rest*) as well as a place for dining (node: *Food*). Places to rest include those that offer a respite for the body (beach, forest, etc.) (node: *Body*) or the soul (restaurant with live music, performances, etc.) (node: *Soul*). Among dining options we can distinguish between those which serve pizza, hamburgers or hot dogs (node: *Fast food*) and those that offer regional, organic, vegetarian cuisine, etc. (node: *Slow food*). Available activities can be classified as follows: sports (node:

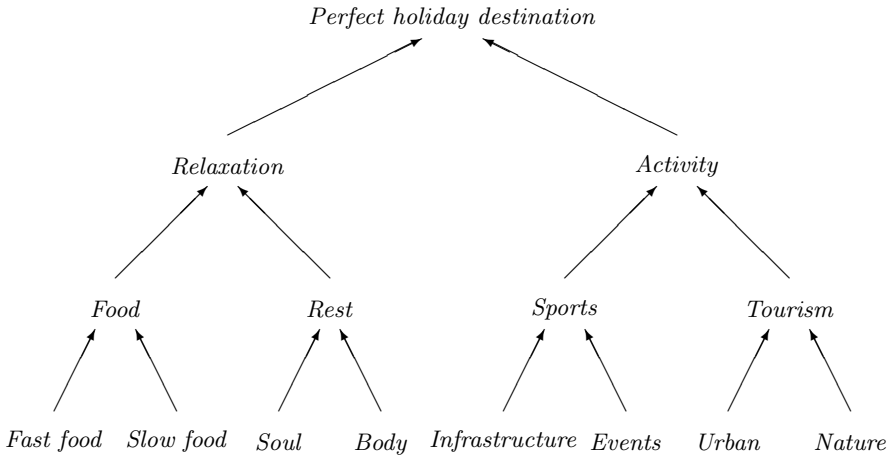


Fig. 2. The tree for *Perfect holiday destination*

Sports) and tourism more generally (node: *Tourism*). Sports require infrastructure (node: *Infrastructure*). Sometimes it is also possible to take part in sporting events, such as a marathon (node: *Events*). Tourism can be in an urban area, and then includes visiting churches, galleries or architecturally interesting buildings (node: *Urban*). Some tourists prefer such activities as walking in the mountains or strolling through a park (node: *Nature*).

The reminder of the tree is built in a similar way. The greater the height of the tree, the more detailed the feedback. Taking into account a large number of attributes leads to more accurate assignment of locations to the tourist. The subtree for the *Infrastructure* node is depicted in Fig. 3. *Infrastructure* consists of locations such as swimming pools (node: *Locations*) and other facilities (node: *Facilities*). The most important of these are buildings (node: *Buildings*) and the possibility of buying (node: *Buy*) or renting (node: *Rent*) equipment. The tree we have built is only an example and can be changed depending on specific applications and needs.

Let us return to the tourist. First we define his expectations. We can do this using one of the following methods: (a) communication in natural language (a dialogue), (b) human-computer communication using an appropriate application (e.g. implementation of formal dialogue games [4]), or (c) a survey. To give an example, let us postulate two tourists with different expectations.

Example 4. Consider two sample tourists, *Ann* and *Ben*.

Ann is an active person. She likes sports, especially running. She strongly prefers nature to the city. She does not eat much, because she is constantly on a diet.

Ben prefers a relaxing holiday. Eating well is his highest priority. He does not like fast food. He enjoys bus tours and sightseeing. He does not like to play sports, but he is a fan of spectator sports. He enjoys watching concerts and shows.

We formalize the preferences of these tourists in example 5.

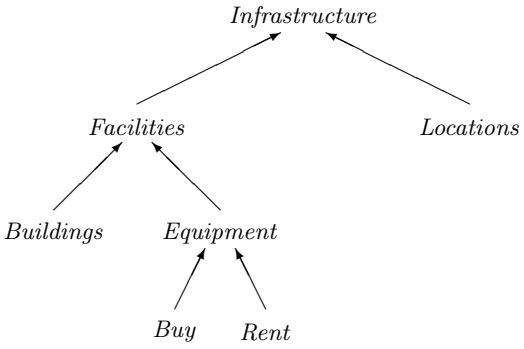


Fig. 3. Subtree for *Infrastructure* node

3.2 Evaluating Client Preferences

Definition 5 assumes uniform distribution of values throughout the nodes in $\mathbb{T}$: each $p \in \|\delta \in \Delta\|$ contributes the value of $\frac{1}{2^{|p|}}$ to $|\delta \in \Delta|$.

In the context discussed in the paper this might be interpreted as a client's indifference as to what to choose: all possible choices represented as nodes within the same level are equally weighted.

For a $p \in \mathbb{T}$ both its children $p \cdot 0$ and $p \cdot 1$ contribute equally to the membership evaluation. Usually, however, clients have some preferences concerning places or activities and this preference may be expressed numerically.

To evaluate the quality of a destination taking client preferences into account we modify the definition 5 slightly to obtain an evaluation function which increases the impact of some nodes and decreases that of others. We build this function based on an interview with the client.

Definition 6. *We define client preference to be a function* $\mathfrak{p}\colon \mathbb{T} \mapsto [0,1]$ *such that*

$$\forall_{q \in \mathbb{T}} \quad \mathfrak{p}(q \cdot 0) + \mathfrak{p}(q \cdot 1) = 1 \,. \tag{3}$$

and we take $\mathfrak{p}(\mathbb{1}) = 1$ *for the root.*

Now we may evaluate the quality of the destination δ taking preferences $\mathfrak{p}$ into account to obtain the subjective value of the quality of the destination as follows:

Definition 7. *Let δ be a destination and let Δ be a metaset of destinations. The* $\mathfrak{p}$*-quality of the destination δ is the following value:*

$$|\delta \in \Delta|_{\mathfrak{p}} = \sum_{q \in \|\delta \in \Delta\|} \prod_{0 \leqslant i \leqslant |q|} \mathfrak{p}(q_{\restriction i}) \, .$$

The symbol $q_{\restriction i}$, where $0 \leqslant i \leqslant |q|$ denotes all the consecutive prefixes of the binary sequence q, including the empty one (for $i = 0$) and q itself (for $i = |q|$, which is the length of the sequence q). Note that $q_{\restriction |q|} = q$, since $\mathrm{dom}(q) = |q|$ and $q_{\restriction \emptyset} = \emptyset = \mathbb{1}$.

The $\mathfrak{p}$-quality of a destination reflects a client's preferences. For different clients with different $\mathfrak{p}$ preference functions it may result in different ratings for the given destination. We discuss this and present examples in the following section.

3.3 Solution to the Problem

To demonstrate the advantages of our approach we take into account the preferences of the clients mentioned in example 4 when comparing the two destinations defined in example 3. We have two sample locations with opposite characteristics: 'active' (RWS) and 'non-active' (PBP). There are also two clients: *Ann*, an active person, and *Ben*, who has a sedentary lifestyle. We show that the evaluated client preferences for particular locations are consistent with common sense: *Ann* prefers RWS while *Ben* prefers PBP. In particular, we claim that

$$|PBP \in \Delta|_{Ann} \leqslant |RWS \in \Delta|_{Ann} \, , \tag{4}$$

and

$$|RWS \in \Delta|_{Ben} \leqslant |PBP \in \Delta|_{Ben} \, . \tag{5}$$

The expression $|\delta \in \Delta|_X$ is the real number representing the quality of δ as a *'perfect holiday destination'* (i.e. the membership value of δ in Δ), taking into account the preferences of the client X. The above formulas formally express the fact that *Ann* prefers active destinations and *Ben* non-active ones. In example 4 we expressed sample preferences using natural language. We now demonstrate the $\mathfrak{p}$ functions for these clients, constructed following a detailed investigation of their preferences (personal interview or computer-aided tool). The functions are depicted in Fig. 4 and Fig. 5.

Example 5. Ann prefers *Activity* to *Relaxation*. We found that this preference is expressed by the ratio 3/1, so we set $\mathfrak{p}(Activity) = 0.75$ and $\mathfrak{p}(Relaxation) = 0.25$ (see Fig. 4). She likes playing sports and having a good rest afterwards. She professed a ratio of 7/10 in favour of *Sports* over *Tourism*. Since she does not eat much, we assumed a ratio of 1/5 between eating and resting. She prefers *Nature* to *Urban* tourism. We assume here a ratio of 4/1 and therefore we set $\mathfrak{p}(Nature) = 0.8$ and $\mathfrak{p}(Urban) = 0.2$. We know that she rarely attends sporting events and therefore we set $\mathfrak{p}(Events) = 0.1$ and $\mathfrak{p}(Infrastructure) = 0.9$. For all other nodes $q \neq \mathbb{1}$ we set $\mathfrak{p}(q) = 0.5$.

Let us now consider *Ben*. Since he dislikes any form of *Activity*, we assume $\mathfrak{p}(Activity) = 0.15$ and $\mathfrak{p}(Relaxation) = 0.85$. Eating well is *Ben*'s most significant preference, so we assume $\mathfrak{p}(Food) = 0.75$ and $\mathfrak{p}(Rest) = 0.25$. Because we know he values good food, we assume $\mathfrak{p}(Fast\ food) = 0.05$ and $\mathfrak{p}(Slow\ food) = 0.95$. Ben's detailed preferences are depicted in Fig. 5.

The value of 0.5 may be interpreted as indifference towards a particular choice in both cases.

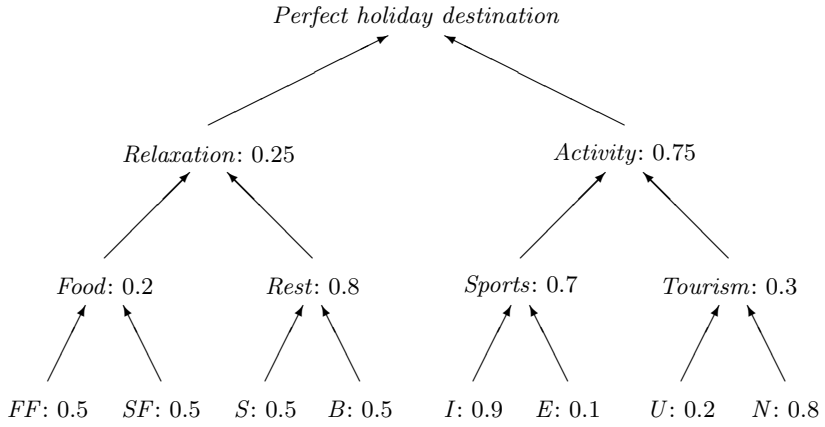


Fig. 4. *Ann's* preferences (we used abbreviations in the last level).

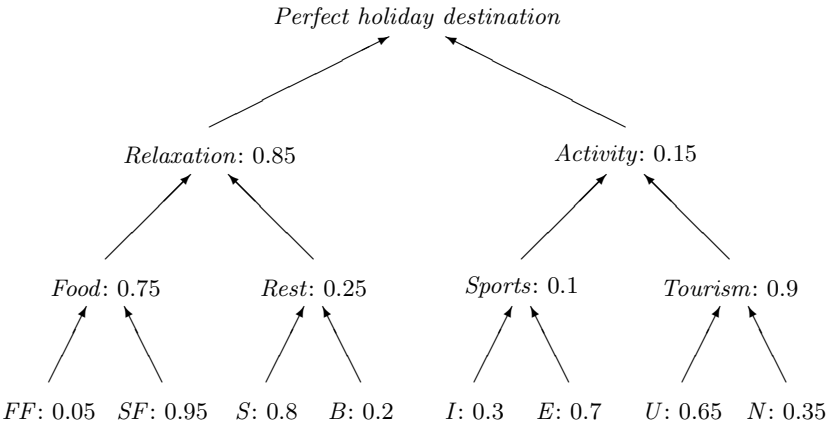


Fig. 5. *Ben's* preferences (we used abbreviations in the last level).

We now show how the preferences of the clients in example 4 affect the subjective quality of a given destination. First, for reference, we calculate the numerical value of the membership, which also represents the preferences of a totally indifferent client.

We consider the two locations, *RWS* and *PBP*, with the following attributes (cf. Ex. 3).

$$RWS : \ Fast\ food, Body, Infrastructure, Nature\ , \tag{6}$$
$$PBP : \ Food, Soul, Urban\ . \tag{7}$$

First, let us calculate the degrees of membership of both places in the metaset Δ consisting of *perfect holiday destinations*. They tell us the measure of objective quality of these places, i.e. the degree to which the idea of a perfect holiday destination is satisfied by these particular destinations. These degrees are also equal to those resulting from evaluation of the preferences of a totally indifferent client.

$$\|RWS \in \Delta\| = \{\ \textit{Fast food},\ \textit{Body},\ \textit{Infrastructure},\ \textit{Nature}\ \}\ , \tag{8}$$
$$= \{\ [000], [011], [100], [111]\ \}\ , \tag{9}$$
$$\|PBP \in \Delta\| = \{\ \textit{Food},\ \textit{Soul},\ \textit{Urban}\ \}\ , \tag{10}$$
$$= \{\ [00], [010], [110]\ \}\ . \tag{11}$$

The numerical values for the quality of the destinations are as follows:

$$|RWS \in \Delta| = \frac{1}{2^{|[000]|}} + \frac{1}{2^{|[011]|}} + \frac{1}{2^{|[100]|}} + \frac{1}{2^{|[111]|}}\ , \tag{12}$$
$$= \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} \ = \ \frac{4}{8} \ = \ 0.5\ , \tag{13}$$
$$|PBP \in \Delta| = \frac{1}{2^{|[00]|}} + \frac{1}{2^{|[010]|}} + \frac{1}{2^{|[110]|}}\ , \tag{14}$$
$$= \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} \ = \ \frac{4}{8} \ = \ 0.5\ . \tag{15}$$

We now apply both client's preferences to calculate subjective qualities:

$$|RWS \in \Delta|_{Ann} = 0.25 \cdot 0.2 \cdot 0.5 + 0.25 \cdot 0.8 \cdot 0.5$$
$$+ \ 0.75 \cdot 0.7 \cdot 0.9 + 0.75 \cdot 0.3 \cdot 0.8 \tag{16}$$
$$= 0.025 + 0.1 + 0.4725 + 0.18 \tag{17}$$
$$= 0.7775\ , \tag{18}$$
$$|PBP \in \Delta|_{Ann} = 0.25 \cdot 0.2 + 0.25 \cdot 0.8 \cdot 0.5 + 0.75 \cdot 0.3 \cdot 0.2 \tag{19}$$
$$= 0.05 + 0.1 + 0.045 \tag{20}$$
$$= 0.195\ , \tag{21}$$
$$\tag{22}$$

$$|RWS \in \Delta|_{Ben} = 0.85 \cdot 0.75 \cdot 0.05 + 0.85 \cdot 0.25 \cdot 0.2$$
$$+ \ 0.15 \cdot 0.1 \cdot 0.3 + 0.15 \cdot 0.9 \cdot 0.35 \tag{23}$$
$$= 0.031875 + 0.0425 + 0.0045 + 0.04725 \tag{24}$$

$$= 0.126125\ , \tag{25}$$

$$|PBP \in \Delta|_{Ben} = 0.85 \cdot 0.75 + 0.85 \cdot 0.25 \cdot 0.8 + 0.15 \cdot 0.9 \cdot 0.65 \tag{26}$$

$$= 0.6375 + 0.17 + 0.08775 \tag{27}$$

$$= 0.89525\ . \tag{28}$$

Thus, we obtained the value of 0.7775 as the measure of *Ann*'s interest in *RWS* and the value of 0.195 representing her interest in *PBP*, which is much lower. The value of 0.126125 confirms *Ben*'s aversion to spending time actively in comparison with the value of 0.89525, which reflects his strong interest in destinations allowing for a good rest and meals.

The results confirm the accuracy of our approach; they are consistent with common sense. As expected, for *Ann* the metaset model suggests *RWS*, where she is able to practise sports, and for *Ben PBP*, where he can have a rest and eat well. At this stage of development we cannot determine whether or not the proposed method is better than others. We will investigate this topic in the future and the results of the comparison will be publicized.

4 Generalization and Further Results

The definitions of metaset and related notions used in the paper are simplified versions of a much more general concept. Although first-order metasets are sufficient for the simple application discussed, for completeness we cite below the general definitions of *metaset* and *interpretation* (see [10] for a further discussion of metasets). The reader familiar with the method of forcing in set theory [3, 7] will find some similarities here. They are rather superficial, since the prototype was designed for an entirely different purpose. Also, when applying metasets we are usually dealing with finite sets, which makes no sense in the case of forcing.

Definition 8. *A set which is either the empty set $\emptyset$ or which has the form:*

$$\tau = \{\, \langle \sigma, p \rangle : \sigma \text{ is a metaset, } p \in \mathbb{T} \,\}$$

is called a metaset.

Formally, this is a definition by induction on the well-founded relation $\in$ (see [7, Ch. VII, §2] for a justification of this type of definitions). The general definition of interpretation for metasets is recursive as well.

Definition 9. *Let τ be a metaset and let $\mathcal{C} \subset \mathbb{T}$ be a branch. The set*

$$\tau_{\mathcal{C}} = \{\, \sigma_{\mathcal{C}} : \ \langle \sigma, p \rangle \in \tau \wedge p \in \mathcal{C} \,\}$$

is called the interpretation of the metaset τ given by the branch $\mathcal{C}$.

For most applications, especially computer applications [9], first-order metasets - even finite first-order metasets - are sufficient. All the results presented here, together with the model itself, remain valid if we omit the assumption that the metasets involved are first-order. The definitions given above are used in the general development of the theory of metasets.

5 Conclusions

In this paper we explained a simple application of the new concept of sets with partial membership relation. We used metasets to model and solve the problem of selecting the best holiday destination for a client with specific preferences regarding how he spends his free time.

The metaset approach enables destinations and their properties to be rated using natural human-language terms. The core of the idea lies in constructing a treelike hierarchy of terms which describe the attributes of destinations and at the same time the requirements of clients. The hierarchy involves a relationship between attributes of the 'generalization-specialization' type.

Metasets are the perfect tool for evaluating imprecise expressions. The example we investigated here is that of a 'perfect holiday destination'. Of course there is no one perfect place, just as there are no two persons having the same taste. The ideal place for one person to relax may give rise to resentment in another. This is a problem frequently encountered by the designers of mobile applications such as mobile tourist guides, which attempt to automatically determine which places to visit in a given region. The easiest way to determine the perfect location is to evaluate its popularity, i.e. how many people visit it or how many recommend it in polls or on Internet forums. In this way we overlook people with unusual preferences. Our approach eliminates this disadvantage, because the starting point of our algorithm is to identify user preferences and describe them in the form of a tree. Then the algorithm selects a location to suit those preferences. In addition, the test objects are set in a partial order, which will precisely take into account the needs of a specific person. Of course, the final decision belongs to the user, but our algorithm can provide professional support. In our further work we will compare the approach presented in this paper with algorithms described in the research literature which are used for similar problems in logistics or tourism.

Acknowledgements

The first author gratefully acknowledge support from the Polish Ministry of Science and Higher Education at the Bialystok University of Technology (grant S/WI/1/2014).

References

1. Atanassov, K. T.: Intuitionistic Fuzzy Sets. Fuzzy Sets and Systems 20, pp. 87–96 (1986).
2. Cohen, P.: The Independence of the Continuum Hypothesis 1. Proceedings of the National Academy of Sciences of the United States of America 50, 1143–1148 (1963).
3. Jech, T.: Set Theory: The Third Millennium Edition, Revised and Expanded. Springer-Verlag, Berlin Heidelberg New York (2006).
4. Kacprzak M., Sawicka A.: Identification of formal fallacies in a natural dialogue, Fundamenta Informaticae (2014) (accepted for publication).
5. Karbowska-Chilinska, J., Zabielski, P.: A Genetic Algorithm vs. Local Search Methods for Solving the Orienteering Problem in Large Networks. Knowledge Engineering, Machine Learning and Lattice Computing with Applications, Lecture Notes in Computer Science. 7828 (2013), pp. 11–20

6. Karbowska-Chilinska, J., Zabielski, P.: A Genetic Algorithm Solving Orienteering Problem with Time Windows. Advances in Systems Science. Advances in Intelligent Systems and Computing, Springer. 240 (2014), 609–619
7. Kunen, K.: Set Theory, An Introduction to Independence Proofs. No. 102 in Studies in Logic and Foundations of Mathematics, North-Holland Publishing Company, Amsterdam (1980)
8. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982).
9. Starosta, B.: Application of Metasets to Character Recognition. In: Rauch, J. et al. (eds.), ISMIS 2009. LNCS (LNAI), vol. 5722, pp. 602–611. Springer-Verlag Berlin Heidelberg (2009)
10. Starosta, B.: Metasets: A New Approach to Partial Membership, In: Rutkowski, L. et al. (eds.), Artificial Intelligence and Soft Computing, LNAI 7267 pp. 325–333. Springer-Verlag (2012).
11. Starosta, B.: Representing Intuitionistic Fuzzy Sets as Metasets. In: Atanassov, K.T. et al. (eds.), Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Volume I: Foundations, pp. 185–208. Systems Research Institute, Polish Academy of Sciences, Warsaw (2010).
12. Starosta, B., Kosiński, W.: Meta Sets. Another Approach to Fuzziness. In: Seising, R. (ed.) Views on Fuzzy Sets and Systems from Different Perspectives. Philosophy and Logic, Criticisms and Applications. STUDFUZZ, vol. 243, pp. 509–522. Springer-Verlag Berlin Heidelberg (2009).
13. Starosta, B., Kosiński, W.: Metasets, Intuitionistic Fuzzy Sets and Uncertainty. In: Rutkowski, L. et al. (eds.), Artificial Intelligence and Soft Computing, LNAI 7894, pp. 388–399. Springer-Verlag (2013).
14. Starosta, B., Kosiński, W.: Metasets, Certainty and Uncertainty. In: Atanassov, K.T. et al. (eds.), New Trends in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Volume I: Foundations, pp. 139–165. Systems Research Institute, Polish Academy of Sciences, Warsaw (2013).
15. Souffriau, W., Vansteenwegen P., Vertommen J., Vanden Berghe G., Van Oudheusden D.: A personalizes tourist trip design algorithm for mobile tourist guides, Applied Artificial Intelligence: An International Journal, vol. 22(10), pp. 964–985 (2008)
16. Zadeh, L. A.: Fuzzy Sets. Information and Control 8, 338–353 (1965)

Part IV

Formal Methods and Data Mining

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

On the SAT-based Verification of Communicative Commitments ⋆

Bożena Woźna-Szcześniak

IMCS, Jan Długosz University, Al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland.
b.wozna@ajd.czest.pl

Abstract. We present CDCTL⋆K, a temporal logic to specify knowledge, correct functioning behaviour, and different social commitments in multi-agent systems (MAS). We interpret the formulae of the logic over models generated by Communication Deontic Interpreted Systems (CDIS). Furthermore, we investigate a SAT-based bounded model checking (BMC) technique for the existential fragments of CDCTL⋆K (called ECDCTL⋆K) and for CDIS. Finally, we exemplify the use of the technique by means of the NetBill protocol, a popular example in the MAS literature related to modelling of business processes.

1 Introduction

Multi-agent systems (MASs) [19] are distributed systems in which computational components (called *agents*) are autonomous and self-interested entities. The agents are able to communicate, negotiate, coordinate, and control their own behaviour in the furtherance of their own goals.

During the last two decades, a plethora of social techniques that aim to define a formal semantics for agent communication languages (ACLs) have been proposed, e.g., [16, 6]. These approaches particularly target to tackle the shortcomings of ACLs semantics defined by means of mental approaches [22]; the mental semantics is expressed in terms of the agents' internal mental states such as believes, goals, desires and intentions. Some of these social approaches use (communicative) commitments to model multi-agent interactions [7, 16] or to represent business contracts [2]. Following [7], in the paper we consider CDCTL⋆K, which is a new ACL that extends CTL⋆ [4] with modalities for knowledge [10], correct functioning behaviour [11], commitments [7], group commitment, and conditional deontic (group) commitment.

The formalism of *interpreted systems* (IS) [10] provides a useful framework to model MASs and to verify various classes of temporal and epistemic properties. The formalism of *deontic interpreted systems* (DIS) [11] is an extension of ISs, which makes possible reasoning about temporal, epistemic and correct functioning behaviour of MASs. The formalism of *communication interpreted systems* (CIS) [7] is an extension of ISs, which makes possible reasoning about temporal, epistemic and social behaviour of MASs. In the paper we consider a new formalism of *communication deontic interpreted systems* (CDIS) which is a fusion of DIS and CIS, and thereby it allows for reasoning about

⋆ Partly supported by National Science Center under the grant No. 2011/01/B/ST6/05317.

temporal, epistemic, correct functioning behaviour, and social behaviour of MASs. CDIS provides a computationally grounded semantics for CDCTL*K.

Verification and modelling of commitment properties of MASs was first investigated by Venkatraman and Singh [18]. Then, it was further investigated, among others, in [5, 1, 6]. However, all the verification approaches that were developed in this line of research are translation-based. Thus, they do not offer dedicated model-checking algorithms for verification of social properties of MASs. Moreover, they do not provide dedicated modalities with computationally grounded semantics for commitments and related concepts. A direct methods via either developing a dedicated model checker from scratch or extending an existing model checker with new algorithms for commitments and their fulfillment and violation have been proposed in [14, 7].

Bounded model checking (BMC) [3, 15] is an automatic verification technique that takes advantage of the immense success of SAT-solvers, and that consists in translating the model checking problem [4] for a modal logic (i.e., the problem of determining whether a given logical formula representing a specification is satisfied in a particular formal model representing the executions of a system) to the satisfiability problem of a propositional formula. Originally developed for the verification of Boolean circuits and for LTL properties [3], during the last decade, BMC has become an active subject of research in the area of MASs [15, 12, 21]. The aim of this paper is to report on recent progress on the application of the SAT-based BMC to verifying no just temporal and epistemic, but also social and deontic properties of MAS. In particular, we propose a SAT-based BMC for CDIS and for ECDCTL*K, which is the existential fragment of CDCTL*K.

The rest of the paper is organised as follows. In Section 2 we introduce CDIS together with its Kripke model, and the CDCTL*K language together with its two subsets: ACDCTL*K and ECDCTL*K. Moreover, we define the unbounded semantics for the whole language, and a bounded semantics for the ECDCTL*K subset. In Section 3 we provide a SAT-based BMC method for ECDCTL*K. In Section 4 we apply the BMC technique to the NetBill protocol. In the last section we conclude the paper with a short discussion and an outline of our future work.

2 Preliminaries

Let us start by fixing some notation used through the paper. Let $\mathbb{A} = \{1, \ldots, n, \mathcal{E}\}$ be the non-empty and finite set of agents, $\mathcal{PV}$ the set of propositional variables, and $\mathbb{Z}$ the set of integers.

Interpreted Systems (IS). The standard formal definition of IS assumes that MASs consist of n agents and the special agent $\mathcal{E}$ that is used to model the environment in which the other agents operate. Furthermore, IS$=(\{\iota_{\mathbf{c}}, L_{\mathbf{c}}, Act_{\mathbf{c}}, P_{\mathbf{c}}, t_{\mathbf{c}}, \mathcal{V}_{\mathbf{c}}\}_{\mathbf{c}\in\mathbb{A}})$ which means that each agent $\mathbf{c} \in \mathbb{A}$ is modelled by using a non-empty set $L_{\mathbf{c}}$ of *local states*, a non-empty set $\iota_{\mathbf{c}} \subseteq L_{\mathbf{c}}$ of *initial local states*, a non-empty set $Act_{\mathbf{c}}$ of *possible actions*, a *protocol function* $P_{\mathbf{c}} : L_{\mathbf{c}} \to 2^{Act_{\mathbf{c}}}$ defining the action selection mechanism, a (partial) *evolution function* $t_{\mathbf{c}} : L_{\mathbf{c}} \times Act \to L_{\mathbf{c}}$ (each element of $Act = \prod_{\mathbf{c}\in\mathbb{A}} Act_{\mathbf{c}}$, as usually, is called *joint action*), and a valuation function $\mathcal{V}_{\mathbf{c}} : L_{\mathbf{c}} \to 2^{\mathcal{PV}}$ that assigns to each local state a set of propositional variables that are assumed to be true at that state.

The local states $L_{\mathbf{c}}$ model the instantaneous configuration of the agent $\mathbf{c}$ in MAS. The content varies according to what we need to model, e.g. it may be the values of some (local) variables. The environment $\mathcal{E}$ captures relevant information that is not

specific to any individual agent, e.g. messages in transit in a communication channel; it is assumed that local states for $\mathcal{E}$ are *public*. Furthermore, as in [10], we represent the instantaneous snapshot of MAS at a given time by means of *global states*. Namely, a set of all possible *global states* is defined as $S = L_1 \times \ldots \times L_n \times L_{\mathcal{E}}$. We will write $l_{\mathbf{c}}(s)$ to denote the local state of agent $\mathbf{c} \in \mathbb{A}$ in a global state $s = (\ell_1, \ldots, \ell_n, \ell_{\mathcal{E}})$.

For each agent $\mathbf{c} \in \mathbb{A}$ we define a standard indistinguishability relation $\sim_{\mathbf{c}} \subseteq S \times S$ as: $s \sim_{\mathbf{c}} s'$ iff $l_{\mathbf{c}}(s') = l_{\mathbf{c}}(s)$. We will use this relation to give the computationally grounded semantics for standard epistemic properties of MAS. Moreover, we define a *global evolution function* $t : S \times Act \to S$ as follows: $t(s, a) = s'$ iff $t_{\mathbf{c}}(l_{\mathbf{c}}(s), a) = l_{\mathbf{c}}(s')$ for all $\mathbf{c} \in \mathbb{A}$. In brief we write the above as $s \xrightarrow{a} s'$.

Deontic Interpreted Systems (DIS). In DIS it is assumed that for each agent $\mathbf{c} \in \mathbb{A}$, its set of local states $L_{\mathbf{c}}$ can be partitioned into two disjoint sets: a non-empty set $\|\Phi\|_{\mathbf{c}}$ of *faultless (green)* states and a set $\mathcal{R}_{\mathbf{c}}$ of *faulty (red)* states. Thus, $L_{\mathbf{c}} = \|\Phi\|_{\mathbf{c}} \cup \mathcal{R}_{\mathbf{c}}$ and DIS is defined as the following tuple: $(\{\iota_{\mathbf{c}}, L_{\mathbf{c}}, \|\Phi\|_{\mathbf{c}}, Act_{\mathbf{c}}, P_{\mathbf{c}}, t_{\mathbf{c}}, \mathcal{V}_{\mathbf{c}}\}_{\mathbf{c} \in \mathbb{A}})$.

As in [11], we make the following assumption on the set S of all possible *global states*: $L_{\mathbf{c}} \supseteq \|\Phi\|_{\mathbf{c}}$ for each $\mathbf{c} \in \mathbb{A}$. Furthermore, for each agent $\mathbf{c} \in \mathbb{A}$ we define a *deontic relation* $\bowtie_{\mathbf{c}} \subseteq S \times S$ as: $s \bowtie_{\mathbf{c}} s'$ iff $l_{\mathbf{c}}(s') \in \|\Phi\|_{\mathbf{c}}$. We will use this relation to give the computationally grounded semantics for the deontic properties of MAS.

Communication Interpreted Systems (CIS). In CIS it is assumed that a finite set $Var_{\mathbf{c}}$ of local integer variables is associated with each agent $\mathbf{c} \in \mathbb{A}$. These variables are used to represent communication channels through which messages are sent and received, and then to define the *social accessibility* relation, which in turn will be used to define the computationally grounded semantics of *communication and deontic commitments*. Each local state $\ell \in L_{\mathbf{c}}$ of agent $\mathbf{c}$ is associated with different values obtained from different assignments to variables in $Var_{\mathbf{c}}$, and CIS is defined as the following tuple: $(\{\iota_{\mathbf{c}}, L_{\mathbf{c}}, Var_{\mathbf{c}}, Act_{\mathbf{c}}, P_{\mathbf{c}}, t_{\mathbf{c}}, \mathcal{V}_{\mathbf{c}}\}_{\mathbf{c} \in \mathbb{A}})$.

Let $s = (\ell_1, \ldots, \ell_n, \ell_{\mathcal{E}}) \in S$ be a global state. As in [7], we denote the value of a variable $x \in Var_{\mathbf{c}}$ at local state $l_{\mathbf{c}}(s)$ by $l^x_{\mathbf{c}}(s)$, and we assume that if $l_{\mathbf{c}}(s) = l_{\mathbf{c}}(s')$, then $l^x_{\mathbf{c}}(s) = l^x_{\mathbf{c}}(s')$ for all $x \in Var_{\mathbf{c}}$. Furthermore, for each pair $(\mathbf{c}_1, \mathbf{c}_2)$ of agents in $\mathbb{A}$ we define a *social accessibility* relation $\sim_{\mathbf{c}_1 \to \mathbf{c}_2} \subseteq S \times S$ as: $s \sim_{\mathbf{c}_1 \to \mathbf{c}_2} s'$ iff $l_{\mathbf{c}_1}(s) = l_{\mathbf{c}_1}(s')$, and $Var_{\mathbf{c}_1} \cap Var_{\mathbf{c}_2} \neq \emptyset$ such that $\forall x \in Var_{\mathbf{c}_1} \cap Var_{\mathbf{c}_2}$ we have $l^x_{\mathbf{c}_1}(s) = l^x_{\mathbf{c}_2}(s')$ and $\forall y \in Var_{\mathbf{c}_2} - Var_{\mathbf{c}_1}$ we have $l^y_{\mathbf{c}_2}(s) = l^y_{\mathbf{c}_2}(s')$, and $s \xrightarrow{a} s'$.

The intuition behind the definition of the social accessibility relation $\sim_{\mathbf{c}_1 \to \mathbf{c}_2}$ is the following. The states s and s' are indistinguishable for $\mathbf{c}_1$ ($l_{\mathbf{c}_1}(s) = l_{\mathbf{c}_1}(s')$), since $\mathbf{c}_1$ initiates the communication and it does not learn any new information. There is a communication channel between $\mathbf{c}_1$ and $\mathbf{c}_2$ ($Var_{\mathbf{c}_1} \cap Var_{\mathbf{c}_2} \neq \emptyset$). The channel is filled in by $\mathbf{c}_1$ in state s, and in state s' $\mathbf{c}_2$ receives the information, which makes the value of the shared variable the same for $\mathbf{c}_1$ and $\mathbf{c}_2$ ($l^x_{\mathbf{c}_1}(s) = l^x_{\mathbf{c}_2}(s')$). The states s and s' are indistinguishable for $\mathbf{c}_2$ with regard to the variables that have not been communicated by $\mathbf{c}_1$, i.e., unshared variables $(\forall y \in Var_{\mathbf{c}_2} - Var_{\mathbf{c}_1}) l^y_{\mathbf{c}_2}(s) = l^y_{\mathbf{c}_2}(s')$.

Communication Deontic Interpreted Systems (CDIS). A formalism of CDIS is defined as the following tuple: $(\{\iota_{\mathbf{c}}, L_{\mathbf{c}}, \|\Phi\|_{\mathbf{c}}, Var_{\mathbf{c}}, Act_{\mathbf{c}}, P_{\mathbf{c}}, t_{\mathbf{c}}, \mathcal{V}_{\mathbf{c}}\}_{\mathbf{c} \in \mathbb{A}})$.

Model. Let $\mathbf{c} \in \mathbb{A}$ and $v_{\mathbf{c}} : Var_{\mathbf{c}} \to \mathbb{Z}$ be a valuation function that assigns to each variable $x \in Var_{\mathbf{c}}$ an integer value $v_{\mathbf{c}}(x)$. Moreover, let c_{max} (c_{min}) be the maximal (minimal) value that can be assign to variable $x \in \bigcup_{\mathbf{c} \in \mathbb{A}} Var_{\mathbf{c}}$, and $\mathbb{C} = \{c_{min}, \ldots, c_{max}\}$. $\mathbb{C}^{|Var_{\mathbf{c}}|}$ is the finite set of all the valuations for agent $\mathbf{c}$ that are induced by the function $v_{\mathbf{c}}$. Then, for a given CDIS we define a *model* as a tuple $M = (S, \iota, T, \mathcal{V}, \sim_{\mathbf{c}}, \bowtie_{\mathbf{c}}, \sim_{\mathbf{c}_1 \to \mathbf{c}_2})$, where

- $S = \prod_{\mathbf{c}\in\mathbb{A}} L_{\mathbf{c}}$ is the set of all possible global states such that $L_{\mathbf{c}} \supseteq \|\Phi\|_{\mathbf{c}}$ and $L_{\mathbf{c}} \subseteq \mathbb{C}^{|Var_{\mathbf{c}}|}$ for each $\mathbf{c} \in \mathbb{A}$,
- $\iota = \prod_{\mathbf{c}\in\mathbb{A}} \iota_{\mathbf{c}}$ is the set of all possible initial global states,
- $T \subseteq S \times S$ is a total transition relation on S defined by: $(s, s') \in T$ iff there exists an action $a \in Act$ such that $s \xrightarrow{a} s'$,
- $\mathcal{V} : S \to 2^{\mathcal{PV}}$ is the valuation function defined as $\mathcal{V}(s) = \bigcup_{\mathbf{c}\in\mathbb{A}} \mathcal{V}_{\mathbf{c}}(l_{\mathbf{c}}(s))$,
- $\sim_{\mathbf{c}} \subseteq S \times S$ is the indistinguishability relation defined as above for IS,
- $\bowtie_{\mathbf{c}} \subseteq S \times S$ is the deontic relation defined as above for DIS, and
- $\sim_{\mathbf{c}_1\to\mathbf{c}_2} \subseteq S \times S$ is the social accessibility relation defined as above for CIS.

A *path* in M is an infinite sequence $\pi = (s_0, s_1, \ldots)$ of states such that $(s_m, s_{m+1}) \in T$ for each $m \in \mathbb{N}$. Now, let $m \in \mathbb{N}$. Then, $\pi(m) = s_m$ and it denotes the m-th state of π. $\pi^m = (s_m, s_{m+1}, \ldots)$ and it denotes the m-th suffix of π. $\Pi(s)$ denotes the set of all the paths starting at $s \in S$, and $\Pi = \bigcup_{s^0\in\iota} \Pi(s^0)$ denotes the set of all the paths starting at initial states.

Syntax of CDCTL*K**.** Let $p \in \mathcal{PV}$ be a propositional variable, $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{A}$, $\Gamma \subseteq \mathbb{A}$. The syntax of CDCTL*K, which is a combination of branching time CTL* [9, 8] with standard epistemic modalities, the deontic notion due to [11], social commitments due to [7], and with new modalities for *group social commitments* and *conditional deontic (group) commitment*, is defined as follows:

$$\begin{aligned}\varphi ::= &\text{true} \mid \text{false} \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{A}\phi \mid \mathrm{K}_{\mathbf{c}}\phi \mid \mathrm{E}_\Gamma\phi \mid \mathrm{D}_\Gamma\phi \mid \mathrm{C}_\Gamma\phi \mid \\ &\mathcal{O}_{\mathbf{c}}\phi \mid \widehat{\mathrm{K}}^{\mathbf{c}_2}_{\mathbf{c}_1}\phi \mid \mathrm{C}_{i\to j}\phi \mid \mathrm{C}_{i\to\Gamma}\phi \mid \mathrm{D}_{i\to j}\phi \mid \mathrm{D}_{i\to\Gamma}\phi \\ \phi ::= &\varphi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathrm{X}\phi \mid \phi\mathrm{U}\phi \mid \phi\mathrm{R}\phi\end{aligned}$$

where φ is a *state formula* and ϕ is a *path formula*, A is the universal quantifier on paths, X, U, and R are CTL* path modalities standing for *next*, *until*, and *release* respectively. The modalities $\mathrm{K}_{\mathbf{c}}, \mathrm{D}_\Gamma, \mathrm{E}_\Gamma$, and C_Γ represent, respectively, *knowledge of agent* $\mathbf{c}$, *distributed knowledge in the group* Γ, *everyone in* Γ *knows*, and *common knowledge among agents in* Γ. The modalities $\mathcal{O}_{\mathbf{c}}$ and $\widehat{\mathrm{K}}^{\mathbf{c}_2}_{\mathbf{c}_1}$ represent, respectively, the *correctly functioning circumstances (with respect to some protocol) of agent* $\mathbf{c}$ and *knowledge of agent* $\mathbf{c}_1$ *under the assumption that* $\mathbf{c}_2$ *is functioning correctly*. The modalities $\mathrm{C}_{i\to j}$ and $\mathrm{C}_{i\to\Gamma}$ stand for *commitment* and *group commitment*, respectively. The modalities $\mathrm{D}_{i\to j}$ and $\mathrm{D}_{i\to\Gamma}$ stand for *conditional deontic commitment* and *conditional deontic group commitment*, respectively. CDCTL*K consists of the set of state formulae generated by the above grammar. For more details on commitment modality $\mathrm{C}_{i\to j}$ we refer to [7]. Other temporal, epistemic and deontic modalities are given by their usual abbreviations, i.e. $\mathrm{F}\phi \stackrel{def}{=} \text{true}\mathrm{U}\phi$, $\mathrm{G}\phi \stackrel{def}{=} \text{false}\mathrm{R}\phi$, $\overline{\mathrm{K}}_{\mathbf{c}}\phi \stackrel{def}{=} \neg\mathrm{K}_{\mathbf{c}}\neg\phi$, $\overline{\mathrm{D}}_\Gamma\phi \stackrel{def}{=} \neg\mathrm{D}_\Gamma\neg\phi$, $\overline{\mathrm{E}}_\Gamma\phi \stackrel{def}{=} \neg\mathrm{E}_\Gamma\neg\phi$, $\overline{\mathrm{C}}_\Gamma\phi \stackrel{def}{=} \neg\mathrm{C}_\Gamma\neg\phi$, $\overline{\mathcal{O}}_{\mathbf{c}}\phi \stackrel{def}{=} \neg\mathcal{O}_{\mathbf{c}}\neg\phi$, $\underline{\widehat{\mathrm{K}}}^{\mathbf{c}_2}_{\mathbf{c}_1}\phi \stackrel{def}{=} \neg\widehat{\mathrm{K}}^{\mathbf{c}_2}_{\mathbf{c}_1}\neg\phi$, $\overline{\mathrm{C}}_{i\to j}\phi \stackrel{def}{=} \neg\mathrm{C}_{i\to j}\neg\phi$, $\overline{\mathrm{C}}_{i\to\Gamma}\phi \stackrel{def}{=} \neg\mathrm{C}_{i\to\Gamma}\neg\phi$, $\overline{\mathrm{D}}_{i\to j}\phi \stackrel{def}{=} \neg\mathrm{D}_{i\to j}\neg\phi$, $\overline{\mathrm{D}}_{i\to\Gamma}\phi \stackrel{def}{=} \neg\mathrm{D}_{i\to\Gamma}\neg\phi$.

In this logic, $\mathrm{C}_{i\to j}\phi$ is read as *agent* i *commits towards agent* j *that* ϕ, or equivalently as ϕ *is committed to by* i *towards* j. $\mathrm{C}_{i\to\Gamma}\phi$ is read as *agent* i *commits towards group of agent* Γ *that* ϕ, or equivalently as ϕ *is committed to by* i *towards group of agent* Γ. Furthermore, $\mathrm{D}_{i\to j}\phi$ is read as *agent* i *commits towards agent* j *that* ϕ *if, and only if agent* j *is functioning correctly*, or equivalently as ϕ *is committed to by* i *towards* j *if, and only if agent* j *is functioning correctly*. $\mathrm{D}_{i\to\Gamma}\phi$ is read as *agent* i *commits towards group of agent* Γ *that* ϕ *if, and only if the group* Γ *is functioning correctly*, or equivalently as ϕ *is committed to by* i *towards group of agent* Γ *if, and only if the group* Γ *is functioning correctly*.

We find useful to consider the universal and the existential fragments of CDCTL*K, which we denote by ACDCTL*K and ECDCTL*K, respectively. The universal frag-

ment is used to express properties of a system in question. The negations of these properties can be expressed in the existential fragment, and therefore they can be verified by means of the bounded model checking method that is presented in the next section.

ACDCTL*K is defined by the following grammar: $\varphi ::= \text{true} \mid \text{false} \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{A}\phi \mid \mathrm{K}_{\mathbf{c}}\phi \mid \mathrm{E}_{\Gamma}\phi \mid \mathrm{D}_{\Gamma}\phi \mid \mathrm{C}_{\Gamma}\phi \mid \mathcal{O}_{\mathbf{c}}\phi \mid \widehat{\mathrm{K}}^{\mathbf{c}_2}_{\mathbf{c}_1}\phi \mid \mathrm{C}_{i\to j}\phi \mid \mathrm{C}_{i\to\Gamma}\phi \mid \mathrm{D}_{i\to j}\phi \mid \mathrm{D}_{i\to\Gamma}\phi$; $\phi ::= \varphi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathrm{X}\phi \mid \phi \mathrm{U} \phi \mid \phi \mathrm{R} \phi$.

ECDCTL*K is defined by the following grammar: $\varphi ::= \text{true} \mid \text{false} \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{E}\phi \mid \overline{\mathrm{K}}_{\mathbf{c}}\phi \mid \overline{\mathrm{E}}_{\Gamma}\phi \mid \overline{\mathrm{D}}_{\Gamma}\phi \mid \overline{\mathrm{C}}_{\Gamma}\phi \mid \overline{\mathcal{O}}_{\mathbf{c}}\phi \mid \underline{\widehat{\mathrm{K}}}^{\mathbf{c}_2}_{\mathbf{c}_1}\phi \mid \overline{\mathrm{C}}_{i\to j}\phi \mid \overline{\mathrm{C}}_{i\to\Gamma}\phi \mid \overline{\mathrm{D}}_{i\to j}\phi \mid \overline{\mathrm{D}}_{i\to\Gamma}\phi$; $\phi ::= \varphi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathrm{X}\phi \mid \phi \mathrm{U} \phi \mid \phi \mathrm{R} \phi$.

Semantics of CDCTL*K**.** The semantics of CDCTL*K formulae is defined with respect to the model $M = (S, \iota, T, \mathcal{V}, \sim_{\mathbf{c}}, \bowtie_{\mathbf{c}}, \sim_{\mathbf{c}_1\to\mathbf{c}_2})$ as defined above. In the semantics we assume the following definitions of epistemic relations: $\sim^{E}_{\Gamma} \stackrel{def}{=} \bigcup_{\mathbf{c}\in\Gamma} \sim_{\mathbf{c}}$, $\sim^{C}_{\Gamma} \stackrel{def}{=} (\sim^{E}_{\Gamma})^{+}$ (the transitive closure of $\sim^{E}_{\Gamma}$), $\sim^{D}_{\Gamma} \stackrel{def}{=} \bigcap_{\mathbf{c}\in\Gamma} \sim_{\mathbf{c}}$, where $\Gamma \subseteq \mathbb{A}$.

Let M be a model, s a state of M, π a path in M, and $m \in \mathbb{N}$. For a state formula α over $\mathcal{PV}$, the notation $M, s \models \alpha$ means that α holds at the state s in the model M. Similarly, for a path formula φ over $\mathcal{PV}$, the notation $M, \pi \models \varphi$ means that φ holds along the path π in the model M. Moreover, let $p \in \mathcal{PV}$ be a propositional variable, α, β be state formulae of CDCTL*K, and φ, ψ be path formulae of CDCTL*K. The relation $\models$ is defined inductively with the classical rules for the CTL* fragment of CDCTL*K, and with the following rules for epistemic, deontic and commitment modalities:

$M, s \models \mathrm{K}_{\mathbf{c}}\alpha$ iff $(\forall \pi \in \Pi)(\forall i \geqslant 0)(s \sim_{\mathbf{c}} \pi(i)$ implies $M, \pi^i \models \alpha)$,

$M, s \models Y_{\Gamma}\alpha$ iff $(\forall \pi \in \Pi)(\forall i \geqslant 0)(s \sim^{Y}_{\Gamma} \pi(i)$ implies $M, \pi^i \models \alpha)$, with $Y \in \{\mathrm{D}, \mathrm{E}, \mathrm{C}\}$,

$M, s \models \mathcal{O}_{\mathbf{c}}\alpha$ iff $(\forall \pi \in \Pi)(\forall i \geqslant 0)(s \propto_{\mathbf{c}} \pi(i)$ implies $M, \pi^i \models \alpha)$,

$M, s \models \widehat{\mathrm{K}}^{\mathbf{c}_2}_{\mathbf{c}_1}\alpha$ iff $(\forall \pi \in \Pi)(\forall i \geqslant 0)((s \sim_{\mathbf{c}_1} \pi(i)$ and $s \bowtie_{\mathbf{c}_2} \pi(i))$ implies $M, \pi^i \models \alpha)$,

$M, s \models \mathrm{C}_{i\to j}\alpha$ iff $(\forall \pi \in \Pi)(\forall i \geqslant 0)(s \sim_{\mathbf{c}_1\to\mathbf{c}_2} \pi(i)$ implies $M, \pi^i \models \alpha)$

$M, s \models \mathrm{C}_{i\to\Gamma}\alpha$ iff $(\forall \pi \in \Pi)(\forall i \geqslant 0)(\forall \mathbf{c}_2 \in \Gamma)(s \sim_{\mathbf{c}_1\to\mathbf{c}_2} \pi(i)$ implies $M, \pi^i \models \alpha)$

$M, s \models \mathrm{D}_{i\to j}\alpha$ iff $(\forall \pi \in \Pi)(\forall i \geqslant 0)((s \sim_{\mathbf{c}_1\to\mathbf{c}_2} \pi(i)$ and $s \bowtie_{\mathbf{c}_2} \pi(i))$ implies $M, \pi^i \models \alpha)$

$M, s \models \mathrm{D}_{i\to\Gamma}\alpha$ iff $(\forall \pi \in \Pi)(\forall i \geqslant 0)(\forall \mathbf{c}_2 \in \Gamma)((s \sim_{\mathbf{c}_1\to\mathbf{c}_2} \pi(i)$ and $s \bowtie_{\mathbf{c}_2} \pi(i))$ implies $M, \pi^i \models \alpha)$.

A CDCTL*K state formula α is *valid* in M, denoted by $M \models \alpha$, iff for each $s \in \iota$, $M, s \models \alpha$, i.e., α holds at every initial state of M. The *model checking problem* asks whether $M \models \alpha$.

Bounded Semantics of ECDCTL*K**.** Let M be a model, $k \in \mathbb{N}$, and $0 \leqslant l \leqslant k$. A *k-path* is a pair (π, l), also denoted by π_l, where π is a finite sequence $\pi = (s_0, \ldots, s_k)$ of states such that $(s_j, s_{j+1}) \in T$ for each $0 \leqslant j < k$. A k-path π_l is a *loop* if $l < k$ and $\pi(k) = \pi(l)$. If a k-path π_l is a loop it represents the infinite path of the form uv^{ω}, where $u = (\pi(0), \ldots, \pi(l))$ and $v = (\pi(l+1), \ldots, \pi(k))$. We denote this unique path by $\varrho(\pi_l)$. Note that for each $j \in \mathbb{N}$, $\varrho(\pi_l)^{l+j} = \varrho(\pi_l)^{k+j}$. Moreover, $\Pi_k(s)$ denotes the set of all the k-paths starting at $s \in S$, and $\Pi_k = \bigcup_{s^0\in\iota} \Pi_k(s^0)$ denotes the set of all the paths starting at initial states.

Let s be a state of M and π_l a k-path in Π_k. For a state formula α over $\mathcal{PV}$, the notation $M, s \models_k \alpha$ means that α is k-true at the state s in the model M. Similarly, for a path formula φ over $\mathcal{PV}$, the notation $M, \pi^m_l \models_k \varphi$, where $0 \leqslant m \leqslant k$, means that φ

is k-true along the suffix $(\pi(m),\ldots,\pi(k))$ of π. The relation $\models_k$ is defined inductively with the [23] rules for the ECTL* fragment of ECDCTL*K, and with the following rules for epistemic, deontic and commitment modalities:

$M,s \models_k \overline{\mathrm{K}}_{\mathbf{c}}\alpha$ iff $(\exists \pi \in \Pi)(\exists i \geqslant 0)(s \sim_{\mathbf{c}} \pi(i)$ and $M,\pi^i_l \models_k \alpha)$,
$M,s \models_k \overline{Y}_{\Gamma}\alpha$ iff $(\exists \pi \in \Pi)(\exists i \geqslant 0)(s \sim^{Y}_{\Gamma} \pi(i)$ and $M,\pi^i_l \models_k \alpha)$ and $Y \in \{\mathrm{D},\mathrm{E},\mathrm{C}\}$,
$M,s \models_k \overline{\mathcal{O}}_{\mathbf{c}}\alpha$ iff $(\exists \pi \in \Pi)(\exists i \geqslant 0)(s \propto_{\mathbf{c}} \pi(i)$ and $M,\pi^i_l \models_k \alpha)$,
$M,s \models_k \widehat{\underline{\mathrm{K}}}^{\mathbf{c}_2}_{\mathbf{c}_1}\alpha$ iff $(\exists \pi \in \Pi)(\exists i \geqslant 0)(s \sim_{\mathbf{c}_1} \pi(i)$ and $s \bowtie_{\mathbf{c}_2} \pi(i)$ and $M,\pi^i_l \models_k \alpha)$,
$M,s \models_k \overline{\mathrm{C}}_{i\to j}\alpha$ iff $(\exists \pi \in \Pi)(\exists i \geqslant 0)(s \sim_{\mathbf{c}_1\to\mathbf{c}_2} \pi(i)$ and $M,\pi^i_l \models_k \alpha)$,
$M,s \models_k \overline{\mathrm{C}}_{i\to\Gamma}\alpha$ iff $(\exists \pi \in \Pi)(\exists i \geqslant 0)(\forall \mathbf{c}_2 \in \Gamma)(s \sim_{\mathbf{c}_1\to\mathbf{c}_2} \pi(i)$ and $M,\pi^i_l \models_k \alpha)$,
$M,s \models_k \overline{\mathrm{D}}_{i\to j}\alpha$ iff $(\exists \pi \in \Pi)(\exists i \geqslant 0)(s \sim_{\mathbf{c}_1\to\mathbf{c}_2} \pi(i)$ and $s \bowtie_{\mathbf{c}_2} \pi(i)$ and $M,\pi^i_l \models_k \alpha)$,
$M,s \models_k \overline{\mathrm{D}}_{i\to\Gamma}\alpha$ iff $(\exists \pi \in \Pi)(\exists i \geqslant 0)(\forall \mathbf{c}_2 \in \Gamma)(s \sim_{\mathbf{c}_1\to\mathbf{c}_2} \pi(i)$ and $s \bowtie_{\mathbf{c}_2} \pi(i)$ and $M,\pi^i_l \models_k \alpha)$.

An ECDCTL*K state formula α is k-valid (true) in M, denoted $M \models_k \varphi$, iff for each $s \in \iota$, $M,s \models_k \varphi$. The *bounded model checking problem* asks whether there exists $k \in \mathbb{N}$ such that $M \models_k \varphi$.

The following theorem states that for a given model M and an ECDCTL*K formula α there exists a bound such that the model checking problem can be reduced to the bounded model checking problem.

Theorem 1. *Let M be a model and α an* ECDCTL*K *state formula. Then, for each $s \in \iota$, $M,s \models \alpha$ iff $M,s \models_k \alpha$ for some $k \in \mathbb{N}$.*

3 Bounded Model Checking

In the following section we present a propositional encoding of the BMC problem for ECDCTL*K and for CDIS. The encoding is based on the BMC encoding introduced in [20, 23, 13]. In [23] a propositional encoding of the BMC problem for ECTL* and for standard Kripke models has been introduced and experimentally evaluated. Next, in [20] a propositional encoding of the BMC problem for RTCTLKD and for deontic interleaved interpreted systems has been introduced and experimentally evaluated. Further, in [13] a SAT- and BDD-based encoding of the BMC problem for LTLK and for (interleaved) interpreted systems has been introduced and experimentally evaluated.

Translation to the propositional satisfiability problem. Let $M = (S,\iota,T,\mathcal{V},\sim_{\mathbf{c}},\bowtie_{\mathbf{c}},\sim_{\mathbf{c}_1\to\mathbf{c}_2})$ be a model, α an ECDCTL*K state formula, and $k \in \mathbb{N}$ a bound. The propositional encoding of the BMC problem for ECDCTL*K and for CDIS, as usually, relies on defining the following propositional formula:

$$[M,\alpha]_k := [M^{\alpha,\iota}]_k \wedge [\alpha]_{M,k} \tag{1}$$

which is satisfiable if and only if $M \models_k \alpha$ holds.

The definition of $[M^{\alpha,\iota}]_k$ assumes that the states and the joint actions of M are encoded symbolically, which is possible, since both the set of states and the set of joint actions are finite. Formally, let $\mathbf{c} \in \mathbb{A}$. Then, each state $s = (\ell_1,\ldots,\ell_n,\ell_{\mathcal{E}}) \in S$ is represented by a *symbolic state* which is a vector $\mathbf{w} = (\mathtt{w}_1,\ldots,\mathtt{w}_n,\mathtt{w}_{\mathcal{E}})$ of *symbolic local states*. Each symbolic local state $\mathtt{w}_{\mathbf{c}}$ is a vector of propositional variables (called *state variables*) whose length t is equal to $|Var_{\mathbf{c}}| \cdot c$ with $c = max\{|c_{min}|,|c_{max}|\}+1$. Next, each joint action $a = (a_1,\ldots,a_n,a_{\mathcal{E}}) \in Act$ is represented by a *symbolic action* which

is a vector $\mathbf{a} = (\mathtt{a}_1, \ldots, \mathtt{a}_n, \mathtt{a}_{\mathcal{E}})$ of *symbolic local actions*. Each symbolic local action $\mathtt{a}_{\mathbf{c}}$ is a vector of propositional variables (called *action variables*) whose length depends on the number of actions of agent $\mathbf{c}$. Moreover, by $\mathbf{u} = (\mathtt{u}_1, \ldots, \mathtt{u}_y)$ we denote a vector of natural variables of length $y = max(1, \lceil log_2(k+1) \rceil)$, which we call a *symbolic number*.

Let $\mathbf{w}$ and $\mathbf{w}'$ be two different symbolic states, $\mathbf{a}$ a symbolic action, and $\mathbf{u}$ a symbolic number. We assume definitions of the following auxiliary propositional formulae:

- $p(\mathbf{w})$ – encodes a set of states of M in which proposition variable $p \in \mathcal{PV}$ holds.
- $I_s(\mathbf{w})$ – encodes the state s of the model M.
- $H(\mathbf{w}, \mathbf{w}')$ – encodes equality of two symbolic states, i.e. it expresses that the symbolic states $\mathbf{w}$ and $\mathbf{w}'$ represent the same states.
- $H_{\mathbf{c}}(\mathbf{w}, \mathbf{w}')$ – encodes that the local states of agent $\mathbf{c}$ are the same in the symbolic states $\mathbf{w}$ and $\mathbf{w}'$.
- $O_{\mathbf{c}}(\mathbf{w})$ – encodes that the local states of agent $\mathbf{c}$ in the global state represented by $\mathbf{w}$ is green,
- $\widehat{H}^{\mathbf{c}_2}_{\mathbf{c}_1}(\mathbf{w}, \mathbf{w}') := H_{\mathbf{c}}(\mathbf{w}, \mathbf{w}') \wedge O_{\mathbf{c}}(\mathbf{w}')$,
- $\mathcal{A}(\mathbf{a})$ – encodes that each symbolic local action $\mathtt{a}_{\mathbf{c}}$ of $\mathbf{a}$ has to be executed by each agent in which it appears.
- $\mathcal{T}_{\mathbf{c}}(\mathtt{w}_{\mathbf{c}}, \mathbf{a}, \mathtt{w}'_{\mathbf{c}})$ – encodes the local evolution function of agent $\mathbf{c}$.
- $\mathcal{T}(\mathbf{w}, \mathbf{a}, \mathbf{w}') := \bigwedge_{\mathbf{c} \in \mathbb{A}} \mathcal{T}_{\mathbf{c}}(\mathtt{w}_{\mathbf{c}}, \mathbf{a}, \mathtt{w}'_{\mathbf{c}}) \wedge \mathcal{A}(\mathbf{a})$ – encodes the transition relation of the model M. We refer to [24] for more details on the definition of the formula.
- $\mathbb{N}^{\sim}_j(\mathbf{u})$ – encodes that the value j is in the arithmetic relation $\sim \in \{<, >, \leqslant, =, \geqslant\}$ with the value represented by the symbolic number $\mathbf{u}$.
- $\mathcal{S}_{\mathbf{c}_1 \to \mathbf{c}_2}(\mathbf{w}, \mathbf{w}')$ – encodes the social accessibility relation.

Furthermore, we define the j-th symbolic k-path $\boldsymbol{\pi}_j$ as: $(\mathbf{w}_{0,j} \xrightarrow{\mathbf{a}_{1,j}} \mathbf{w}_{1,j} \xrightarrow{\mathbf{a}_{2,j}} \ldots \xrightarrow{\mathbf{a}_{k,j}} \mathbf{w}_{k,j}, \mathbf{u})$, where $\mathbf{w}_{i,j}$ are symbolic states, and $\mathbf{a}_{i,j}$ are symbolic actions, for $0 \leqslant i \leqslant k$ and $1 \leqslant j \leqslant f_k(\alpha)$, and $\mathbf{u}$ is the symbolic number; the function f_k is defined bellow. Moreover, we take $\mathcal{L}^l_k(\boldsymbol{\pi}_n) := \mathbb{N}^{=}_l(\mathbf{u}_n) \wedge H(\mathbf{w}_{k,n}, \mathbf{w}_{l,n})$.

Let $p \in \mathcal{PV}$ and $Y \in \{\overline{\mathrm{K}}_{\mathbf{c}}, \overline{\mathrm{D}}_{\Gamma}, \overline{\mathrm{E}}_{\Gamma}, \overline{\mathcal{O}}_{\mathbf{c}}, \underline{\widehat{\mathrm{K}}}^{\mathbf{c}_2}_{\mathbf{c}_1}, \overline{\mathrm{C}}_{\mathbf{c}_1 \to \mathbf{c}_2}, \overline{\mathrm{D}}_{\mathbf{c}_1 \to \mathbf{c}_2}, \mathrm{E}, \overline{\mathrm{C}}_{\mathbf{c}_1 \to \Gamma}, \overline{\mathrm{D}}_{\mathbf{c}_1 \to \Gamma}\}$. The function $f_k : \mathrm{ECDCTL}^{\star}\mathrm{K} \to \mathbb{N}$ specifies the number of k-paths of the model M that are sufficient to validate an ECDCTL*K formula, and it is defined as follows: $f_k(\mathrm{true}) = f_k(\mathrm{false}) = f_k(p) = f_k(\neg p) = 0$, $f_k(\varphi \wedge \phi) = f_k(\varphi) + f_k(\phi)$, $f_k(\varphi \vee \phi) = max\{f_k(\varphi), f_k(\phi)\}$, $f_k(\mathrm{X}\varphi) = f_k(\varphi)$, $f_k(\varphi \mathrm{U} \phi) = k \cdot f_k(\varphi) + f_k(\phi)$, $f_k(\varphi \mathrm{R} \phi) = (k+1) \cdot f_k(\phi) + f_k(\varphi)$, $f_k(\overline{\mathrm{C}}_{\Gamma}\varphi) = f_k(\varphi) + k$, $f_k(Y\varphi) = f_k(\varphi) + 1$.

The formula $[M^{\alpha,\iota}]_k$, which encodes the unfolding of the transition relation of the model M $f_k(\alpha)$-times to the depth k, is defined as follows:

$$[M^{\alpha,\iota}]_k := \bigvee_{s \in \iota} I_s(\mathbf{w}_{0,0}) \wedge \bigwedge_{j=1}^{f_k(\alpha)} \bigvee_{l=0}^{k} \mathbb{N}^{=}_l(\mathbf{u}_j) \wedge \bigwedge_{j=1}^{f_k(\alpha)} \bigwedge_{i=0}^{k-1} \mathcal{T}(\mathbf{w}_{i,j}, \mathbf{a}_{i,j}, \mathbf{w}_{i+1,j})$$

where $\mathbf{w}_{i,j}$, $\mathbf{a}_{i,j}$, and $\mathbf{u}_j$ are, respectively, symbolic states, symbolic actions, and symbolic numbers, for $0 \leqslant i \leqslant k$ and $1 \leqslant j \leqslant f_k(\alpha)$.

Then, the next step is a translation of an ECDCTL*K state formula α to a propositional formula $[\alpha]_{M,k}$. Observe that since ECDCTL*K is an epistemic, deontic and commitment extension of ECTL*, our definition of $[\alpha]_{M,k}$ agrees with the one presented in [23] on the part where α is an ECTL* formula.

Let α be an ECDCTL*K state formula, $A \subset \mathbb{N}_+$ a set of numbers of symbolic k-paths such that $|A| = f_k(\alpha)$, and $g_s(A)$ denote the set $A \setminus \{min(A)\}$. If $n \in \mathbb{N} - A$ and $0 \leqslant m \leqslant k$, then by $\langle \alpha \rangle^{[m,n,A]}_k$ we denote the translation of an ECDCTL*K state formula α at the symbolic state $\mathbf{w}_{m,n}$ by using the set A. Let φ be an ECDCTL*K

path formula, and $A \subset \mathbb{N}_+$ a set of numbers of symbolic k-paths such that $|A| = f_k(\varphi)$. If $n \in \mathbb{N} - A$ and $0 \leqslant m \leqslant k$, then by $[\varphi]_k^{[m,n,A]}$ we denote the translation of an ECDCTL*K path formula φ along the symbolic k-path $\boldsymbol{\pi}_n$ with starting point m by using the set A.

The translation of ECDCTL*K formulae to a propositional formula is defined inductively with the [23] rules for the ECTL* fragment of ECDCTL*K, and with the following rules for epistemic, deontic and commitment modalities:

- $\langle \overline{\mathrm{K}}_{\mathbf{c}} \alpha \rangle_k^{[m,n,A]} := I_s \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,n',g_s(A)]} \wedge H_{\mathbf{c}}(\mathbf{w}_{m,n}, \mathbf{w}_{j,n'}))$,
- $\langle \overline{\mathrm{D}}_{\Gamma} \alpha \rangle_k^{[m,n,A]} := I_s \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,n',g_s(A)]} \wedge \bigwedge_{\mathbf{c} \in \Gamma} H_{\mathbf{c}}(\mathbf{w}_{m,n}, \mathbf{w}_{j,n'}))$,
- $\langle \overline{\mathrm{E}}_{\Gamma} \alpha \rangle_k^{[m,n,A]} := I_s \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,n',g_s(A)]} \wedge \bigvee_{\mathbf{c} \in \Gamma} H_{\mathbf{c}}(\mathbf{w}_{m,n}, \mathbf{w}_{j,n'}))$,
- $\langle \overline{\mathrm{C}}_{\Gamma} \alpha \rangle_k^{[m,n,A]} := \left\langle \bigvee_{j=1}^{k} (\overline{\mathrm{E}}_{\Gamma})^j \alpha \right\rangle_k^{[m,n,A]}$,
- $\langle \overline{\mathcal{O}}_{\mathbf{c}} \alpha \rangle_k^{[m,n,A]} := I_s \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,n',g_s(A)]} \wedge O_{\mathbf{c}}(\mathbf{w}_{j,n'}))$,
- $\left\langle \widehat{\underline{\mathrm{K}}}_{\mathbf{c}_1}^{\mathbf{c}_2} \alpha \right\rangle_k^{[m,n,A]} := I_s \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,n',g_s(A)]} \wedge \widehat{H}_{\mathbf{c}_1}^{\mathbf{c}_2}(\mathbf{w}_{m,n}, \mathbf{w}_{j,n'}))$,
- $\langle \overline{\mathrm{C}}_{\mathbf{c}_1 \to \mathbf{c}_2} \alpha \rangle_k^{[m,n,A]} := I_s \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,n',g_s(A)]} \wedge \mathcal{S}_{\mathbf{c}_1 \to \mathbf{c}_2}(\mathbf{w}_{m,n}, \mathbf{w}_{j,n'}))$,
- $\langle \overline{\mathrm{C}}_{\mathbf{c}_1 \to \Gamma} \alpha \rangle_k^{[m,n,A]} := I_s \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,n',g_s(A)]} \wedge \bigwedge_{\mathbf{c}_2 \in \Gamma} \mathcal{S}_{\mathbf{c}_1 \to \mathbf{c}_2}(\mathbf{w}_{m,n}, \mathbf{w}_{j,n'}))$,
- $\langle \overline{\mathrm{D}}_{\mathbf{c}_1 \to \mathbf{c}_2} \alpha \rangle_k^{[m,n,A]} := I_s \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,n',g_s(A)]} \wedge O_{\mathbf{c}_2}(\mathbf{w}_{j,n'}) \wedge \mathcal{S}_{\mathbf{c}_1 \to \mathbf{c}_2}(\mathbf{w}_{m,n}, \mathbf{w}_{j,n'}))$,
- $\langle \overline{\mathrm{D}}_{\mathbf{c}_1 \to \Gamma} \alpha \rangle_k^{[m,n,A]} := I_s \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,n',g_s(A)]} \wedge \bigwedge_{\mathbf{c}_2 \in \Gamma} (\mathcal{S}_{\mathbf{c}_1 \to \mathbf{c}_2}(\mathbf{w}_{m,n}, \mathbf{w}_{j,n'}) \wedge O_{\mathbf{c}_2}(\mathbf{w}_{j,n'}))$,

where $n' = min(A)$, and I_s denotes the formula $\bigvee_{s \in \iota} I_s(\mathbf{w}_{0,min(A)})$.

Now let α be an ECDCTL*K state formula. Then $[\alpha]_{M,k} := \langle \alpha \rangle_k^{[0,0,F_k(\alpha)]}$, where $F_k(\alpha) = \{j \in \mathbb{N} \mid 1 \leqslant j \leqslant f_k(\alpha)\}$.

Theorem 2. *Let M be a model and α be an* ECDCTL*K *state formula. Then for every $k \in \mathbb{N}$ and for every $s \in \iota$, $M, s \models_k \alpha$ if, and only if, the propositional formula $[M, \alpha]_k$ is satisfiable.*

4 Example – the NB protocol

The NetBill (NB) protocol [17] is an electronic commerce protocol designed to be used for the selling and delivery of low-priced information goods over the Internet. The NB transaction model involves three agents: the customer (Cus), the merchant (Mer) and the NetBill transaction server (NB). A transaction involves three phases: price negotiation, goods delivery, and payment. The basic protocol consists of eight steps:

1. $Cus \Rightarrow Mer$: Price request
2. $Mer \Rightarrow Cus$: Price quote
3. $Cus \Rightarrow Mer$: Goods request
4. $Mer \Rightarrow Cus$: Goods, encrypted with a key K
5. $Cus \Rightarrow Mer$: Signed electronic payment order (EPO)
6. $Mer \Rightarrow NB$: Endorsed EPO (including K)
7. $NB \Rightarrow Mer$: Signed result (including K)
8. $Mer \Rightarrow Cus$: Signed result (including K)

In this section we are interested in applying our SAT-based BMC method for CDIS to verify a version of the NB protocol, where Mer does not operate as it is supposed to. Specifically, we consider the possibility that Mer may send the receipt without delivering the goods.

Modelling the NB protocol. We used the model $M = (S, \iota, T, \mathcal{V}, \sim_{\mathbf{c}}, \bowtie_{\mathbf{c}}, \sim_{\mathbf{c}_1 \to \mathbf{c}_2})$, that is shown in Fig. 1 to model the behaviour of the violated NB protocol. The associated $CDIS = (\{\iota_{\mathbf{c}}, L_{\mathbf{c}}, \|\Phi\|_{\mathbf{c}}, Var_{\mathbf{c}}, Act_{\mathbf{c}}, P_{\mathbf{c}}, t_{\mathbf{c}}, \mathcal{V}_{\mathbf{c}}\}_{\mathbf{c} \in \mathbb{A}})$ is defined in the next paragraph. The NB protocol begins at s_0 with a customer (Cus) requesting a price for some desired goods (e.g. journal articles). This request is followed by the merchant (Mer) reply with sending an offer (the price quote), which means creating a commitment. Cus can either reject this offer, which means releasing this offer at s_7, or accept this offer, which means creating a commitment at s_3. Cus's commitment means that if Mer delivers the requested goods, then he is willing to pay for the goods. At this state, Cus has two possibilities: to withdraw his commitment at s_7 or to delegate it to his bank to pay Mer on his behalf. When Cus pays for the requested goods Cus has two possibilities: either Mer delivers goods or not. If Mer delivers goods at s_5 (i.e., Mer fulfills his commitment), then he sends the receipt to Cus at s_6. If Mer withdraw his offer, then Mer violates his commitment at s_8 and either moves to the failure state s_9 after refunding the payment to Cus or moves to the state s_6 after sending erroneously the receipt to Cus. This protocol can be extended to any number n of agents greater than two.

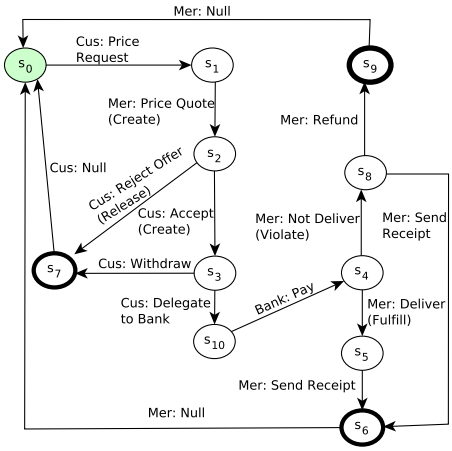


Fig. 1. The model of NB; the solid lines refer to the temporal relation; the social accessibility relation is: $\sim_{Cus \to Mer} = \{(s_0, s_0), (s_1, s_1), (s_2, s_2), (s_5, s_5), (s_6, s_6), (s_7, s_7), (s_8, s_8), (s_9, s_9), (s_3, s_4), \}$, $\sim_{Mer \to Cus} = \{(s_0, s_0), (s_1, s_1), (s_2, s_2), (s_5, s_5), (s_6, s_6), (s_7, s_7), (s_8, s_8), (s_9, s_9), (s_4, s_5), \}$.

In the CDIS setting the NB protocol involves three agents: the merchant (Mer) and the customer (Cus), and the NetBill transaction server (NB). Each agent of the protocol can be modelled by considering its set of local states, set of local initial state, set of green states, finite set of local integer variables, set of local actions, local protocol, local evolution function, and local valuation function.

For Cus, it is enough to consider 11 possible local states. Since we are not admitting the possibility of faults, its local states are all green. Thus, w have: $L_{Cus} = \{c_0, \ldots, c_{10}\}$, $\mathcal{R}_{Cus} = \emptyset$, and $\|\Phi\|_{Cus} = L_{Cus}$. Further, $\iota_{Cus} = \{c_0\}$. For Mer, it is also enough to consider 11 possible local states, i.e., $L_{Mer} = \{m_0, \ldots, m_{10}\}$, and $\iota_{Mer} = \{m_0\}$.

Since we assume that it is possible that a faulty receipt was sent before the goods was delivered, we classify the local states of Mer as follow: $\|\Phi\|_{Mer} = L_{Mer} \setminus \{m_8\}$ and $\mathcal{R}_{Mer} = \{m_8\}$. For NB, to simplify the presentation, we shall to consider just one local state: $L_{NB} = \{\cdot\} = \iota_{NB}$. Moreover, we assume that $L_{NB} = \|\Phi\|_{NB}$, $\mathcal{R}_{NB} = \emptyset$. Now we can define the set of possible global states S for the scenario as the product $L_{Cus} \times L_{Mer} \times L_{NB}$, and we consider the following set of initial states $\iota = \{(c_0, m_0, \cdot)\}$. The set of boolean (integer) variables available to the agents are as follows: $Var_{Cus} = \{x_1, x_2, x_3, x_4\}$, $Var_{Mer} = \{x_1, x_2, x_5, x_6\}$, $Var_{NB} = \{x_3, x_4, x_5, x_6\}$. The set of actions available to the agents are as follows:
Act_{Cus} = {$SendPriceRequest$, $ReceivePriceQuote$, $SendAcceptOffer$, $ReceiveReceipt$, $ReceiveRefund$, $SendRejectOffer$, $SendWithdraw$, $SendDelegateToBank$, $ReceiveGoods$, $ReceiveNoGoods$, End, λ},
Act_{Mer} = {$ReceivePriceRequest$, $SendPriceQuote$, $SendNotDeliver$, $ReceiveAcceptOffer$, $ReceiveRejectOffer$, $ReceiveWithdrawOffer$, $ReceiveDelegateToBank$, $SendDeliver$, $SendRefund$, $ReceivePay$, $SendReceipt$,End, λ}, $Act_{NB} = \{SendPay, \lambda\}$, where λ stands for no action. The local protocols of the agents are the following:

- $P_{Cus}(c_0) = \{SendPriceRequest\}$, $P_{Cus}(c_1) = \{ReceivePriceQuote\}$,
  $P_{Cus}(c_2) = \{SendAcceptOffer, SendRejectOffer\}$,
  $P_{Cus}(c_3) = \{SendWithdraw, SendDelegateToBank\}$,
  $P_{Cus}(c_4) = \{ReceiveNoGoods, ReceiveGoods\}$, $P_{Cus}(c_5) = \{ReceiveReceipt\}$,
  $P_{Cus}(c_6) = \{\lambda\}$, $P_{Cus}(c_7) = \{End\}$,
  $P_{Cus}(c_8) = \{ReceiveReceipt, ReceiveRefund\}$,
  $P_{Cus}(c_9) = \{\lambda\}$, $P_{Cus}(c_{10}) = \{\lambda\}$.
- $P_{Mer}(m_0) = \{ReceivePriceRequest\}$, $P_{Mer}(m_1) = \{SendPriceQuote\}$,
  $P_{Mer}(m_2) = \{ReceiveAcceptOffer, ReceiveRejectOffer\}$,
  $P_{Mer}(m_3) = \{ReceiveWithdrawOffer, ReceiveDelegateToBank\}$,
  $P_{Mer}(m_4) = \{SendDeliver, SendNotDeliver\}$,
  $P_{Mer}(m_5) = \{SendReceipt\}$, $P_{Mer}(m_6) = \{End\}$, $P_{Mer}(m_7) = \{\lambda\}$,
  $P_{Mer}(m_8) = \{SendReceipt, SendRefund\}$, $P_{Mer}(m_9) = \{End\}$,
  $P_{Mer}(m_{10}) = \{ReceivePay\}$.
- $P_{NB}(\cdot) = Act_{NB}$.

Given Fig. 1 and the above protocol functions, it should be straightforward to infer the local evolution function of the agents. Furthermore, in the Kripke model of the NB protocol, we assume the following set of proposition variables: $\mathcal{PV} = \{Pay, Deliver, Price, Accept\}$ with the following interpretation:
$(M, s) \models Pay$ if $l_{Cus}(s) = c_4$, $(M, s) \models Accept$ if $l_{Cus}(s) = c_3$,
$(M, s) \models Deliver$ if $l_{Mer}(s) = m_5$, $(M, s) \models Price$ if $l_{Mer}(s) = m_2$.

Some temporal, deontic and social properties we may be interested in checking for the example above are the following (we express them in the universal language, but we verify their negations, i.e., existential properties):

1. $\varphi_1 = \mathrm{AG}(Pay \rightarrow \mathrm{F}Deliver)$ – Cus pay for the goods, then the goods will eventually be delivered.
2. $\varphi_2 = \mathrm{AG}(\mathrm{C}_{Cus \rightarrow Mer} Accept \rightarrow \mathrm{F}Pay)$ - whenever Cus commits towards Mer that he accept the offer, then Cus will eventually pay for the offer.
3. $\varphi_3 = \mathrm{AG}(\mathrm{D}_{Cus \rightarrow Mer} Pay \rightarrow \mathrm{F}Deliver)$ - whenever Cus commits towards Mer under the assumption that Mer is functioning correctly that he pays for the goods then Cus will eventually receive the goods.

Having the above description of the NB protocol, we can easily infer propositional formulae that encode both the model and all the properties mentioned above. Further,

checking that the NB satisfies the properties 1–3 can now be done by feeding a SAT solver with the propositional formulae generated in the way explained above.

5 Conclusions

We proposed the SAT-based BMC for ECDCTL*K and for CDIS. The BMC of the CDIS may also be performed by means of Ordered Binary Diagrams (OBDD). This will be explored in the future. Moreover, our future work include an implementation of the algorithm presented here, a careful evaluation of experimental results to be obtained, and a comparison of the OBDD- and SAT-based BMC method for CDIS.

References

1. J. Bentahar, J. Ch. Meyer, and W. Wan. *Specification and Verification of Multi-Agent Systems*, chapter Model Checking Agent Communication, pp. 67–102. Springer, 2010.
2. A. Chopra, N. Desai, and M. Singh. Amoeba: a methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology*, 19(2):1–40, 2009.
3. E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.
4. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.
5. N. Desai, Z. Cheng, A.K. Chopra, and M.P. Singh. Toward verification of commitment protocols and their compositions. In *Proc. of AAMAS 2007*, pp. 144–146. IFAAMAS, 2007.
6. M. El-Menshawy, J. Bentahar, and R. Dssouli. Verifiable semantic model for agent interactions using social commitments. In *Proc. of LADS 2009*, volume 6039 of *LNAI*, pp. 128–152. Springer-Verlag, 2010.
7. M. El-Menshawy, J. Bentahar, W. El Kholy, and R. Dssouli. Reducing model checking commitments for agent communication to model checking arctl and GCTL*. *Autonomous Agents and Multi-Agent Systems*, 27(3):375–418, 2013.
8. E. A. Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science*, volume B, chapter 16, pp. 996–1071. Elsevier, 1990.
9. E. A. Emerson and J. Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of ACM*, 33(1):151–178, 1986.
10. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
11. A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75(1):63–92, 2003.
12. X. Luo, K. Su, A. Sattar, and M. Reynolds. Verification of multi-agent systems via bounded model checking. In *Proc. of AI 2006*, volume 4304 of *LNAI*, pp. 69–78. Springer-Verlag, 2006.
13. A. Męski, W. Penczek, M. Szreter, B. Woźna-Szcześniak, and A. Zbrzezny. BDD-versus SAT-based bounded model checking for the existential fragment of linear temporal logic with knowledge: algorithms and their performance. *Autonomous Agents and Multi-Agent Systems*, 28(4):558–604, 2014.
14. M. El Menshawy, J. Benthar, H. Qu, and R. Dssouli. On the verification of social commitments and time. In *Proc. of AAMAS 2011*, pp. 483–490. IFAAMAS, 2011.
15. W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

16. M. P. Singh. A social semantics for agent communication languages. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pp. 31–45. Springer-Verlag, 2000.
17. Marvin A. Sirbu. Credits and debits on the internet. *IEEE Spectrum*, 34(2):23–29, 1997.
18. M. Venkatraman and M. P. Singh. Verifying compliance with commitment protocols: Enabling open web-based multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, 1999.
19. M. Wooldridge. *An introduction to multi-agent systems – Second Edition.* John Wiley & Sons, 2009.
20. B. Woźna-Szcześniak and A. Zbrzezny. Sat-based bounded model checking for deontic interleaved interpreted systems. In *Proc. of KES-AMSTA 2012*, volume 7327 of *LNCS*, pp. 494–503. Springer-Verlag, 2012.
21. B. Woźna-Szcześniak, A. M. Zbrzezny, and A. Zbrzezny. Sat-based bounded model checking for weighted interpreted systems and weighted linear temporal logic. In *Proc. of PRIMA 2013*, volume 8291 of *LNAI*, pp. 355–371. Springer-Verlag, 2013.
22. L. Wu, J. Su, K. Su, X. Luo, and Z. Yang. A concurrent dynamic logic of knowledge, belief and certainty for multi-agent systems. *Knowledge-Based Systems*, 23(2):162–168, 2010.
23. A. Zbrzezny. A new translation from ECTL* to SAT. *Fundamenta Informaticae*, 120(3-4):377–397, 2012.
24. A. Zbrzezny. On boolean encodings of transition relation for parallel compositions of transition systems. In *Proc. of CS&P 2013*, volume 1032 of *CEUR Workshop Proceedings*, pp. 478–489, 2013.

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Action Rules Mining in Laryngological Disorders

Agnieszka Dardzińska-Głębocka[1], Bożena Kosztyła-Hojna[2], and Anna Łobaczuk-Sitnik[2,3]

[1] Bialystok University of Technology
Dept. of Mechanics and Computer Science
Wiejska 45c, 15-351 Białystok, Poland
[2] Medical University of Białystok
Department of Clinical Phonoaudiology and Logopedics
Szpitalna 37, 15-295 Białystok, Poland
[3] Laryngologic Medical Centre, LAR-COM, Józefa Bema 2, 15-369 Bialystok, Poland
a.dardzinska@pb.edu.pl, fono@umb.edu.pl

Abstract. Action rule is an implication rule that shows the expected change in a decision value of an object as

1 Introduction

Recently we can observe very dynamic growth in the field of computer-based medical systems. It resulted from noticeable improvements in medical care, from ease of storage and access of digital imaging through gathering of computerized medical data to accessing on-line literature, patient monitoring, observing online surgeries, or computer support for most medical diagnosis. Similarly to other domains, decision-support systems have proven to be valuable tools that help professionals in facing challenging medical problems, such as diagnosis and therapy. The etiology of dysphonia is very diverse and rarely monocausal. For example, in the course of allergic rhinitis occurs edema of nasal, sinuses and larynx mucosa which leads to frequent inflammations of these structures, and even organic changes of the larynx: reincke edema, laryngeal polyps, chronic hypertrophic laryngitis and voice nodules [8]. There is strong association between gastroesophageal reflux and pharyngolaryngeal reflux as factors leading to respiratory disease, manifested as dysphonia, wheezing, coughing, recurrent laryngitis [21], [22], [23]. Given that stakes happen to be extremely high, support of decision-making plays an extremely wide role in the field of medicine. Due to a number of diseases which symptom is hoarseness is possible to make an incorrect diagnosis (misdiagnosis). Incorrect diagnosis leads to treatment that means waste of many resources, time and sometimes even of human life. Automating those processes that can be captured by focus on characteristic symptoms and signs, minimalize human error and leads to overall improvements in the quality of medical care. The primary purpose of this information system is to improve diagnostic accuracy for hoarseness (dysphonia), reduce inappropriate antibiotic use, reduce inappropriate steroid use, reduce inappropriate use of anti-reflux medications, reduce inappropriate use of radiographic imaging, and promote appropriate use of laryngoscopy, voice therapy, and surgery – according to clinical practice guideline [19], [21].

2 Larynx disorders as suggested area for computer-aided diagnostics

The domain of larynx diseases is especially suited to application of computer-based methods and there are several reasons for high expectations from computer-aided diagnosis. Firstly, the number of cases of throat and larynx cancers is on the rise. Secondly, correct diagnosis, especially in early stages of disease, is difficult. There is variety of diseases that manifests with similar symptoms. Finally, early diagnosis is critical, as in some cases damage to the vocal cords caused by an untreated disorder may by irreversible. Typically, a patient suffering from symptoms suggesting of hoarseness seeks help of a family doctor. Primary care physicians face the daunting task of determining the source of discomfort connected with abnormal sound, based on patient-reported data and physical examination, possibly enhanced with the results of basic medical tests. According to Ruitz et al. [18], many physician are willing to empirically prescribe reflux medication as primary therapy to patients with chronic hoarseness of unknown origin, even when symptoms of gastroesophageal reflux disease are not present [19]. According to Traister et al., vocal cord dysfunction is often misdiagnosed and mistreated as asthma, which can lead to increased and unnecessary medication use and increased health care utilization [21].

Correct diagnosis under these circumstances is not easy and accuracy can be low. This rather low diagnostic performance is caused by several etiological and organizational factors. These include the nature of the larynx. Numerous conditions can cause hoarseness, ranging from simple inflammatory processes to more serious systemic, neurologic, or cancerous conditions involving the larynx. Evaluation of a patient with hoarseness includes a careful history, physical examination, and in many cases, laryngoscopy [3], [9]. Any patient with hoarseness lasting longer than two weeks in the absence of an apparent benign cause requires a thorough evaluation of the larynx by direct or indirect laryngoscopy. Visualization of vocal fold vibration is essential for assessing the causes of voice quality disorders, the so-called dysphonia. In clinical practice endoscopic and stroboscopic laryngeal methods are widely used [5], [7]. Detailed information related to vocal folds is obtained through application of the high-speed digital video recording method [5], [6]. The management of hoarseness includes identification and treatment of any underlying conditions, vocal hygiene, voice therapy, and specific treatment of vocal cord lesions. Referral for surgical or other targeted interventions is indicated when conservative management of vocal cord pathology is unsuccessful, when dysplasia or carcinoma is suspected, or when significant airway obstruction is present.

3 New computer supported diagnosis system

The system is creating in collaboration between the Department of Mechanics and Computer Science at Bialystok University of Technology and physicians at the Department of Clinical Phonoaudiology and Logopedics at Medical University of Bialystok, and Laryngologic Medical Centre. The system allows for systematic collection and processing of clinical data of hoarseness patients diagnosed and treated in the medical centre. The system is equipped with a diagnostic module built around several statistical methods that have been found useful in diagnosis. An integral part of the system is database, created in 90s and thoroughly maintained. The current database contains approximately 500 patients records – with the ultimate diagnosis verified by means of blood tests, laryngoscopy, CT scan, biopsy, x-rays. Each case is described by over 30

different medical findings, such as patient self-reported data, results of physical examination, laboratory tests and finally a histopatologically verified diagnosis. The system includes the module supporting the physicians in making a diagnosis and it can be treated as database system because decision rules used there relay mainly on information extracted from the database. The patients from this database have been classified by clinicians into several adequate throat and larynx diseases. The support of diagnosis in the system is based on a comparison of a new patient case with similar cases from its database. This paper presents how new ideas with computer-aided diagnosis can improve typical medical diagnosis of patients with throat and larynx diseases, especially with hoarseness.

4 Action rules mining

Finding useful rules is an important task of a knowledge discovery process. Most researchers focus on techniques for generating classification or association rules. They assume that it is user's responsibility to analyze the connections in order to infer solutions for specific problems within a given domain. The classical knowledge discovery algorithms have the potential to identify enormous number of significant patterns from data. But at the same time people are overwhelmed by a large number of uninteresting patterns and it is very difficult for a human being to analyze them, because of the huge time consuming tasks. Therefore, a need for new methods with the ability to assist users in analyzing a large number of rules for a useful knowledge [1] is seeking.

An action rule is a rule extracted from a decision system that describes a possible transition of objects from one state to another with respect to a distinguished attribute called a decision attribute [11]. We assume that attributes used to describe objects in a decision system are partitioned into stable and flexible. Values of flexible attributes can be changed. This change can be influenced and controlled by users. Action rules mining initially was based on comparing profiles of two groups of targeted objects – those that are desirable and those that are undesirable [1], [2], [11], [12].

An action rule was defined as a term

$$r = [\omega * (\alpha \to \beta)] \to (\varphi \to \psi)],$$

where $\omega, \alpha, \beta, \varphi, \psi$ are descriptions of objects, in our case seen as patients. The term r states that when a fixed condition ω is satisfied and the changeable behavior $(\alpha \to \beta)$ occurs in patients registered in a database so does the expectation $(\varphi \to \psi)$. This paper proposes a method for constructing action rules directly from single classification rules. This method (ARAS algorithm) has been implemented as one of the modules in new computer aided support system, which consists of a medical database and a set of tests and procedures, facilitating decision-making process for patients with throat and larynx disorders.

Action rules introduced in [14] has been further investigated in [1], [2], [16], [11], [12], [13]. Paper [12] was the first attempt towards formally introducing the problem of mining action rules without pre-existing classification rules. Authors explicitly formulated it as a search problem in a support-confidence-cost framework. The algorithm proposed by them has some similarity with Apriori [10]. The definition of an action rule in [12] allows changes on flexible attributes. However changing the value of an attribute, is directly linked with a cost [13]. In this paper we propose a very simple LERS-type algorithm for constructing action rules from a single classification

rule. LERS is a classical example of a bottom-up strategy which constructs rules with a conditional part of the length $k+1$ after all rules with a conditional part of the length k have been constructed [4]. Relations representing rules produced by LERS are marked. System ARAS assumes that LERS is used to extract classification rules. This way system ARAS has to check if these relations are marked by LERS. The same, if we use LERS as a pre-processing module for ARAS, then the overall complexity of the algorithm is decreased [11], [12]. An action rule can be constructed in ARAS from two classification rules only if both of them belong to the same cluster and one of them is a target classification rule.

5 Action rules in support diagnosis medical system

Tested database contains information about 500 patients described by 28 attributes (including 14 laboratory tests with values discretized to: below normal, normal, above normal). It has 10 stable attributes.

The decision attribute has the following values:
A – normal,
B – vocal cord polyps,
$C1$ – voice nodules,
$C2$ – voice nodules and reflux disease,
D – functional dysphonia,
E – larynx-cancer.

The diagnosis of larynx disease depends on a combination of patient's history, physical examinations, laboratory tests, radiological tests, and frequently a larynx biopsy. Blood tests should be analysed along with the patient's history and physical examination. A medical treatment is naturally associated with reclassification of patients from one decision class into another one. In our research we are mainly interested in the reclassification of patients from the class $C2$ into class $C1$ and from class $C1$ to class A. Obviously database in our system has many missing values so we decided to remove all its attributes with more than 0.80 of null values. We do the same also with subjective attributes (like history of smoking). We used classical null value imputation techniques based on ERID algorithm [1] and applied RSES software to find d-reducts [1], [2]. For the testing purpose, we have chosen the same d-reduct. By d-reduct we mean a minimal subset of attributes which by itself can fully characterize the knowledge about attribute d in the database. The description of attributes (tests) listed in system R is given below:

$$R = \{a, g, ii, sa, ya, is, h, sp, hh, bi, vi, ad, ph, vw, sd\},$$

where
a – age,
g – gender,
ii – inhalation injuries,
sa – seasonal allergy,
ya – year-round allergy,
is – irritating substances,
h – hoarseness,
sp – surgeries in the past,
hh – history of hospitalization,

bi – bacterial infection,
vi – viral infection,
ad – autoimmune diseases,
ph – pH level,
vw – voice work,
sd – special diet.

Many action rules have been discovered. Two of them with high confidence (close to 90) are given below:

$$[(hh,1)*(g,2)*(h,1)]*(sd,2\rightarrow 1)*(ph,2\rightarrow 1)\rightarrow (d,C2\rightarrow C1).$$

The first rule is applicable to women with a history of hospitalization. It says that if we change the diet into special one and we change the level of ph, then we should be able to reclassify such patients from the category $C2$ to $C1$.

$$[(hh,1)*(g,1)]*(vw,2\rightarrow 1)*(sa,2\rightarrow 1)\rightarrow (dd,C1\rightarrow A1).$$

The second rule is applicable to men with history of hospitalization. It says that if we minimize seasonal allergy and we minimize voice expose then we should be able to reclassify such patients from the category $C1$ to $A1$.

6 Conclusion

Action rules mining can be successfully applied in other medical databases. We previously built system to support flat feet treatment [1]. In this model authors suggested that the arch height correction is increased by age and place of living, and decreased as body mass increased. Therefore changing body mass is one of the main purposes to obtain promising results.

The paper presents preliminary results which finally will lead us to the full construction of diagnostic system for improving fast throat disorders diagnoses. In future the authors will construct a flexible temporal feature retrieval system based on grouping patients of similar visiting frequencies with connection to an action-rules engine, which will consist of four modules: a data grouping device, a temporal feature extraction engine, a decision tree classification device, and an action rules generation device. The data grouping device is to filter out less relevant records in terms of visiting duration patterns measured from the initial visits. The temporal feature extraction engine is to project temporal information into patient-based records for classic classifiers to learn effects of treatment upon patients. The decision tree classification device is to generate classification rules. The action rules generation device is to build action rules from certain pairs of classification rules.

References

1. A. Dardzinska (2013), Action Rules Mining, Studies in Computational Intelligence, vol. 468, Springer
2. A. Dardzinska, Z. Ras (2006), Extracting rules from incomplete decision systems, Foundations and Novel Approaches in Data Mining, Studies in Computational Intelligence, Springer, Vol. 9, 143–154

3. R. Feierabend, M. Shahram (2009), Hoarseness in adults, American Family Physician, 80(4), 363–70
4. J. Grzymala-Busse, (1997), A new version of the rule induction system LERS, Fundamenta Informaticae, IOS Press, Vol. 31, No. 1, 27–39
5. B. Kosztyla-Hojna, D. Moskal, D. Falkowski, J. Othman, A. Lobaczuk-Sitnik (2013), The innovative method of visualization of vocal folds vibrations in the chosen cases of occupational dysphonia, Otorynolaryngologia – Przegląd Kliniczny Vol. 12, 23
6. B. Kosztyla-Hojna, D. Moskal, D. Falkowski (2013) Ocena przydatności metody szybkiego filmu highspeed imaging (HSI) w diagnostyce zaburzeń jakości głosu, Ogólnopolskie Sympozjum Logopedyczne Metodologia badań logopedycznych, 1–2 (in Polish)
7. B. Kosztyla-Hojna, B. Rogowski, I. Rzewnicki, R. Rutkowski, A. Lobaczuk-Sitnik, A. Lachowicz (2007) Usefulness of some diagnostic methods in differential diagnosis of occupational voice disorders, Polish Journal of Environmental Studies, Vol. 16 no 1A, 23–29
8. B. Kosztyla-Hojna, B. Poludniewska, M. Tupalska, W. Mikiel (1997), Zaburzenia głosu u chorych z alergicznym nieżytem nosa, Otolaryngologia Polska, Vol. 51–2, 191–199 (in Polish)
9. T. Mau (2010), Diagnostic evaluation and management of hoarseness, The Medical Clinics of North America, 945–960
10. Z. Pawlak (1981), Information systems – theoretical foundations, Information Systems Journal, Vol. 6, 205–218
11. Z. Ras, A. Dardzinska (2006), Action rules discovery, a new simplified strategy, Foundations of Intelligent Systems, Proceedings of ISMIS'06 Symposium, Springer, LNAI, Vol. 4203, 445–453
12. Z. Ras, A. Dardzinska (2008), Action rules discovery without pre-existing classification rules, Proceedings of the International Conference on Rough Sets and Current Trends in Computing, Springer, LNAI, Vol. 5306, 181–190
13. Z. Ras, A. Dardzinska, L.S. Tsay, H. Wasyluk (2008), Association Action Rules, IEEE/ICDM Workshop on Mining Complex Data, Pisa, Italy, ICDM Workshops Proceedings, IEEE Computer Society, 283–290
14. Z. Ras, A. Wieczorkowska (2000), Action-rules: how to increase profit of a company, in Principles of Data Mining and Knowledge Discovery, Proceedings of PKDD'00, Lyon, France, LNAI, No. 1910, Springer, 587–592
15. Z. Ras, E. Wyrzykowska, H. Wasyluk (2008), ARAS: Action rules discovery based on agglomerative strategy, in Mining Complex Data, Post-Proceedings of 2007 ECML/PKDD Third International Workshop (MCD 2007), LNAI, Vol. 4944, Springer, 196–208
16. J. Rauch, M. Simunek (2009), Action rules and the GUHA method: Preliminary considerations and results, in Proceedings of ISMIS 2009, LNAI 5722, Springer, 76–87
17. J.S. Rubin, R.T. Sataloff, G. Korovin (eds.) (2006), Diagnosis and treatment of voice disorders, Plural Publishing Inc. Sandiego Oxford
18. R.Ruiz, S. Jeswani, K. Andrews, B. Rafii, B.C. Paul, R.C. Branski, M.R. Amin (2014), Hoarseness and laryngopharyngeal reflux: a survey of primary care physician practice patterns, Vol. 140(3): 192–196
19. S.R. Schwartz, S.M. Cohen, S.H. Dailey, R.M. Rosenfeld, E.S. Deutsch, M.B. Gillespie, E. Granieri, E.R. Hapner, C.E. Kimball, H.J. Krouse, J.S. McMurray, S. Medina, K. O'Brien, D.R. Ouellette, B.J. Messinger-Rapport, R.J. Stachler, S. Strode,

D.M. Thompson, J.C. Stemple, J.P. Willging, T. Cowley, S. McCoy, P.G. Bernad, M.M Patel (2009), Clinical practice guideline: hoarseness (dysphonia), Otolaryngol Head Neck Surg. Vol. 141 (3 Suppl 2):S1–S31.

20. D. Skrodzka, U. Wereszczynska-Siemiatkowska, B. Poludniewska, J.B. Kasprowicz (2006), Powikłania laryngologiczne choroby refluksowej przełyku, Przegląd Lekarski, Vol. 63, 9, 752–755 (in Polish)
21. R.S. Traister, M.L. Fajt, D. Landsittel, A.A. Petrov (2014), A novel scoring system to distinguish vocal cord dysfunction from asthma, J Allergy Clin Immunol Pract. Vol. 2(1):65-69
22. M. Zalesska-Krecicka, D. Wasko-Czopnik, T. Krecicki, L. Paradowski, M. Zatonski, M. Horobiowska (2003), Objawy laryngologiczne u chorych z chorobą refluksową przełyku, Otolaryngologia Polska, Vol. 57, 6, 819–822 (in Polish)
23. H. Zielinska-Blizniewska, K. Buczylko, J. Olszewski (2011), Diagnostyka alergologiczno-laryngologiczna w chorobie refluksowej przełyku u dorosłych, Alergoprofilaktyka, Vol. 7, 2, 15–23 (in Polish)

Trends in Contemporary Computer Science, Podlasie 2014, ISBN 978-83-62582-58-7
by A. Gomolińska, A. Grabowski, M. Hryniewicka, M. Kacprzak, E. Schmeidel (Eds.)

Similarity-based Searching in Structured Spaces of Music Information ⋆

Mariusz Rybnik[1] and Władysław Homenda[2]

[1] Faculty of Mathematics and Computer Science, University of Bialystok,
ul. Sosnowa 64, 15-887 Białystok, Poland, MariuszRybnik@wp.pl
[2] Faculty of Mathematics and Information Science, Warsaw University of Technology,
Plac Politechniki 1, 00-660 Warsaw, Poland

Abstract. The study is focused on structural analysis of musical pieces, based on specialized grammars covering music information. The analysis may be performed on a single musical piece, where one searches for leitmotifs. A comparative analysis may also be performed, where a comparison of rhythmic and melodic motives is performed. Rhythmic motives are regarded here as patterns of notes' durations of various length, starting at length one, that allows for a simple distribution of notes length for a piece. The operators allow to relativize rhythmic and melodic motives of length 2 and more so they became more universal. In this paper we focus on searching for transformed and non-transformed motives, analysing melodic and rhythmic sequences, structure discovery and comparative analysis of musical pieces.
Keywords: music information, searching, knowledge discovery, knowledge understanding, syntactic structuring, querying.

1 Introduction

In this paper we analyze structured spaces of music information. The study is performed on Western-style music mostly, however it is possible to apply the same methodic for another music-styles, as long as they are based on a specific scale.

The study is focused on structural analysis of musical pieces [1], [4], [5], based on operators proposed in [7]. The analysis may be performed on a single musical piece, where one searches for lait-motives. A comparative analysis may be performed, regarding rhythmic and melodic motives [6]. The study is based on proposed operators, that serve for extensive searching related to querying as detailed in the authors' previous work: [8]. The idea is implemented basically on standard paginal music notation represented in Music XML form [2], but may be adopted to other formats of describing music information, like Braille music description [10], MIDI [11] etc. The subject could be seen as a continuation of the new issue of "automatic image understanding" raised by R. Tadeusiewicz a few years ago, c.f. [12] [13].

Section 2 presents the proposed simplified *searching-oriented grammar* and three operators for detection of melodic and rhythmic transformations. Section 3 discuses advanced searching in spaces of music information with the use of proposed operators

⋆ This work is supported by The National Center for Research and Development, Grant no N R02 0019 06/2009.

and grammar. Section 4 presents methodology and examples for searching for leitmotifs, rhythmic analysis and melodic analysis. Finally, section 5 concludes the paper and proposes future work.

2 Searching-oriented grammar

In paper [7] we simplified the *graphically oriented grammar* proposed in our previous work [8] for *paginated music notation* (mostly of graphical characteristic). The proposed grammar may be seen as *searching-oriented grammar*. The searching regards melodic and rhythmic dependencies, typical to music spaces of information, and therefore disregard any graphical aspects of music notation.

For this purpose we propose to omit most graphical components typical to *paginated music notation* (e.g. page, stave, barline, clef). Searching is much easier in continuous *notes* space, with no artificial divisions, therefore traditionally fundamental *<measure>* tag is also disregarded. For the same reason we propose to convert indispensable music notation symbols connected with *stave* or *measures* into granular *note* attributes and disregard the unnecessary ones.

In order to simplify searching in space of music information three new operators were proposed, described in detail in the sections below. We propose to append to *note* the values produced by the operators as additional properties useful in searching.

2.1 Rhythm ratio operator

In order to efficiently process searching for rhythm-transformed patterns we propose an operator designated **rhythm ratio** (q_{rh}) defined as the *ratio* of the current note duration to the previous note duration (for example: 16th preceded by 8th would have q_{rh} value of 0.5). We consider rests as intrinsic element of rhythm, therefore they are processed in the same way, except that the ratios values for rests are negative. That allows for easy differentiation in between notes and rests, as q_{rh} for notes are always positive. The q_{rh} is defined for all notes/rests in the particular voice except for the first one.

2.2 Scalar difference operator

We introduce an operator designated **scalar difference** (d_s) that defines the difference of *the current note* to *the previous note*, expressed in diatonic scale degrees. The d_s is defined for all notes in particular voice except for the first one, providing that the key (scale) is defined. Please note that the d_s value could be also a fraction for *accidental note* (a note outside of defined scale). For this operator rests are ignored, as arbitrarily they do not contribute to the melody. Operator d_s is similar to *melodic diatonic interval* in music theory (but more convenient).

2.3 Halftone difference operator

Similarly we introduce an operator designated **halftone difference** ($d_{1/2}$) that defines the difference of *the current note* to *the previous note*, expressed in halftones. The $d_{1/2}$ would be defined for all notes in particular voice except for the first one. For this operator rests are ignored, as arbitrarily they do not contribute to the melody. Operator $d_{1/2}$ is very similar to *melodic enharmonic interval* in music theory.

2.4 Exemplary values of the three operators

The exemplary values of the three operators are depicted in Fig. 1. Boxes represent identical melodic patterns (3 notes) for $d_{1/2}$ and d_s and identical rhythmic pattern (4 notes) for q_{rh}.

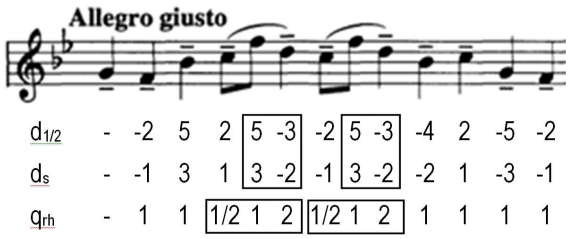


Fig. 1. Three proposed operators used on a motive from Promenade (Pictures at an Exhibition) by Modest Mussorgsky (please note the lack of measures)

2.5 Context-free productions of proposed grammar

A raw description of the *searching-oriented music notation* could be approximated by the set of context-free productions [9] given below. Components of the grammar $G = (V, T, P, S)$ are as follows. The set of nonterminals includes all identifiers in triangle brackets printed in italic. The nonterminal <*score*> is the initial symbol of G. The set of terminals includes all non bracketed identifiers.

<*score*> → <*score_part*> <*score*> | <*score_part*>
<*score_part*> → <*voice*> <*score_part*> | <*voice*>
<*voice*> → <*chord*> <*voice*> | note <*voice*> | rest <*voice*> | <*voice*>
<*chord*> → note <*chord*> | note

Grammar description and comments: <*score*> may consist of one or several elements of type <*score_part*>, that represent subsequent parts of score in time dimension. It is preferable that <*score_part*> would be maintained in the constant *key*, as searching (when including diatonic transpositions of pattern) may depend on *scale*. Each <*score_part*> consists of one or several elements of type <*voice*>, that represent voices (in the sense of *instrumental parts*, *polyphonic voices*, leading motive, accompaniment, etc.), performed simultaneously for a given <*score_part*>. <*voice*> may contain terminals **note**, **rest** or non-terminal <*chord*>, that represent subsequent vertical events in time dimension. Definition of <*chord*> cannot be strict in practice, as it may contain notes of various durations and even rests (depending on the strictness of <*voice*> definition). Terminal **rest** contains obligatory attribute *duration*. Terminal **note** contains obligatory attributes *pitch* and *duration* and may contain non-obligatory attributes, for example related to articulation or dynamics. We also propose to include meta-data attributes: values of proposed operators: d_s, $d_{1/2}$ and q_{rh}.

3 Advanced searching in spaces of music information

Searching is an operation of locating instance(s) matching a given pattern, i.e. locating instance(s) identical or similar to the pattern. The operation *search* in the space of music information concerns a *pattern*, which is a structure of music information. Searched pattern is usually a result of another non-trivial operation: *selection*.

Searching in more general context could be seen as a particular *querying* operation. According to the Cambridge Dictionaries Online *query* is *a question, often expressing doubt about something or looking for an answer from an authority*, c.f. [3]. In this paper we assume that *answer from an authority* is also understood as accomplishment of an operation for a given request. *Querying* in spaces of music information could signify operations like: selecting, searching, copying, replacing, pasting, transposing etc. In this work we are interested in advanced searching operations with regard to melody transpositions and rhythm transformations. Please note that the searching for transformed patterns may occur in the same <*voice*>, for different <*voice*> derivation branches and for different <*score*> derivation trees (representing musical pieces).

3.1 Melodic transformations

This section discuses melodic transformation and searches in this dimension using two introduced operators d_s and $d_{1/2}$. For the discussion we propose the following melodic transformations taxonomy:

1. exact melodic match;
2. *ottava*-type transposition (*all' ottava*, *all' ottava bassa*, etc.) - a particular case of transposition;
3. diatonic transposition (maintaining the diatonic intervals, what could result in slight change of the number of halftones between the corresponding notes);
4. chromatic transposition (maintaining the exact number of halftones between the corresponding notes);
5. irregular melodic transformations (variation type).

Exact melody match is detected with a given d_s (or $d_{1/2}$) sequence and at least one corresponding note being identical. Please note that due to lack of artificial divisions (alike measures, staves, pages, etc.) it is relatively easy to query the structure of information. Longer motives could also start in different *moments* of measure, what is natural for the proposed representation.

Chromatic transposition With $d_{1/2}$ it is very easy to detect *chromatic transposition*, as sequence of identical values would signify chromatically transposed melody fragments.

Diatonic transposition With d_s it is very easy to detect *scalar transposition*, as identical sequences would signify similar melody fragments with regard to the defined *key* (the exact number of halftones can vary however).

***Ottava*-type transpositions** are detected with a given d_s (or $d_{1/2}$) sequence and at least one corresponding *note* name (excluding *octave designation*) being identical. This is an alternative to matching of *note names* sequences. It is more potent approach as *ottava*-type transpositions could be detected during general transposition searches.

Irregular melodic transformations (as for example *tonal answer* in fugue) may be detected with the use of similarity measures for the compared sequences. Such similarity measures may be for example *Levenstein editing distance*, as occasional interval modifications may occur due to harmonic relations, not alternating the most sequence. This is a frequent case for *fugae* tonal answer (*comes*) Using such similarity measure on an unfixed-length sub-sequence affects heavily the computational complexity, therefore we propose to limit the possible alternations to a small amount (1 to 3, depending on the length).

3.2 Rhythm transformations

This section discuses rhythmic transformation and searches in *duration dimension* using introduced operator q_rh. For the discussion we propose the following rhythmic transformations taxonomy:

1. exact match
2. diminution - a melodic pattern is presented in shorter note durations than previously used (usually twice);
3. augmentation - a melodic pattern is presented in longer note durations than previously used (usually twice);
4. irregular rhythmic transformations (of variation type).

Exact rhythmic match is detected with a given q_{rh} sequence and at least one corresponding duration being identical.

Diminutions and augmentations With q_{rh} it is very easy to detect *diminutions* as well as *augmentations* of any kind, as sequence of identical q_{rh} values would signify identical rhythm dependencies.

Irregular rhythm transformations may be detected with the use of similarity measures for the compared sequences.

3.3 Generalization of proposed operators

The three proposed operators are the most potent for a single voice line, however they are applicable as well to chords and even whole musical pieces. Chords sequences could be processed in the following ways:

- **parallel calculations**: assuming the equal number of notes in each chord, operators could be determined in parallel. That would be useful for frequent *thirds* or *sixths* sequences.
- **each-to-each calculations**: each-to-each relations are determined and stored. That increases the amount of data to be generated and could result in arbitrary false searching matches.

Advantage of the *each-to-each calculations*: the whole musical piece could be analyzed at once, disregarding voice selection. It could be very useful for missing, incomplete or erroneous voice data (resulted frequently from automatic transcription or conversion) and may serve as meta-information for discovering structures.

3.4 Similarity-based searching

The melodic and rhythmic transformations described above - when combined - could be complex. It is important to consider intrinsic structure of musical piece, that heavily affects the variability and repeatability of motives. One such example is fugae, that

is dedicated to repeating motives, frequently transformed melodically and rhythmically. A common fugae structure is one voice presenting the *subject* (main theme), and then another voice coming with *the answer* usually altered harmonically (what implies melodic transformations). Another example is musical form *Variation*, where the alteration of main motive are multiple and combined, including rhythm, melody, harmony, structure, dynamics etc.

Searching at two planes simultaneously (melodic and rhythmic planes) may be difficult regarding the unknown length of motives and the fact that motive may be heavily transformed. This is simplified by using the proposed operators, that are of relative character and do not depend on absolute values (pitch and duration). The proposed grammar largely simplifies searching, as operator values may be seen as (mostly parallel) sequences of floating point values. The task of identifying unknown-length sub-sequences in a long sequences (including two or three dimensions of values) is computationally complex. Possible transformations induce the need for efficient similarity measures, that will overlook small changes but react to large ones.

4 Methodology and examples

In this section we present examples of advanced searching with a short analysis of operators' values and motives similarity. The analysis is based on an short except (18 measures in 5 voices) from W. A. Mozart work, presented in Fig. 2. The musical piece is an interesting example because of its slightly variational characteristic.

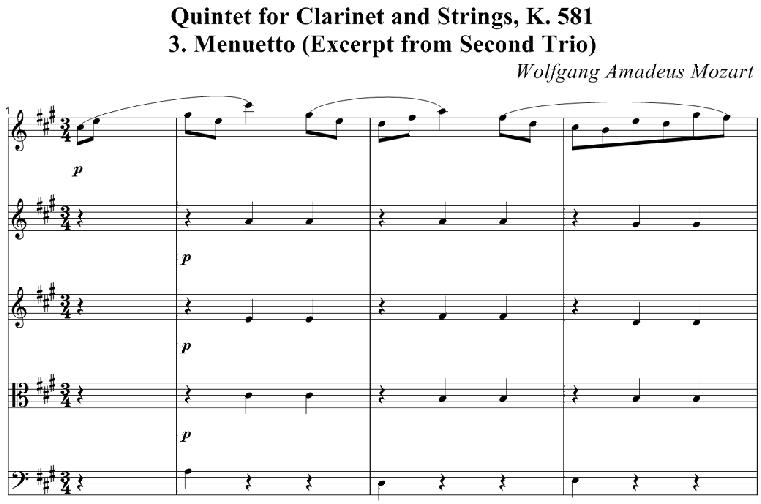


Fig. 2. Quintet for Clarinet and Strings (the beginning), K.581, W. A. Mozart

The presented fragment consists of 5 instrumental parts, described by proposed operators as in Fig. 3.

The most viable method for rhythm or melody analysis would be to analyze each *part of music piece* separately, as specific parts differ in character depending on musical

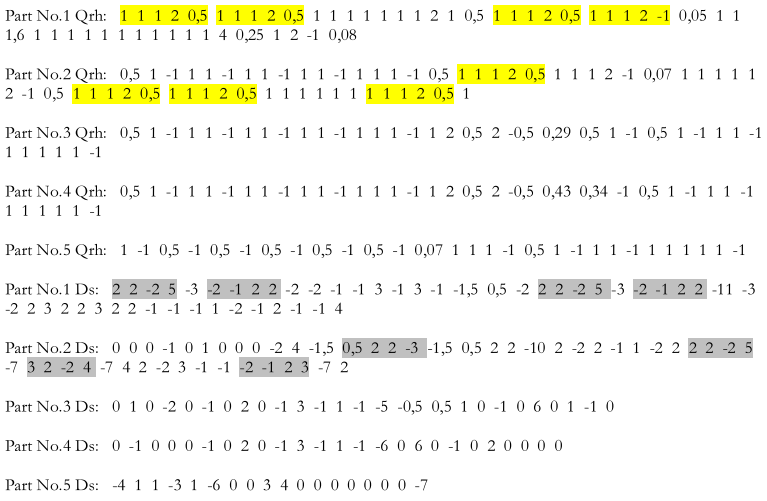


Fig. 3. Rhythmic motives detected in different voices by operator q_{rh} and verified by d_s

genre, form, composer and epoque. One may define various metrics for characterization of instrumental (vocal) *part*;

With a method for part characterization one may specify various comparisons:

- comparison of instrumental (vocal) parts in a single musical piece;
- comparison of two musical pieces, taken as a whole;
- comparison of specific instrumental (vocal) parts from two musical pieces; for example leading voices of two songs;
- comparison of two groups of musical pieces, for example works of two composers or comparison of fugaes to preludes;

4.1 Searching for leitmotives

Searching for leitmotifs, especially transformed ones, may be greatly facilitated with the use of proposed operators. Rhythmic and melodic analysis should be performed in parallel for that cause. An exemplary search for leitmotifs for the musical piece presented in Fig. 2 (only the beginning, due to the lack of space) and Fig. 3 follows in this section.

Main motive occurs mostly in top and second voice. The operator q_{rh} may be used to detect several similar rhythmic patterns (5 notes long), that correspond to the main motive (consisting of two five-notes movements). When analyzing corresponding d_s values, one may see two occurrences of main melodic motive in top voice, and a single modified occurrence in second voice. Coda seems to be appearing in second voice and seems to be similar to second part of main motive. Rhythmic pattern in the middle of the second voice seem to be misleading as melodic pattern is totally different from the main motive.

As one can see the operators can partially detect modified patterns, however in order to fully match them a strict measures of similarity of the operator-generated

sequences should be defined. Such similarity measure could match corresponding operators' values. It may even employ harmony knowledge (inherent relations between pitches of simultaneous or close notes).

4.2 Rhythmic analysis

Analysis of rhythm of music part (or the whole piece) may be done and presented using various (strictly domain-dependent) features, as proposed in the following enumeration, with graphical examples in Fig 4:

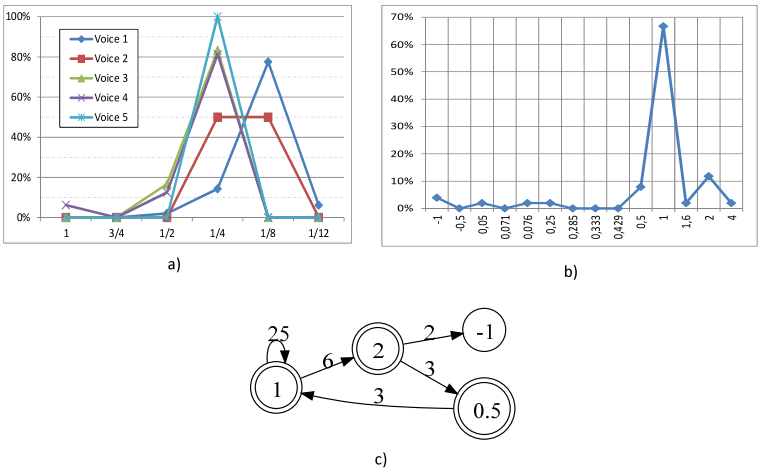


Fig. 4. Rhythmic analysis: a) note lengths; b) q_{rh} distribution; c) oriented graph representing threesome notes with transitions

1. statistics of occurrences of lengths for separate notes and rests, example in Fig 4a. Five parts are presented, it is easy to identify the role of each part: voice 1 consists mostly of eight-notes, signifying that most of the leading melody occurs in the voice. Voices 3, 4 and 5 consist mostly of quarters and serve as accompaniment. Voice 2 is in-between, as it accompanies voice 1, but has also leading moments, what could be seen in the graph, where quarter-notes and eighth-notes are of equal count.
2. statistics of occurrences of separate q_{rh} ratios, it corresponds to pairs of consequent notes/rests example in Fig 4b. Only voice 1 is shown for clarity reasons. The rhythm is mostly uniform, with occasional slight changes, mostly doubling or dichotomizing (values 2 and 1/2).
3. statistics of occurrences of two consequent q_{rh} ratios, it corresponds to threesome consequent notes/rests. This could be quite efficiently shown as a directed graph, where nodes are q_{rh} ratios and branches are labeled with popularity of the transition from one q_{rh} ratio value to another. An simplified (disregarding rare transitions) example for voice 1 is shown in Fig 4c.

4. longer sequences of q_{rh} ratios corresponding to rhythmic patterns may be seen as equivalent to real-number time series, as seen in the Fig. 3, q_{rh} values. The problem however may be simplified at various points:
   - These theoretically real-numbers in practice are a small set, as length differences of notes/rests are limited in practice. Similarly complicated rhythms resulting from non-standard note/rest lengths are rather rare.
   - the length of a pattern corresponding to a musical motive (regarding rhythm) is limited in practice to approximated values [2;15].
   - noisiness of the data is usually absent, as the data is precise, providing that the data source is music notation. In a rare case that the data source is a live capturing that involves approximation however (like MIDI capturing of live music performance or music notation acquirement from raw sound files, alike PCM), the noise may be present.

Basing on the features similarity metrics may be developed. For the above-mentioned cases *a)* and *b)* similarity may be measured as for a feature vector of a fixed length, with a plethora of distances proposed in the literature [14]. One may consider if a kind of weighting should be defined for specific vector elements, regarding them as more/less important than others. Similarity for case *c)* may be seen as comparison of two weighted bi-directional graphs. Finally, the most complicated case *d)* may be simplified using the above-mentioned domain-dependent knowledge.

4.3 Melodic analysis

Melodic analysis is quite similar to rhythm analysis. Melody however could be defined as *a sequence of notes of specified pitch*, therefore it is unlikely to regard melody of a **single** note. Regarding rhythm, a single note length could be seen in relation to a unforeseen standard length (usually music pieces use 2 to 4 *quarter-notes* for a *measure*, therefore a *quarter-note* may be seen as a substantial portion of a measure, while sixteenth-note may be see as a not-so-important one. The separate pitches of notes however are more likely to carry the harmonic information than the melodic one.

For melodic analysis both operators (scalar differences d_s and halftone difference $d_{1/2}$, introduced in section 2) may be of great help, however regarding the Western music (strictly related to major/minor scales) scalar differences d_s is much more viable. Therefore it is used for the following considerations and examples. Likewise, melodic analysis of a music part (or the whole piece) may be done and presented using various (strictly domain-dependent) features, as proposed in the following enumeration, with graphical examples in Fig 5:

1. statistics of occurrences of separate scalar differences d_s, it corresponds to pairs of consequent notes. An example is shown in Fig 5a. Such information may be interpreted to find out the overall melody character: is it smooth with lots of small differences or repetitions (d_s values close to 0) voices 1, 2 and 3 or *jumpy* with a significant amount of larger differences (voices 4 and 5)? It may be also smoothly progressing upwards and jumping down, what could be seen as negative skewness of the histogram (voice 1). Descriptive statistics may be easily used to gather the melody characteristics, based on this simple feature. Various bin sizes may be used to gather slightly less detailed and more general tendencies. We propose to use domain dependent bins, that will more efficiently describe the data. Examples of such bins are given in Fig 5b-d. Bins defined as in Fig 5b and c may serve as a detector of melody progression, with the second one being more detailed.

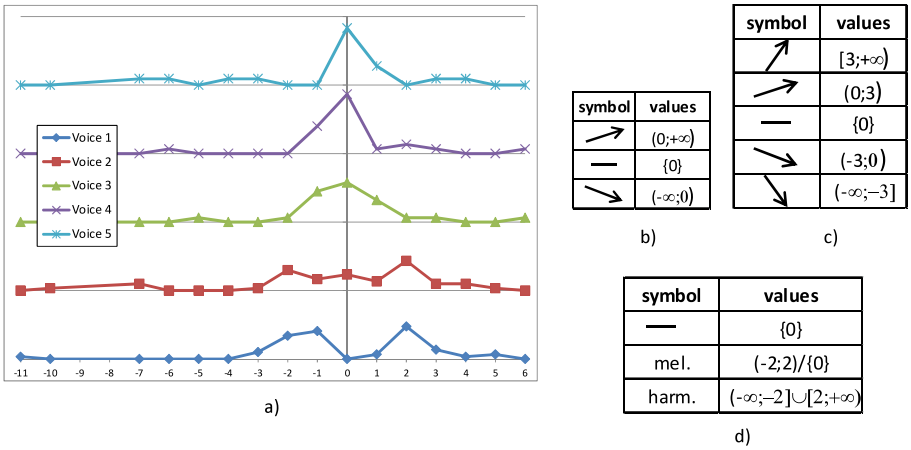


Fig. 5. Melodic analysis: a) d_s distribution; b) c) d) domain-dependent bins

Bins defined as in Fig 5d may detect overall characteristics of the *part*, slight changes hint leading melody, while large jumps are more likely to signal *passaged* accompaniment.

2. statistics of occurrences of two consequent scalar differences d_s, it corresponds to a melodic progress of three notes, related to the specific scale. Similarly to rhythm, a directed graph, where nodes are d_s differences and branches represent transitions, would be visually inefficient, due to a large amount of nodes and lots of branches. It could be however still processed digitally. We propose to use mentioned-above domain-dependent bins, that will more efficiently describe the data.
3. longer sequences of d_s ratios corresponding to melodic patterns may be seen as equivalent to real-number time series, with remarks as above for rhythmic patterns.

Basing on the features similarity metrics may be developed, with parallel methodology as described above for rhythmic patterns.

5 Conclusions and future work

In this paper we present methodology and examples for analysis of rhythmic and melodic sequences as well as searching for leitmotives. For the purpose we use specialized grammar oriented at advanced searching (described in details in [7]) with three operators that describe relations between neighboring notes, regarding melody and rhythm. The resulting values are attached to notes as properties, for use in searches.

Further applications of proposed *searching-oriented grammar* may include knowledge discovery by automated searches in whole musical pieces. Disregarding voice selection it could be performed by using generalizations of proposed operators (as described in section 3.3). The melodic and rhythmic analysis using proposed operators could be used to detect the structure of musical work e.g. divide it into separate voices of consistent constitution.

This study continues construction of specialized grammars covering music information. The grammar is *searching-oriented* but other orientation could be more suitable in

some applications. Future work in this domain include: a) developing details of another specialized grammars, b) research on semantics, i.e. developing methods of construction of valuation relation, as a key issue in automation of querying, c) development of *Select* and *Replace* operations for music notation, d) development of similarity metrics to detect irregular transformations.

References

1. Bargiela A., Homenda W., Information structuring in natural language communication: Syntactical approach, Journal of Intelligent & Fuzzy Systems 17 (2006), 575–581, 2006
2. Castan G. et al., Extensible Markup Language (XML) for Music Applications: An Introduction, Hewlett W. B. and Selfridge-Field E. (Eds.) The Virtual Score: Representation, Retrieval, Restoration, The MIT Press, 2001
3. Collins Cobuild English Language Dictionary, `http://dictionary.cambridge.org/`
4. Homenda W., Breaking Accessibility Barriers: Computational Intelligence in Music Processing for Blind People, Studies in Computational Intelligence (SCI) 107, pp. 207–232, Springer-Verlag Berlin Heidelberg, 2008
5. Homenda W., Automatic data understanding: a necessity of intelligent communication, Artifical Intelligence and Soft Computing, LNAI 6114, pp. 476–483, Springer, 2010
6. Homenda W., Sitarek T., Notes on automatic music conversions, Foundations of Intelligent Systems, LNAI 6840, pp. 533–542, Springer-Verlag Berlin Heidelberg, 2011
7. Rybnik M., Homenda W., Sitarek T., Advanced Searching in Spaces of Music Information, Lecture Notes in Artificial Intelligence (LNAI), Vol. 7661, Springer, Proceedings of 20th International Symposium on Methodologies for Intelligent Systems, pp. 218–227, Macau, China, December 4–7, 2012
8. Homenda W., Rybnik M., Querying in Spaces of Music Information, LNCS 7027, pp. 243–255, Integrated Uncertainty in Knowledge Modelling and Decision Making, Hangzhou, China, 2011
9. Hopcroft J. E., Ullman J. D., Introduction to Automata Theory, Languages and Computation, Addison-Wesley Publishing Company, 1979, 2001
10. Krolick B., How to Read Braille Music, 2nd Edition, Opus Technologies, 1998
11. MIDI 1.0, Detailed Specification, Document version 4.1.1, February 1990
12. Tadeusiewicz R., Ogiela M. R.: Automatic Image Understanding A New Paradigm for Intelligent Medical Image Analysis, Bioalgorithms and Med-Systems, Vol. 2, No. 3, pp. 5–11, 2006
13. Tadeusiewicz R., Ogiela M. R.: Why Automatic Understanding?, Adaptive and Natural Computing Algorithms, LNCS Vol. 4432, Springer-Verlag Berlin Heidelberg, pp. 477–491, 2007
14. Abello J. M., Pardalos P. M., and Resende M. G. C. (editors): Handbook of Massive Data Sets, Springer, ISBN 1-4020-0489-3, 2002

Author Index