Proceedings of the 7th International Conference on Applied Informatics
Eger, Hungary, January 28–31, 2007. Vol. 1. pp. 75–81.

Efficient skolemization

Zoltán Lengyel

University of Debrecen, Institute of Informatics
e-mail: lengyelz@inf.unideb.hu

Abstract

One of the most important tasks of each first-order automated theorem prover system is the skolemization. This satisfiably equivalent rewriting method makes it possible to use efficient proving engines like resolution on first-order formulas. Though it is a fundamental step of each algorithm, the resulting skolem formula can be various. Some technics can eliminate false dependencies while others can result in more general formulas. Using such versions of skolemization can strongly reduce the running time of the theorem prover engine. In this paper we study such methods and also investigate the running time and effectiveness of these improved skolemization processes themselves.

Keywords: classical first-order logic, skolemization, binary decision diagram

MSC: 03B10

1. Introduction

In 1920 Thoralf A. Skolem [10] introduced an algorithm to eliminate one type of quantifiers (mostly $\exists$) in a formula and replace the eliminated variables with new function applications. Though it is not an equivalence preserving transformation, it is still very useful in theorem proving since it preserves satisfiability and unsatisfiability. Recently, Skolemization has been an important step in most of the automated first-order theorem prover systems.

The running time of a theorem prover engine strongly depends on the quality of Skolemization. The naive algorithm has poor quality.

Naive Algorithm. We assume, the formula is clear which means each quantifier binds different variables and there is no free variable that is bounded by a quantifier.

1. Prenexing: move every quantifier in front of the formula.

$$(\exists x \forall y R(x,y) \vee \neg \forall z P(z))$$
$$\exists x \forall y \exists z \, (R(x,y) \vee \neg P(z))$$

2. Skolemization: Eliminate all variables, bounded by existential quantifiers, by introducing new Skolem functions (we call 0-ary functions as *Skolem constants*)
$$\forall y \left(R(c_x, y) \vee \neg P(f_z(y))\right)$$

We refer to this algorithm also as Standard Skolemization.

Some improved algorithms work only on formulas in *Negation Normal Form* (NNF). We say a formula is in NNF iff it contains only $\neg \vee \wedge$ operators and $\neg$ occurs only in front of atoms.

This paper is organized as follows: Section 2 introduces improved Skolemization methods. In Section 3 we show a BDD-based method to eliminate most of the false dependencies. Section 4 contains our conclusion.

2. Improved skolemization

First, we investigate improvements only which can generate "easier" sentences for the theorem prover engine. We can achieve shorter running time by eliminating false dependences, by reducing the number of Skolem symbols and by creating more general Skolem formula (i.e. reducing the dependencies between universally quantified variable occurrences by introducing new universally quantified variables).

2.1. Antiprenexing

The prenexing step of the naive algorithm can produce many false dependencies. In the above example, after prenexing we got

$$\exists x \forall y \exists z \left(R(x, y) \vee \neg P(z)\right).$$

So we should replace z with $f_z(y)$, however, it is obvious that z does not depend on y.

Such false dependencies can be generated also during formalizing the problem, so it worths antiprenexing the formula before replacing existential variables. Egly's idea [3] was to move universal quantifiers inward as far as possible whereas move existential quantifiers outward in order to avoid duplication of Skolem functions. As an example consider the formula

$$\forall x \forall y \exists u \forall z \exists w \left((\neg R(x, y) \vee R(x, u) \vee R(a, w)) \wedge \neg R(a, z)\right).$$

Without antiprenexing we have

$$\left((\neg R(x, y) \vee R(x, f_u(x, y)) \vee R(a, f_w(x, y, z))) \wedge \neg R(a, z)\right),$$

while with antiprenexing we get

$$\begin{aligned} &\forall x \left((\forall y \neg R(x, y) \vee \exists u \left(R(x, u) \vee R(a, u)\right)) \wedge \forall z \neg R(a, z)\right), \\ &\quad \left((\neg R(x, y) \vee R(x, f_u(x)) \vee R(a, f_u(x))) \wedge \neg R(a, z)\right). \end{aligned}$$

2.2. Optimized skolemization

Let Δ be a sentence in NNF and $\Theta = \exists x_1 \ldots \exists x_k(\Phi \wedge \Psi)$ be a sub-formula of Δ and assume that $\Delta \models \forall y_1 \ldots \forall y_n \exists x_1 \ldots \exists x_k \Phi$ where $y_1 \ldots y_n$ denote the free variables in Θ. We use the $\Phi_{\{x \mapsto t\}}$ notation for the formula that we get after replacing x free variables with t terms in Φ. Now, we say

$$\forall y_1 \ldots \forall y_n \Phi_{\{x_i \mapsto f_i(y_1,\ldots,y_n)\}} \ \wedge \ \Delta\left[\Theta | \Psi_{\{x_i \mapsto f_i(y_1,\ldots,y_n)\}}\right]$$

is the result of an Optimized Skolemization step on Δ where $f_1, \ldots, f_k$ are new Skolem functions and $\Delta\left[\Theta | \Psi_{\{x_i \mapsto f_i(y_1,\ldots,y_n)\}}\right]$ denotes the formula that we get by replacing Θ with $\Psi_{\{x_i \mapsto f_i(y_1,\ldots,y_n)\}}$ in Δ. In [7], Ohlbach and Weidenbach proved that Δ is satisfiable iff the resulting formula of Optimized Skolemization is satisfiable. Let us have the following sentence (or part of a sentence) as an example:

$$\forall x \forall y \forall z \left(\neg R(x,y) \vee \neg R(x,z) \vee \exists u \left(R(y,u) \wedge R(z,u)\right)\right).$$

With Standard Skolemization we get

$$\left(\neg R(x,y) \vee \neg R(x,z) \vee R(y, f(y,z))\right) \ \wedge \ \left(\neg R(x,y) \vee \neg R(x,z) \vee R(z, f(y,z))\right),$$

but with Optimized Skolemization, assuming that $\forall y \exists u R(y,u)$ is provable from the whole problem, we have

$$R(y, f(y,z)) \ \wedge \ \left(\neg R(x,y) \vee \neg R(x,z) \vee R(z, f(y,z))\right).$$

This method seems to be very efficient but do not forget that we have to use a theorem prover engine to prove the precondition. Systems, that use this algorithm, are defining a time-limit, within the embedded engine has to prove the condition. If it does not meet this threshold, the algorithm go on without applying Optimized Skolemization step on the given subproblem.

2.3. Strong skolemization

First let us introduce the so called *free variable splitting*. We use the $\bar{x}$ notation to refer a list of variables. We say $\langle \bar{z_1}, \bar{z_2}, \ldots, \bar{z_n} \rangle$ is a free variable splitting of $\exists \bar{x} \left(\Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_n\right)$ where use the $\bar{x}$ notation to refer a list of variables. We say $\langle \bar{z_1}, \bar{z_2}, \ldots, \bar{z_n} \rangle$ is a free variable splitting of $\exists \bar{x} \left(\Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_n\right)$ where

$$\bar{z_i} = \mathrm{Var}(\Phi_i) \setminus \bigcup_{k<i} \bar{z_k} \setminus \bar{x}.$$

Now let Δ be a sentence in NNF. We can replace $\exists \bar{x} \left(\Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_n\right)$ sub-formula of Δ with

$$\begin{aligned} &\forall \bar{w_2} \forall \bar{w_3} \ldots \forall \bar{w_n} \ \Phi_{1 \ \{x_i \mapsto f_i(\bar{z_1}, \bar{w_2}, \bar{w_3}, \ldots, \bar{w_n})\}} \wedge \\ &\forall \bar{w_3} \ldots \forall \bar{w_n} \ \Phi_{2 \ \{x_i \mapsto f_i(\bar{z_1}, \bar{z_2}, \bar{w_3}, \ldots, \bar{w_n})\}} \wedge \end{aligned}$$

$$\vdots$$
$$\Phi_{n\ \{x_i \mapsto f_i(\bar{z_1},\bar{z_2},\bar{z_3},\ldots,\bar{z_n})\}}.$$

where f_i are new functions and the cardinality of $\bar{w_i}$ and $\bar{z_i}$ is equal. Nonnengart also proved in his paper [6] that Strong Skolemization preserves satisfiability and unsatisfiability. Let us get back to our example

$$\forall x \forall y \forall z \left(\neg R(x,y) \vee \neg R(x,z) \vee \exists u \left(R(y,u) \wedge R(z,u)\right)\right).$$

With Strong Skolemization we get

$$\left(\neg R(x,y) \vee \neg R(x,z) \vee R(y,f(y,w))\right) \ \wedge \ \left(\neg R(x,y) \vee \neg R(x,z) \vee R(z,f(y,z))\right).$$

The only difference to the result of Standard Skolemization is that we have a new universal variable in the first clause but actually this makes the outcome more general and in our case allows us to apply a condensation step resulting

$$\left(\neg R(x,y) \vee R(y,f(y,w))\right) \ \wedge \ \left(\neg R(x,y) \vee \neg R(x,z) \vee R(z,f(y,z))\right).$$

Though this method is not as effective as the Optimized Skolemization, at least there is no condition of application. Methods, that implement both algorithms, try to apply Optimized Skolemization first and than apply Strong Skolemization where possible.

3. Binary decision diagrams

In 1938, Shannon [9] introduced a tree representation of formulas that was refined in several papers (e.g. Lee [5], Akers [1] and Bryant [2]). We call reduced ordered Shannon-graphs as BDDs. "Reduced" has different meaning in propositional and first-order logic while "ordered" means that we have a total ordering of kernels.

3.1. BDD in propositional logic

In propositional logic, we refer variables as *kernels*. Assuming that there is a total ordering of variables $A_1 < A_2 < \cdots < A_n$, we say $\Phi = \xi \rightarrow \Phi[\xi|1]; \Phi[\xi|0]$ is a BDD of Φ where ξ is the least kernel in Φ. On Figure 1, the truth-table and the BDD representation of $\Phi = (A_1 \wedge (\neg A_2 \vee A_3))$ can be seen (true-branches are marked with solid while false-branches with dashed lines). In propositional logic we have the following reduction steps:

1. *sharing*: we "store" isomorphic subgraphs only once making DAG (Directed Acyclic Graph) from the Shannon-tree,
2. *eliminating*: we remove all non-terminal node that has the same subgraph on its true and false branches.

These two steps with the kernel-ordering ensure that BDD is canonical form in propositional logic.

A_1	A_2	A_3	Φ
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

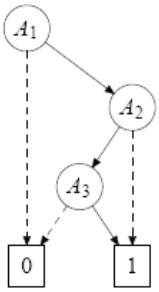

Figure 1: The truth-table and the BDD representation of $(A_1 \wedge (\neg A_2 \vee A_3))$

3.2. BDD in first-order logic

First-order BDDs were defined in 1992 by Posegga [8] as extension of Shannon graphs to first-order logic. We refer predicate symbol applications (e.g. $Q(x, c)$) and existentially quantified formulas (e.g. $\exists x\,(P(x) \vee \neg P(c))$) as kernels. The construction of BDD is the same as in propositional logic, but now, we also build the BDD of Φ for all quantified kernel $\exists x\Phi$ with the following additional rule:

$$\Phi = \xi \rightarrow \Phi\,[\,\exists x\Psi \mid \exists x\Psi[\xi|1]\,]\,;\Phi\,[\,\exists x\Psi \mid \exists x\Psi[\xi|0]\,]\,, \tag{3.1}$$

i.e. we replace the ξ kernel in the nested BDDs as well. We do not consider universal quantifiers since $\forall x\Phi$ can be rewritten to $\neg\exists x\neg\Phi$. In first-order logic we need additional reduction steps:

3. $\Phi \rightsquigarrow \Phi[\exists x1|1]$ and $\Phi \rightsquigarrow \Phi[\exists x0|0]$,
4. $\Phi \rightsquigarrow \Phi[\exists x\Psi|\exists x\Psi']$ if Ψ reduces to Ψ',
5. the already mentioned 3.1 rule.

In first-order logic BDDs are not canonical forms.

Semantic Dependence

As we mentioned above, eliminating false dependencies can reduce the running time of the theorem prover engine, however false dependencies, in some cases, are not "visible". We say Φ *semantically depends* on x iff $\forall x\forall y\left(\Phi \equiv \Phi_{\{x\mapsto y\}}\right)$ is not provable (where y does not occur in Φ) while Φ *semantically independent* of x iff $\forall x\forall y\left(\Phi \equiv \Phi_{\{x\mapsto y\}}\right)$ holds. As an easy example, let us consider the formula $\exists z\,(P(x, z) \vee \neg P(x, z))$ that is semantically independent of x since it is tautology.

3.3. Skolemization with BDD

There are several application areas where BDDs are very useful. One of these areas is the equivalence testing. Goubault [4] shown that we can use BDDs to prove

equivalence of $\exists z\Phi$ and $\exists z\Phi_{\{x\mapsto y\}}$. If these are equivalent, $\exists z\Phi$ is semantically independent of x. Eliminating this false dependence, we can introduce a more general Skolem function. As an example, assume that we have the formula

$$\forall x\exists y(P(y) \wedge (R(x,y) \vee \neg R(x,y))).$$

Then let us build the BDD for $\exists y\Phi$ and $\exists y\Phi_{\{x\mapsto z\}}$ where

$$\Phi = P(y) \wedge (R(x,y) \vee \neg R(x,y)).$$

After applying BDD reduction steps described in [8] we get the same first-order BDD for both formulas, that can be seen on Figure 2. It means that they are equal

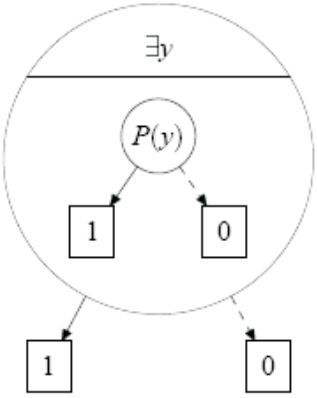

Figure 2: BDD representation of $\exists y\Phi$ and $\exists y\Phi_{\{x\mapsto z\}}$.

and after skolemization we get

$$(P(c_y) \wedge (R(x,c_y) \vee \neg R(x,c_y))),$$

but with converting the BDD back into NNF, we get the more simple formula of $P(c_y)$.

4. Concluding remark

With these methods, we can eliminate most of the false dependencies and we can generate more general Skolem formulas but it does not worth the effort in every case. For example, building the BDD of a formula is a time consuming process but if we have it already, we can eliminate the most of the false dependencies, moreover we can simplify the formula as well.

There are some false dependencies (e.g. $\forall x\Phi \supset \exists x\Phi$) that can not be automatically recognized yet. In the future we will try to find algorithms that can remove all the false dependencies.

Acknowledgements. I want to thank Magda Várterész for her valuable comments.

References

[1] Akers, S. B., Binary Decision Diagrams, *IEEE Transactions on Computers*, 27(6), June (1978), 509–516.

[2] Bryant, R. E., Graph-based Algorithms for Boolean Function Manipulation, *IEEE Transactions on Computers*, 8(35), (1986), 677–691.

[3] Egly, U., On the Value of Antiprenexing, In *LPAR '94: Proceedings of the 5^{th} International Conference on Logic Programming and Automated Reasoning*, Springer Verlag, London, UK, (1994), 69–83.

[4] Goubault, J., A BDD-based simplification and skolemization procedure, *J. of the IGPL*, 3(6), (1995), 827–855.

[5] Lee, C.-Y., Representation of Switching Circuits by Binary-Decision Programs, *Bell System Technical Journal*, 38, July (1959), 985–999.

[6] Nonnengart, A., Strong Skolemization, *Research Report MPI-I-96-2-010*, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, December (1996).

[7] Ohlbach, H. J., Weidenbach, C., A Note on Assumptions about Skolem Functions, *Journal of Automated Reasoning*, 15(2), (1995), 267–275.

[8] Posegga, J., Ludascher, B., Towards First-order Deduction Based on Shannon Graphs, In *GWAI*, (1992), 67–75.

[9] Shannon, C. E., A Symbolic Analysis of Relay and Switching Circuits, *Transactions of the AIEE*, 57, (1938), 713–723.

[10] Skolem, T. A., Logisch-kombinatorische Untersuchungen uber die Erfullbarkeit oder Beweisbarkeit Mathematischer Satze, *Skriftner utgit ar Videnskapsselskaper i Kristiania*, (4), (1920), 4–36.

Zoltán Lengyel
H-4032 Debrecen
Egyetem tér 1.
Hungary