JLog: Automated Theorem Prover for Johansson's Minimal Logic of Predicates

Tatevik Ohanyan
Yerevan State University
Department of Informatics and Applied Mathematics
Yerevan, Armenia
e-mail: tatev.ohanyan@gmail.com

Hovhannes Bolibekyan
Yerevan State University
Department of Informatics and Applied Mathematics
Yerevan, Armenia
e-mail: bolibekhov@ysu.am

ABSTRACT
In the paper a theorem prover based on a cut-free sequent system for Johansson's minimal logic of predicates is described. It uses loop checking mechanism to avoid redundancies. Proofs may be displayed in a Fitch-style natural deduction format.

Keywords
Prover, predicate, calculus, sequent, minimal, logic.

1. OVERVIEW
Automated theorem proving has matured into one of the most advanced areas of computer science. The area of applications varies from proof of mathematical statements to formal verification of complex software and hardware systems. There are numerous provers for classical logic(for example SPASS, Flotter, Vampire, Fiesta, etc. [1]-[3]).

In the paper a fully automated theorem prover JLog for a cut-free sequent system for Johansson's minimal logic of predicates is described. Though the method it uses is unsophisticated on small problems JLog is fast. Its speed is achieved by careful coding in a low-level language (C++) and by eliminating many obvious redundancies in proof searches. The system is useful for a comparison of some new strategies for proof search implementations based on cut-free sequent systems for first order predicate logic.

2. LOGIC
Following to [4] let us introduce a sequential cut-free calculus for Johansson's minimal logic of predicates. By a sequent we mean a pair $<\Gamma, B>$ where Γ is a set of predicate formulae of Johansson's minimal logic in a given language and B is a single formula at most. As usual where Γ is finite the sequent $A_1, \ldots, A_n \rightarrow B$ is considered and read as $A_1, \ldots, A_n$ entail B. Let's recall axiomatic system introduced in [4].

We consider that in the scope of this system A, B, C and D are any formulae, Γ and Δ are finite (may be empty) sets of formulae, Θ is empty or consists of only one formula, x is a variable, t is a term which is free for substitution instead of x in $A(x)$ formula, d is a variable which is free for substitution instead of x in $A(x)$ formula and (if d is different from x) then it isn't free in $A(x)$. (Note that if A and B are formulae and x is a variable, then $(A \supset B)$, $(A \& B)$, $(A \vee B)$, $\neg(A)$, $\forall x(A)$, $\exists x(A)$ are formulae.)

Logical axioms of the system are presented as sequents of the form $C, \Gamma \rightarrow C$, where Γ is a set of predicate formulae and C is an arbitrary predicate formula.

Given below inference rules of the system are divided into antecedential and succedential ones for every logical connective – $\&$, $\vee$, $\supset$, $\neg$, $\forall$, $\exists$ used in the language.

$$\frac{A, \Gamma \rightarrow B}{\Gamma \rightarrow (A \supset B)} \qquad \frac{(A \supset B), \Gamma \rightarrow A \text{ and } B, (A \supset B), \Gamma \rightarrow \Theta}{(A \supset B), \Gamma \rightarrow \Theta}$$

$$\frac{\Gamma \rightarrow A \text{ and } \Gamma \rightarrow B}{\Gamma \rightarrow (A \& B)}$$

$$\frac{A, (A \& B), \Gamma \rightarrow \Theta \text{ or } B, (A \& B), \Gamma \rightarrow \Theta}{(A \& B), \Gamma \rightarrow \Theta}$$

$$\frac{\Gamma \rightarrow A \text{ or } \Gamma \rightarrow B}{\Gamma \rightarrow (A \vee B)}$$

$$\frac{A, (A \vee B), \Gamma \rightarrow \Theta \text{ and } \ B, (A \vee B), \Gamma \rightarrow \Theta}{(A \vee B), \Gamma \rightarrow \Theta}$$

$$\frac{A, \Gamma \rightarrow}{\Gamma \rightarrow \neg(A)} \qquad \frac{\neg(A), \Gamma \rightarrow A}{\neg(A), \Gamma \rightarrow}$$

$$\frac{\Gamma \rightarrow A(d)}{\Gamma \rightarrow \forall x A(x)}\ {}^{1} \qquad \frac{A(t), \forall x A(x), \Gamma \rightarrow \Theta}{\forall x A(x), \Gamma \rightarrow \Theta}$$

$$\frac{\Gamma \rightarrow A(t)}{\Gamma \rightarrow \exists x A(x)} \qquad \frac{A(d), \exists x A(x), \Gamma \rightarrow \Theta}{\exists x A(x), \Gamma \rightarrow \Theta}\ {}_{1}$$

In [4] equivalence of this cut-free system and the system with cut rule was proved as well as equivalence to the Hilbert system of minimal logic. As it is generally accepted antecedential rules (rules of the left column) are denoted by $\supset\rightarrow$, $\&\rightarrow$, $\vee \rightarrow$, $\neg \rightarrow$, $\forall \rightarrow$, $\exists \rightarrow$ and succedential rules (rules of the right column) - by $\rightarrow\supset$, $\rightarrow\&$, $\rightarrow \vee$, $\rightarrow \neg$, $\rightarrow \forall$, $\rightarrow \exists$ respectively.

[1] d does not occur free in a bottom sequent.

3. ALGORITHM

The implementation of the prove based on the above mentioned sequent calculus is fairly straightforward. The main function is a module INFERENCE which takes as an argument an initial sequent and delivers as a value a pointer to a proof tree of that sequent if there is one or null pointer if there is not. JLog deletes that sequent from the list of those under test on returning a proof or disproof of an irreducible sequent.

As we have seen earlier inference rules are of two types - rules for logical connectives in an antecedent and in a succedent. The inference of the given sequent goes in the following order:

1. The sequent is verified on being an axiom or not. If it is an axiom the proof is trivial otherwise it should be decided for which formula apply an inference rule.
2. If all formulae in a sequent are trivial and the sequent is not an axiom, failure. Otherwise for the formula in a sequent selected by the second filter described below an inference rule is applied. The precedence is given to a formula in succedent. If the succedential formula is trivial (it cannot be divided any more) a formula from antecedent is selected. Filtering described after step five is applied on each step of proof search excepting trivial case when sequent is an axiom.
3. As branching in the proof tree occurs if the principle formula is parsed by $\vee$, & in antecendent or in succedent, and if the principle formula is parsed by $\supset$ in antecedent we introduce some proof restrictions.
   - 3.1 Return the proof in the case it is found if an axiom is achieved at least in one branch of the tree in the following cases
     - 3.1.1 the formula in succedent contains $\vee$ as a main connective
     - 3.1.2 the formula in antecedent contains & as a main connective
   - 3.2 Return the proof if one is found when an axiom is achieved in both branches of the tree in the following cases
     - 3.2.1 the formula in succedent contains & as a main connective
     - 3.2.2 the formula in antecedent contains $\supset$ as a main connective
     - 3.2.3 the formula in antecedent contains $\vee$ as a main connective
4. If the principle formula is parsed by $\supset$ in succedent, $\neg$, $\forall$, $\exists$ in any of succedent or antecedent the tree has only one branch. The proof is found if axiom is achieved.
5. Nothing has worked, so return failure.

It remains to describe the filters applied during the proof search. For each formula in a sequent we introduce some label (true or false, it is false by default for all formulae) indicating if some inference rule has been applied to it. Its value can be changed no more than once during the proof search. If the rule has been applied to it there will be no more applications of that rule on a considered formula as its label is not reset until the end of a proof or a failure. It is worth to notice that if in the result of an application of some inference rule some formula should be added to a succedent or an antecedent that already contains an entry of that formula then the label of the added formula is changed. This filter is applied on each step of the proof.

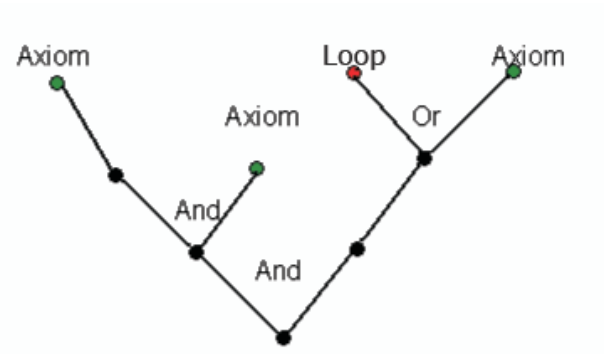
On step two we should decide to which formula inference rule should be applied. This is done using subformula property. To be more precise, this filter is applied only for $\supset\rightarrow$ inference rule. It is verified if the left promise of the principal formula is contained in Γ. If it is contained then the rule is applied to that formula, otherwise the search to satisfy this filter condition is continued. If no formula is found the first one from the set is selected. Further success or failure of proof search depends on the filter described in the next paragraph.

Finally the third filter developed in JLog is a loop detection mechanism. Before proceeding to the filter description let us define the notion of the loop. *Loop* is a sequence of sequents where the last item of the sequence coincides with its first item. Using the notion of a loop proof search in JLog fails in the following cases:

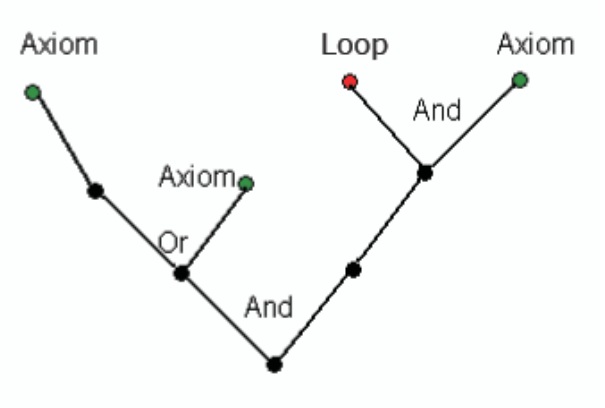
- If every branch of the tree with the root of the form $(A\supset B)$, $\Gamma\rightarrow\Theta$ or of the form $\Gamma\rightarrow(A\vee B)$, where A , B are formulae, and Γ , Θ are sets of formulae, contains a loop. (To prove a sequent in the case of $\supset\rightarrow$ and $\rightarrow\vee$ rules one should prove both sequents in new branches.)
- For the rest of the rules ($\rightarrow\supset$,$\rightarrow\&$, $\rightarrow\neg$, $\rightarrow\forall$, $\rightarrow\exists$, $\supset\rightarrow$, $\vee\rightarrow$, $\neg\rightarrow$, $\forall\rightarrow$, $\exists\rightarrow$) failure occurs if any of the branches generated by their application contains a loop.

If no loop is detected the proof search will go on.

The tree on the figure below schematically demonstrates an example when proof is found though one of the branches in proof tree contains a loop.



Contrary to the above given case the tree below demonstrates an example when proof is failed as it is required that all branches should be terminated by axioms.



4. FORMAT OF INPUT AND OUTPUT

A sequent is presented in JLog as a string where the antecedent of the sequent is presented by formulae separated by commas, and the succedent consists of at most one formula. Succedent and antecedent are separated by an arrow. Influenced by the specifications of described logical system both succedent and antecedent might also be empty sets of formulae. Formulae are written in infix notation. It is important to mention that parentheses for each formula are required as it was described above. Disjunction is represented as "|", conjunction as ampersand "&", negation as "!", implication as ">", generalization quantifier as "#", existential quantifier as "$", and the arrow which divides the antecedent and succedent as "−>" (as a 'minus' followed by 'greater than'). Capital letters with indices are reserved for predicates (propositions should also be given in uppercase without parenthesis), f, g, and h with indices are reserved for functional symbols, a, b, and c with indices are used for constants, and finally x, y, z with indices for variables. The symbol "t" is reserved for implementation purposes so it cannot be used during the input and serves as a term in application of $\rightarrow\forall$, $\forall\rightarrow$, $\rightarrow\exists$, $\exists\rightarrow$.

For example, an input for JLog should look like the following one:

$$(P(x)>Q(y)),(Q(y)>R(a))->((P(x)|Q(y))>R(a))$$

In sequential systems the proof usually presented by inference tree. JLog stores it as a natural deduction proof with indication of premises and inference rules. The proof of $\rightarrow(A\supset(B\supset(A\&B)))$ sequent given below:

$$\dfrac{\dfrac{\dfrac{B, A\rightarrow B \text{ and } B, A\rightarrow B}{B, A\rightarrow(A\&B)}}{A\rightarrow(B\supset(A\&B))}}{\rightarrow(A\supset(B\supset(A\&B)))}$$

JLog presents that proof in the following format:

1. $\rightarrow(A\supset(B\supset(A\&B)))$
2. $A\rightarrow(B\supset(A\&B))$ (1) $\rightarrow\supset$
3. $B, A\rightarrow(A\&B)$ (2) $\rightarrow\supset$
4. $B, A\rightarrow B$ (3) $\rightarrow\&$ (axiom)
5. $B, A\rightarrow B$ (3) $\rightarrow\&$ (axiom)

REFERENCES

[1] Weidenbach C., Gaede B. and Rock G. Spass and flotter, Version 0.42. Proceedings of the 13th International Conference on Automated Deduction, pp. 141-145, 1996.

[2] Riazanov A. and Voronkov A. Vampire. Proceedings of the 16th International Conference on Automated Deduction, pp. 292-296, 1999.

[3] Nieuwenhuis R., Rivero J. and Vallejo M. The Barcelona Prover. Journal of Automated Reasoning, 18(2), pp. 171-176, 1997.

[4] Bolibekyan H.R., Chubaryan A.A. On some proof systems for I. Johansson's minimal logic of predicates. Proceedings of the Logic Colloquium, p. 56, 2003 (and the Bulletin of Symbolic Logic, 10(2), p.250, 2004).