ILF-SETHEO Processing Model Elimination Proofs for Natural Language Output[⋆]

Andreas Wolf, Johann Schumann

Institut für Informatik, Technische Universität München, D-80290 Munich

e-mail: wolfa,schumann@informatik.tu-muenchen.de

Several powerful automated theorem provers for first order predicate logic are based on the *Model Elimination* (ME) calculus (e.g., SETHEO [10], KoMeT [1]). Unfortunately, the proof generated by such a prover, a closed tableau, is not suited to a human who really wants to *understand* the proof. In response to the need for human readable proofs, the *Block Calculus* (BC) [4] has been developed as a variant of *Natural Deduction* (ND) [6]. Proofs represented in this calculus can be easily transformed into a natural language form which is reasonably legible by a human. ILF-SETHEO closes the gap between the machine output of SETHEO (or similar provers[2]), and the proof structure needed to represent an equivalent proof in BC. The natural-language output itself is generated by a tool of the ILF [3] system. Although this transformation of proofs was originally used within the interactive system ILF, ILF-SETHEO is a stand-alone post-processor for SETHEO. It takes a ME tableau, as generated by SETHEO, and produces a natural language proof in LaTeX. For details on SETHEO and its calculus see [10, 9, 7]. In the following, we briefly describe the overall system architecture of ILF-SETHEO and the basic characteristics of the surrounding ILF system. Then, we focus on the transformation itself. An example, taken from the TPTP problem library [12], concludes this system abstract.

ILF. ILF-SETHEO is based on tools from the ILF [3] system. ILF has been developed at the Humboldt-University, Berlin, as an interactive set of tools to "*I*ntegrate *L*ogical *F*unctions".

The default calculus used in ILF is the BC as described in [4]. Compared to the original Natural Deduction calculus, BC is extended with a *block structure* which allows proofs and sub-proofs to be linearized in a canonical way. Furthermore, the concept of *usable formulae* allows formulae to be referenced in a uniform way. Experience with ILF showed that BC is well suited as a target calculus for transformations from other logic calculi, and as a basic structure for natural language output of computer generated proofs [5].

ILF includes specific tools to improve the style of the natural language presentation, e.g., removal of duplicated formulae, transformation of indirect proofs

[⋆] This work is supported by the Deutsche Forschungsgemeinschaft within the Sonderforschungsbereich 342, subproject A5 (PARIS) and the Schwerpunkt Deduktion.

[2] In fact, an adaption of this tool to other model elimination style provers like KoMeT [1] is possible with little implementation effort.

into direct ones (if possible), and reordering of inference chains of the proof to reach optimal readability. For details see [5].

Other approaches [8] extract more complex inference patterns directly from the machine generated proof. This works well on calculi with many rules like ND but brings poor results on ME tableaux. ILF provides another way: the original proof will be transformed into BC at first, followed by a structure analysis of the transformed proof.

The Transformation. The core of ILF-SETHEO is the transformation [13] of a given ME tableau into a proof represented in BC. A ME tableau is constructed by applying four different kinds of inference steps: the ME start-step, extension steps, reduction steps, and factorization [7] steps.

From the given tableau all structures that can be interpreted as linear chains of inferences are extracted and represented as blocks in the BC. A ME start step corresponds to writing down the non-negated formula and creating a new block containing the assumed formula (negated) and the contradiction ($\Box$), marked "unproven"[3]. In BC, proofs are generally constructed beginning from facts. So, the ME extension step is reflected as the application of the rules needed for Modus Ponens. Reduction steps lead to indirect subproofs. The structure of the whole proof is preserved in the order of these blocks. Finally, the BC proof generated this way is enriched with the (optional) LaTeX typesetting information (see below) and sent via e-mail to the automatic ILF-SERVER[4]. After a few minutes a LaTeX file containing the proof in natural language returns.

Features of ILF-SETHEO include:

- ILF-SETHEO works as a true post-processor for SETHEO. This means that there is no loss of performance of the prover. Only the generated proof tree and information about the input clauses is used by ILF-SETHEO.
- ILF supports the construction of large proofs combining several sub-proofs, even originating from different proof sessions. Therefore management functions are available for unique identification of partial proofs. This facilitates the combination of several proofs including references to formulae across the partial proofs (e.g., for the citation of lemmata).
- ILF-SETHEO allows clause names to be specified using the directive `#clausename <name>`. Per default, the number of the clause is used.
- The input language of SETHEO only allows predicates, constants and function symbols to be represented using the ASCII character set. ILF-SETHEO is able to use all features of LaTeX to display the formulae according to predefined (but user-definable) declarations (see the example below).

ILF-SETHEO is implemented in `perl` (300 lines of code) and ECLIPSE-PROLOG (500 loc, portable PROLOG code)[5]. The transformation procedure is

[3] In BC, each formula is marked by "assumption", "unproven", or "proven".

[4] `ilf_serv@mathematik.hu-berlin.de`

[5] ILF-SETHEO is available for interested users, but currently not part of the standard SETHEO distribution. Please contact the authors.

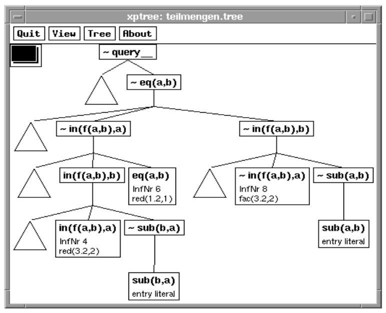
also a significant part of ILF.

An Example. The following example is taken from [2] (SET019-3 in the TPTP [12]). We present the original formulae as a set of clauses in SETHEO input syntax (1), some lines of the translation table of operators defining the LaTeX-output (2), and the ME tableau, comprising a proof of the given problem as displayed by SETHEO's `xptree` tool (3). The output of ILF-SETHEO is appended below. This proof representation has been generated *fully automatically* (except for some line-breaks).

(1)

```
#clausename extensionality2
in(f(X,Y),X);in(f(X,Y),Y);eq(X,Y)<-.
#clausename extensionality3
eq(X,Y) <-in(f(X,Y),X),in(f(X,Y),Y).
#clausename subset1
in(U,Y) <- sub(X,Y),in(U,X).
#clausename a_contains_b
sub(b,a) <- .
#clausename b_contains_a
sub(a,b) <- .
#clausename theorem
<- eq(a,b).
```

(3)



(2)

```
    struct 500   not(in(X,Y)) :- x(X)," \\not\\in ",x(Y)
    struct 500   in(X,Y) :- x(X)," \\in ",x(Y)
    struct 500   eq(X,Y) :- x(X)," \\simeq ",x(Y)
    struct 500   f(X,Y) :- "f_{",x(X),",",x(Y),"}"
```

Conclusions. ILF-SETHEO is an indispensable tool for SETHEO in all cases where there is a need to read and understand the proofs found by SETHEO. Even for users familiar with ME, this tool helps to structure larger proofs and makes them understandable (and printable). Its LaTeX-post-processing features allows a proof to be represented in its original, problem-oriented notation, instead of SETHEO's hard to read ASCII prefix notation. For all kinds of applications of ATP, where the actual proof is required (and not just a yes/no answer), ILF-SETHEO is an important tool (e.g., for verification [11]). Its BC is furthermore suited for combining proofs from different provers.

Future versions of ILF-SETHEO will incorporate important features to further enhance the readability of the proofs, such as hiding of technical details (e. g. low-level equality transformations), introducing lemmata, and further means to structure a proof.

[1] W. Bibel, S. Brüning, U. Egly, and T. Rath. KoMeT, system description. In *Proc. CADE-12*, pp. 783–787, Springer, 1994.

[2] R. Boyer, E. Lusk, W. McCune, R. Overbeek, M. Stickel, and L. Wos. Set theory in first-order logic: Clauses for Gödel's axioms. *JAR*, 2(3): pp. 287–327, 1986.

[3] B. I. Dahn, J. Gehne, Th. Honigmann and A. Wolf. Integration of Automated and Interactive Theorem Proving in ILF. In this volume, 1997.

[4] B. I. Dahn and A. Wolf. A Calculus Supporting Structured Proofs. *Journal for Information Processing and Cybernetics (EIK)*, (5–6): pp. 261–276, 1994.

[5] B. I. Dahn and A. Wolf. Natural Language Presentation and Combination of Automatically Generated Proofs. In *Proc. FroCoS'96*, pp. 175–192, Kluwer, 1996.

[6] G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift 39*, 1939.

[7] Chr. Goller, R. Letz, K. Mayr, and J. Schumann. SETHEO V3.2: Recent Developments (System Abstract) . In *Proc. CADE-12*, pp. 778–782, Springer, 1994.

[8] X. Huang. Reconstructing Proofs at the Assertion Level. In *Proc. CADE-12*, pp. 738–752, Springer, 1994.

[9] R. Letz, K. Mayr, and C. Goller. Controlled Integration of the Cut Rule into Connection Tableau Calculi. *JAR*, (13): pp. 297–337, 1994.

[10] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *JAR*, 8(2): pp. 183–212, 1992.

[11] J. Schumann Automatic Verification of Cryptographic Protocols with SETHEO In this volume, 1997.

[12] G. Sutcliffe, C.B. Suttner, and T. Yemenis. The TPTP Problem Library. In *Proc. CADE-12*, pp. 252–266, Springer, 1994.

[13] A. Wolf. A Translation of Model Elimination Proofs into a Structured Natural Deduction. FLAIRS-97, Daytona Beach, 1997.

A Proof of ILF.

Axiom 1 (`subset 1`): $U \in Y \leftarrow X \subseteq Y \wedge U \in X$.

Axiom 2 (`extensionality 2`): $(f_{X,Y}) \in X \vee (f_{X,Y}) \in Y \vee X \simeq Y$.

Axiom 3 (`extensionality 3`): $X \simeq Y \leftarrow (f_{X,Y}) \in X \wedge (f_{X,Y}) \in Y$.

Axiom 4 (`a contains b`): $b \subseteq a$.

Axiom 5 (`b contains a`): $a \subseteq b$.

Theorem. $a \simeq b$.

Proof. We show indirectly that $a \simeq b$. (1)

Let us assume that $a \not\simeq b$. (2)

We show indirectly that $(f_{a,b}) \in a$. (3)

Let us assume that $(f_{a,b}) \notin a$. (4)

Because of `a contains b` $b \subseteq a$. Because of `subset 1` $A \notin B \leftarrow B \subseteq C \wedge A \notin C$. Therefore by (4) $(f_{a,b}) \notin b$. Because of `extensionality 2` $(f_{A,B}) \in A \leftarrow A \not\simeq B \wedge (f_{A,B}) \notin B$. Therefore by (2) $(f_{a,b}) \in a$. This contradicts (4). Thus we have completed the proof of (3).

Because of `b contains a` $a \subseteq b$. Because of `subset 1` $A \in B \leftarrow C \subseteq B \wedge A \in C$. Therefore by (3) $(f_{a,b}) \in b$. Because of `extensionality 3` $A \simeq B \leftarrow (f_{A,B}) \in A \wedge (f_{A,B}) \in B$. Therefore by (3) $a \simeq b$. This contradicts (2). Thus we have completed the proof of (1). q.e.d.