Reports of Enlarged Session of the
Seminar of I. Vekua Institute
of Applied Mathematics
Volume 28, 2014

# TOWARDS THEOREM PROVING TECHNIQUES IN FORMULA SCHEMATA

Rukhaia M.

**Abstract**. The theorem proving techniques are divided into two parts, goal-directed and refutational. In this paper we present a goal-directed proof-search algorithm, which is based on a sequent calculus. Usually sequent calculus inference rules can be applied freely, producing a redundant search space. The technique, called focusing, removes this nondeterminism and redundancy in proof-search. Although we do not present a focused calculus, our algorithm is obtained according to the principles of focusing, achieving similar effect.

**Keywords and phrases**: Theorem proving, formula schemata, focusing.

**AMS subject classification**: 03B70, 68T15.

**1. Introduction.** The proof theory takes its roots from G. Gentzen, when he introduced a sequent calculus, **L***ogische* **K***alkül*, for first-order logic [10]. Since then, proofs are heavily used in computer science, in particular, program and hardware verification. This gave rise to the theorem proving, a new branch of mathematical logic. There are various theorem proving techniques, like resolution, tableaux, etc. It is well known that first-order logic is undecidable, therefore all complete proof-search procedures are non-terminating.

The concept of *term schematization* was introduced in [5] to avoid non-termination in symbolic computation procedures and to give finite descriptions of infinite derivations. Later, *formula schemata* for propositional logic was developed [1, 3] to deal with schematic problems (graph coloring, digital circuits, etc.) in more uniform way and a tableaux prover, called **RegSTAB**, for a class of formula schemata was implemented [2].

In [6, 9] the language of formula schemata was extended to first-order logic and a sequent calculus was defined, introducing a notion of *proof schema*. The aim of this paper is to define a proof-search procedure using a sequent calculus, that will, for a given formula schema, obtain a proof schema.

The sequent calculus **LK** leads to a redundant search space, since inference rules can be applied freely. In [7], a concept of *polarity* was introduced and a *focused* sequent calculus **LC**, based on polarity, was defined. In [8], **LC** was adopted to proof-search and a sequent calculus **LKF** was obtained. The latter significantly reduced the proof-search space.

To achieve the aim, we use a similar approach. Although we do not define a focused sequent calculus for formula schemata, we give a proof-search procedure based on principles of focusing.

**2. Preliminaries.** We define a *schematic first-order language*, following [6], that is an extension of the language described in [1, 3] to first-order logic. It allows us to specify an (infinite) set of first-order formulas by a finite term.

We consider two sorts $\omega$, to represent the natural numbers, and $\iota$, to represent an arbitrary first-order domain. Our language consists of countable sets of *variables* of both sorts, and sorted $n$-ary function and predicate symbols partitioned into *constant function/predicate symbols* and *defined function/predicate symbols*. The latter allows primitive recursively defined functions/predicates in the language.

*Terms* are built from variables and constant function symbols as usual. We assume the predefined constant functions zero $0\colon \omega$ and successor $s\colon \omega \to \omega$ to be present. By $V(t)$ we denote a variable set of a term $t$, and by $\overline{\cdot}$ we denote sequence of terms of appropriate sort.

For every defined function symbol $f$, we assume that its sort is $\omega \times \tau_1 \times \cdots \times \tau_n \to \tau$ (with $n \geq 0$ and $\tau ::= \omega \mid \iota \mid \tau \to \tau$), and we assume given two rewrite rules

$$f(0, \overline{x}) \to t_0 \qquad f(s(y), \overline{x}) \to t[f(y, \overline{x})]$$

such that $V(t_0) \subseteq \{x_1, \ldots, x_n\}$ and $V(t[f(y, \overline{x})]) \subseteq \{y, x_1, \ldots, x_n\}$, and $t_0, t$ are terms not containing $f$; if a defined function symbol $g$ occurs in $t_0$ or $t$ then $g \prec f$. We assume that these rewrite rules are primitive recursive, i.e. that $\prec$ is irreflexive.

We write $t \twoheadrightarrow t'$ to denote that an expression $t$ rewrites to an expression $t'$ in arbitrarily many steps.

*Formulas* are built inductively from atoms using the logical connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$ and $\exists$ as usual. A variable occurrence in a formula is called *bound* if it is in the scope of $\forall$ or $\exists$ connectives, otherwise it is called *free*. The notions of interpretation, satisfiability and validity of formulas are defined in the usual classical sense.

Analogously to defined function symbols, we assume that for defined predicate symbols rewrite rules are given and have an irreflexive order $\prec$ for the latter, to build *formula schemata*.

**Example 1.** Let $P\colon \omega$ be a defined predicate symbol, $R\colon \omega \times \iota$ a constant predicate symbol, and $x\colon \iota$ a variable. Let the rewrite rules for $P$ be

$$P(0) \to \forall x R(0, x) \qquad P(s(n)) \to \exists x (R(n, x) \wedge P(n)).$$

Then we have $P(2) \twoheadrightarrow \exists x (R(2, x) \wedge \exists x (R(1, x) \wedge \forall x R(0, x)))$ which is equivalent to (by renaming of bound variables) $\exists x_2 (R(2, x_2) \wedge \exists x_1 (R(1, x_1) \wedge \forall x_0 R(0, x_0)))$.

**Proposition 1.** *Let $A$ be a formula. Then every rewrite sequence starting at $A$ terminates, and $A$ has a unique normal form.*

**Proof.** Trivial, since all definitions are primitive recursive.

*Sequents* are expressions of the form $\Gamma \vdash \Delta$, where $\Gamma$ and $\Delta$ are multisets of formula schemata. Sequents are denoted by $S(\overline{x})$, where $\overline{x}$ are free variables occurring in $S$.

The sequent calculus **LKs** is defined as in [6, 9], but without the *cut* rule. The *proof axioms*, which are called *proof links* in [6, 9], may appear only at the leaves of a proof. A proof axiom corresponds to induction hypothesis, i.e. it is assumed that a proof $\varphi$ at step $k$ proves a sequent $S(k)$.

A *proof schema* $\Psi$ is a tuple of **LKs**-proof pairs for $\varphi_1, \ldots, \varphi_n$ proof symbols, where each pair contains base and recursive case of inductive definition. The proof symbols in a proof schema must be ordered in a sense that if $i > j$, $\varphi_i$ must not contain a proof axiom referring to $\varphi_j$. We also say that the end-sequent of $\varphi_1$ is the *end-sequent* of $\Psi$. For a formal definition of proof schemata we refer an interested reader to [6, 9].

According to the above definition, it is easy to see that proof schemata naturally represent infinite sequences of first-order proofs.

**Proposition 2.** *The sequent calculus* **LKs** *is sound.*

**Proof.** Although the calculus is not the same, proof is similar to the ones in [6, 9].

**3. Focusing.** The idea of focusing lies on the notion of *polarity* of formulas and logical connectives. There are *positive* and *negative* polarities. Positive (negative) formulas are constructed from atomic (negation of atomic) formulas and positive (negative) applications of the connectives. The polarity of connectives is isomorphic to their truth table, where positive stands for *false* and negative stands for *true*.

These polarities does not affect the provability of formulas, but the shape of proofs and proof-construction steps. Unlike Gentzen's sequent calculus, the proof-construction steps become deterministic using the *focusing* technique.

First of all, sequents are either *focused* or *unfocused*. Focused sequent consists of positive formulas or negative literals and exactly one formula is in *focus*. Unfocused sequents might contain arbitrary formulas. Second, inference rules are partitioned to *asynchronous* and *synchronous* rules, where asynchronous ones are invertible [1] and operate on unfocused sequents. Synchronous rules operate on focused sequents only, and in general, they are not invertible, thus producing *backtrack points*.

Because of these restrictions, asynchronous rules are applied eagerly, until a focused sequent is reached. Then synchronous rules decompose the focused formula, until the branch is closed (i.e. positive literal is reached) or negative formula is obtained. In the latter case the focus is dropped and asynchronous rules are applied again.

**4. The algorithm.** Considering the method described in the previous section, we end-up with the algorithm, described below, for efficient proof-search for formula schemata.

To handle quantifiers, we use similar method described in [4]. This means that we work with *skolemized* [2] sequents and the choice for the quantifier instance term is postponed until it is obtained via unification. Extension of the algorithm to non-skolemized sequents is a subject of future work.

Next, note that all propositional rules in **LKs** are invertible and the binary rules duplicate the context. Thus, the application of binary rules must be postponed as far as possible to reduce the search space. Hence, the unary rules must be applied first when applicable.

Construction of a proof schema of a sequent is divided into two tasks: first the **LKs**-proof for base case and then the **LKs**-proof for recursive case must be constructed. The difference between these two lies on the usage of proof axiom and rewriting inference rules (i.e. they are obsolete in the base case).

In the proof-search of recursive case, rewriting of defined function and predicate symbols must be done in a way that each symbol is rewritten only one step down (e.g. going from $k+1$ to $k$). After such rewriting, if no other rules are applicable on a sequent, the proof axiom should be introduced or algorithm must terminate with "no

[1] Informally, invertible rule means that it can be applied in a unique way on a selected formula(s).

[2] Informally, a sequent is skolemized if it does not contain universal quantifiers on the right-hand side and existential quantifiers on the left-hand side.

proof found". If the sequent is a match instance of the end-sequent modulo $k$ parameter, then a proof axiom to itself must be made; otherwise a proof axiom referring to a new proof symbol must be created and new proof-search for this sequent must be scheduled.

It is easy to see that this algorithm is terminating, therefore it is not complete. In fact, the unsatisfiability of formula schemata is a property which is not semi-decidable even for propositional schemata [3], thus no complete algorithm exists.

**5. Conclusions.** We presented a proof-search algorithm for formula schemata, but the algorithm can be improved significantly. The main task is to handle arbitrary sequents. An algorithm, that takes an arbitrary sequent and produces its skolemized version must be defined, together with the algorithm, that will *deskolemize* a result of the proof-search procedure. Alternatively, existing algorithm can be modified to accept arbitrary sequents, but in this case better termination condition should be provided. Finally, the algorithms must be implemented and their efficiency must be tested.

**Acknowledgment.** This work was supported by the project No. PG/6/4-102/13 of the Shota Rustaveli National Science Fundation.

## REFERENCES

1. Aravantinos V., Caferra R., Peltier N. A Schemata calculus for propositional logic. *Tableaux'09, LNCS*, **5607** (2009), 32-46.

2. Aravantinos V., Caferra R., Peltier N. RegSTAB: A SAT-solver for propositional iterated schemata. *International Joint Conference on Automated Reasoning*, (2010), 309-315.

3. Aravantinos V., Caferra R., Peltier N. Decidability and undecidability results for propositional schemata. *Journal of Artificial Intelligence Research*, **40** (2011), 599-656.

4. Autexier S., Mantel H., Stephan W. Simultaneous quantifier elimination. *KI-98: Advances in Artificial Intelligence*, *Springer*, (1998), 141-152.

5. Hong Chen H., Jieh Hsiang J., Kong H.C. On finite representations of infinite sequences of terms. *Edited by S. Kaplan and M. Okada, Conditional and Typed Rewriting Systems, Lecture Notes in Computer Science, Springer Berlin Heidelberg,* **516**, (1991), 99-114.

6. Dunchev C., Leitsch A., Rukhaia M., Weller D. CERES for First-Order Schemata. *Technical report, Vienna University of Technology*, 2012. Available at: `http://arxiv.org/abs/1303.4257`.

7. Girard J.Y. A new constructive logic: classical logic. *Mathematical Structures in Computer Science*, **1**, 3 (1991), 255-296.

8. Liang C., Miller D. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, **410**, 46 (2009), 4747-4768.

9. Rukhaia M. About Cut-Elimination in Schematic Proofs. A monograph. *Lambert Academic Publishing, Saarbrücken*, 2013.

10. Takeuti G. Proof Theory. *North Holland, second edition,* 1987.

Received 25.05.2014; revised 22.11.2014; accepted 27.12.2014.

Author's address:

M. Rukhaia
I. Vekua Institute of Applied Mathematics of
Iv. Javakhishvili Tbilisi State University
2, University St., Tbilisi 0186
Georgia
E-mail: mrukhaia@logic.at