CHAPTER 6

Computing Small Clause Normal Forms

Andreas Nonnengart

Christoph Weidenbach

SECOND READERS: Thierry Boy de la Tour and Uwe Egly.

Contents

HANDBOOK OF AUTOMATED REASONING
Edited by Alan Robinson and Andrei Voronkov

1. Introduction

Automated reasoning systems for first-order predicate logic usually operate on sets of clauses, see [Bachmair and Ganzinger 2001, Fermüller et al. 2001, Nieuwenhuis and Rubio 2001, Weidenbach 2001] (Chapters 2, 25, 7, and 27 of this Handbook). However, many problem formulations, in particular problems from application domains, are given in full first-order logic and thus require a transformation into clause normal form (CNF). It is well-known that the quality of such a translation has a great impact on the success of an afterwards applied automated reasoning system. From this perspective, a CNF of some formula is better than another one if it enables a system to find a proof/counter-model in a shorter period of time. Since this is an undecidable criterion, there cannot be an optimal CNF transformation algorithm in this sense. We employ the following heuristics: The smaller a set of clauses is the easier a proof or a counter-model can be found. Small can relate to different measures like the number of clauses, the number of literals, the length of the clauses or the number of different symbols. For example, in Section 4 we show that the number of generated clauses can be reduced by the introduction of new predicate symbols. There is no definite answer to the question what should be kept minimal. In a propositional context, the introduction of new propositional variables may cause the Davis-Putnam procedure to fail although it succeeds on the original formula. Even if we restrict our attention to one measure, the design of an algorithm producing "smallest" CNFs is non-trivial. For example, the introduction of new predicate symbols may cause simplification procedures to fail that would have significantly reduced the original formula. Nevertheless, experience shows that heading towards small CNFs pays off in practice. For some first-order logic formula with nested equivalences, renaming techniques (Section 4) are mandatory to effectively generate a CNF in practice. This means that there are theorem proving problems where the (doubly) exponential explosion of the number of clauses produced by a standard CNF algorithm without renaming makes the computation of the CNF intractable.

The approach we propose here emphasizes on three major aspects in the CNF transformation of first-order predicate logic formulae. These are *Formula Renaming* (Section 4), *Skolemization* (Section 5) and *Simplification* (Section 6).

By *Renaming* we mean the replacement of subformulae with some new predicates and adding a suitable definition for these new predicates. This is done whenever there is evidence that such a step will ultimately lead to fewer and smaller clauses. It shows that this method can be turned into efficient algorithms and produces smaller clause sets compared with other techniques described in the literature (see Section 7).

Moreover, we propose two advanced *Skolemization* techniques, namely *optimized* and *strong* Skolemization which turned out to be superior to the standard Skolemization approaches, i.e., they usually result in smaller and/or more general clauses. These methods can be turned into algorithms showing a reasonable behavior for all examples we tested (see Section 8).

The article is now organized as follows: after a section on preliminaries (Section 2) where we fix notations and introduce a standard CNF transformation procedure (Section 3), Section 4 is concerned with the renaming of formulae. In Section 5 we discuss Skolemization techniques and present methods to make them tractable in practice. We finish with bibliographic notes (Section 7) and implementation notes (Section 8).

2. Preliminaries

A first-order language is constructed over a signature $\Sigma = (\mathcal{F}, \mathcal{R})$, where $\mathcal{F}$ and $\mathcal{R}$ are non-empty, disjoint, in general infinite sets of function and predicate symbols, respectively. Every function or predicate symbol has some fixed arity. In addition to these sets that are specific for a first-order language, we assume a further, infinite set $\mathcal{X}$ of variable symbols disjoint from the symbols in Σ. Then the set of all *terms* $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is recursively defined by: (i) every *constant* symbol $c \in \mathcal{F}$ with arity zero is a term, (ii) every variable $x \in \mathcal{X}$ is a term and (iii) whenever $t_1, \ldots, t_n$ are terms and $f \in \mathcal{F}$ is a function symbol with arity n, then $f(t_1, \ldots, t_n)$ is a term. If $t_1, \ldots, t_n$ are terms and $R \in \mathcal{R}$ is a predicate symbol with arity n, then $R(t_1, \ldots, t_n)$ is an *atom*. We recursively construct *formulae* over atoms, the logical constants $\top$ (truth), $\bot$ (falsity) and the operators $\supset$ (implication), $\equiv$ (equivalence), $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation) and the quantifiers $\forall$ (universal), $\exists$ (existential) as usual: (i) every atom is a formula, (ii) $\top$ and $\bot$ are formulae. (iii) if ϕ_1 and ϕ_2 are formulae, so are $\phi_1 \equiv \phi_2$, $\phi_1 \supset \phi_2$, $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$ and $\neg\phi_1$ and (iv) if ϕ is a formula and $x \in \mathcal{X}$ a variable, then $\forall x\, \phi$ and $\exists x\, \phi$ are formulae. An atom or the negation of an atom is called a *literal*. For convenience, we often write $\forall x_1, \ldots, x_n\, \phi$ instead of $\forall x_1 \ldots \forall x_n\, \phi$ and analogously for the existential quantifier. For the sequence $x_1, \ldots, x_n$ $(t_1, \ldots, t_n)$ we use the abbreviation $\overline{x}_n$ $(\overline{t}_n)$. Furthermore, for associative operators we often omit parentheses as in $\phi_1 \vee \phi_2 \vee \phi_3$. Disjunctions of literals are *clauses* where all variables are implicitly universally quantified. Clauses are often denoted by their respective multisets of literals where we write multisets in usual set notation.

An *interpretation* is a triple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \nu \rangle$ where $\mathcal{D}$ is a non-empty set, the domain of discourse, $\mathcal{I}$ associates n-ary predicate symbols from $\mathcal{R}$ and function symbols from $\mathcal{F}$ with n-place relations and functions respectively. The function $\nu : \mathcal{X} \mapsto \mathcal{D}$ is a variable valuation that associates variable symbols with concrete values taken from $\mathcal{D}$. By $\nu[x/a]$ we denote the variable valuation that is exactly like ν except that it maps the variable x to the domain value a. We sometimes use $\mathcal{M}[x/a]$ as a short form for $\langle \mathcal{D}, \mathcal{I}, \nu[x/a] \rangle$ provided $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \nu \rangle$. Given an interpretation $\mathcal{M}$ and a term t we define the interpretation of t with respect to $\mathcal{M}$ – denoted by $\mathcal{M}(t)$ – recursively as (i) $\mathcal{M}(x) = \nu(x)$, (ii) $\mathcal{M}(f(t_1, \ldots, t_n)) = \mathcal{I}(f)(\mathcal{M}(t_1), \ldots, \mathcal{M}(t_n))$. An interpretation $\mathcal{M}$ *satisfies* a formula ϕ if and only if $\mathcal{M} \models \phi$, where the relation $\models$ is defined recursively as follows

$$
\begin{array}{lll}
\mathcal{M} \models \top, \quad \mathcal{M} \not\models \bot & & \\
\mathcal{M} \models R(t_1, \ldots, t_n) & \text{iff} & (\mathcal{M}(t_1), \ldots, \mathcal{M}(t_n)) \in \mathcal{I}(R)
\end{array}
$$

$$\begin{array}{lll} \mathcal{M} \models \neg\phi & \text{iff} & \mathcal{M} \not\models \phi \\ \mathcal{M} \models \phi \wedge \psi & \text{iff} & \mathcal{M} \models \phi \text{ and } \mathcal{M} \models \psi \\ \mathcal{M} \models \forall x\, \phi & \text{iff} & \mathcal{M}[x/a] \models \phi \quad \text{for every } a \in \mathcal{D} \\ \mathcal{M} \models \exists x\, \phi & \text{iff} & \mathcal{M}[x/a] \models \phi \quad \text{for some } a \in \mathcal{D} \end{array}$$

As usual $\phi_1 \vee \phi_2$ is interpreted as $\neg(\neg\phi_1 \wedge \neg\phi_2)$, $\phi_1 \supset \phi_2$ is interpreted as $\neg\phi_1 \vee \phi_2$ and $\phi_1 \equiv \phi_2$ is interpreted as $(\phi_1 \supset \phi_2) \wedge (\phi_2 \supset \phi_1)$.

In case there exists an interpretation $\mathcal{M}$ that satisfies ϕ, we say that ϕ is *satisfiable* and that $\mathcal{M}$ is a *model* of ϕ. If every interpretation satisfies ϕ then ϕ is called *valid* and we write $\models \phi$. If a formula ϕ is transformed into a formula ψ by some operation (rule), we say that such a transformation *preserves satisfiability* if ϕ is satisfiable iff ψ is. We say that such an operation *preserves equivalence* if for any interpretation $\mathcal{M}$ we have $\mathcal{M} \models \phi$ iff $\mathcal{M} \models \psi$. Note the subtle difference between these definitions. For an operation to be satisfiability preserving only the existence of a (possibly different) model is required whereas an equivalence preserving operation guarantees satisfiability in the very same model.

A *substitution* σ is a mapping from the set of variables to the set of terms such that $x\sigma \neq x$ for only finitely many $x \in \mathcal{X}$. We define the *domain* of σ to be $dom(\sigma) = \{x \mid x\sigma \neq x\}$ and the co-domain of σ to be $cdom(\sigma) = \{x\sigma \mid x\sigma \neq x\}$. Hence, we can denote a substitution σ by the finite set $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where $x_i\sigma = t_i$ and $dom(\sigma) = \{x_1, \ldots, x_n\}$. An injective substitution σ where $cdom(\sigma) \subset \mathcal{X}$ is called a *variable renaming*. The application of substitutions to terms is given by $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$ for all $f \in \mathcal{F}$ with arity n. For the purpose of this paper, we extend substitutions to formulae as follows: $P(t_1, \ldots, t_n)\sigma = P(t_1\sigma, \ldots, t_n\sigma)$, $(\neg\phi)\sigma = \neg(\phi\sigma)$, $(\phi_1 \circ \phi_2)\sigma = \phi_1\sigma \circ \phi_2\sigma$ where $\circ \in \{\supset, \equiv, \wedge, \vee\}$, $(\forall x\, \phi)\sigma = \forall x\sigma\, \phi\sigma$ if $x\sigma \in \mathcal{X}$ and $\forall x\, \phi$ otherwise, $(\exists x\, \phi)\sigma = \exists x\sigma\, \phi\sigma$ if $x\sigma \in \mathcal{X}$ and $\exists x\, \phi$ otherwise. In general, the extension of substitutions to formulae is problematic, because it might accidentally turn free variables into bound variables. However, we shall only make use of this definition in very restricted contexts where the desired semantics is preserved by the application of the substitution.

A clause C is said to *subsume* a clause D, if there exists a substitution σ with $C\sigma \subseteq D$. A clause C is a *condensation* of a clause D if there exists a substitution σ where $C \subset D\sigma$, C is obtained from $D\sigma$ by the deletion of multiple occurrences of identical literals and C subsumes D. Equivalently we could say that C is a condensation of a clause D if C is a proper (multiple) factor (see [Weidenbach 2001], Chapter 27 of this Handbook) of D that subsumes D. Note that we consider clauses to be multisets of literals and not sets. Hence, the clause $\{R(x,x), R(y,y)\}$ does not subsume the clause $\{R(a,a)\}$.

A *position* is a word over the natural numbers. The set $pos(\phi)$ of positions of a given formula ϕ is defined as follows: (i) the empty word $\epsilon \in pos(\phi)$ (ii) $i.\pi \in pos(\phi)$ if $\phi = \phi_1 \circ \phi_2$ and $\pi \in pos(\phi_i)$, $i \in \{1,2\}$ where $\circ \in \{\supset, \equiv, \wedge, \vee\}$ and (iii) $1.\pi \in pos(\phi)$ if $\phi = \neg\psi$ or $\phi = \forall x\, \psi$ or $\phi = \exists x\, \psi$ and $\pi \in pos(\psi)$. Now, if $\pi \in pos(\phi)$ we define $\phi|_\epsilon = \phi$ and $\phi|_{i.\tau} = \phi_i|_\tau$ where $\phi = \phi_1 \circ \phi_2$, $\circ \in \{\supset, \equiv, \wedge, \vee\}$ and define $\phi|_{1.\tau} = \psi|_\tau$ if $\phi = \neg\psi$ or $\phi = \forall x\, \psi$ or $\phi = \exists x\, \psi$. We write $\psi[\phi]_\pi$ for $\psi|_\pi = \phi$. With $\psi[\pi/\phi]$, where $\pi \in pos(\psi)$, we denote the formula obtained by replacing $\psi|_\pi$ with ϕ

at position π in ψ. The *length* of a position π is defined by $|\epsilon| = 0$ and $|i.\tau| = 1 + |\tau|$. Let $\leq$ denote the usual prefix ordering on positions: $\pi \leq \tau$ if there exists a position ι with $\pi.\iota = \tau$.

The polarity of a formula occurring at position π in a formula ψ is denoted by $pol(\psi, \pi)$ and is defined in the usual way: $pol(\psi, \epsilon) = 1$; $pol(\psi, \pi.i) = pol(\psi, \pi)$ if $\psi|_\pi$ is a conjunction, disjunction, formula starting with a quantifier or an implication with $i = 2$; $pol(\psi, \pi.i) = -pol(\psi, \pi)$ if $\psi|_\pi$ is a formula starting with a negation symbol or an implication with $i = 1$ and, finally, $pol(\psi, \pi.i) = 0$ if $\psi\,|_\pi$ is an equivalence.

The *depth* of a formula ϕ (term t) is given by $depth(\phi) = max(\{|\pi| \mid \pi \in pos(\phi)\})$ ($depth(t) = max(\{|\pi| \mid \pi \in pos(t)\})$.

The set of *free* variables of a formula ϕ (term t), denoted by $free(\phi)$ is defined as follows: $free(P(t_1, \ldots, t_n)) = \cup_i free(t_i)$, $free(\neg\phi) = free(\phi)$, $free(\phi_1 \circ \phi_2) = free(\phi_1) \cup free(\phi_2)$ where $\circ \in \{\supset, \equiv, \wedge, \vee\}$, $free(\forall x\, \phi) = free(\phi) \setminus \{x\}$, $free(\exists x\, \phi) = free(\phi) \setminus \{x\}$ ($free(f(t_1, \ldots, t_n)) = \cup_i free(t_i)$, $free(x) = \{x\}$). A formula ϕ is a *sentence* if ϕ does not contain any free variables ($free(\phi) = \emptyset$). The set of *bound* variables of a formula ϕ (term t), denoted by $bound(\phi)$ is defined as follows: $bound(P(t_1, \ldots, t_n)) = \emptyset$, $bound(\neg\phi) = bound(\phi)$, $bound(\phi_1 \circ \phi_2) = bound(\phi_1) \cup bound(\phi_2)$ where $\circ \in \{\supset, \equiv, \wedge, \vee\}$, $bound(\forall x\, \phi) = bound(\phi) \cup \{x\}$, $bound(\exists x\, \phi) = bound(\phi) \cup \{x\}$ ($bound(t) = \emptyset$). Finally, the set of all variables of a formula ϕ (term t) is defined by $vars(\phi) = bound(\phi) \cup free(\phi)$ ($vars(t) = free(t)$).

3. Standard CNF-Translation

In this section we introduce a standard way of translating first-order formulae into clause normal form. Standard means that algorithms similar to the one suggested here can be found in many textbooks on automated theorem proving or first-order logic. There are many useful techniques beyond the material presented in this chapter. In Section 4, Section 5 and Section 6 we suggest some important extensions. For further material consider the references presented in Section 7.

A sentence ϕ is in *clause normal form* (CNF), if $\phi = \forall x_1 \ldots \forall x_k\, [C_1 \wedge \ldots \wedge C_n]$ where $C_i = L_{i,1} \vee \ldots \vee L_{i,l_i}$ and each $L_{i,j}$ is a literal. An alternative representation of a CNF is a set of clauses, where variables in the clauses are implicitly universally quantified, their literals disjunctively connected and all clauses form a conjunction. Clause normal form translation converts an arbitrary formula into a CNF, preserving satisfiability. Thus, important properties of the CNF-translation algorithm are termination, that it preserves satisfiability and that it generates CNFs.

Main parts of the algorithm are presented by collections of rules. The rules are either of the form $\phi_1 \to \phi_2$ or $\phi[\phi_1]_\pi \to \phi[\pi/\phi_2]$, where the former is an abbreviation for the latter, in case we are not interested in the particular position. Informally, these rules allow us to replace the subformula ϕ_1 (occurring at position π) with the formula ϕ_2 within some arbitrary formula ϕ. Collections of rules are applied in an exhaustive, don't-care non-deterministic way. Furthermore, formula matching is performed modulo the commutativity of $\wedge$, $\vee$ and $\equiv$.

$\phi \wedge \phi$	$\to$	ϕ	$\phi \vee \phi$	$\to$	ϕ
$\phi \wedge \neg\phi$	$\to$	$\bot$	$\phi \vee \neg\phi$	$\to$	$\top$
$\phi \equiv \phi$	$\to$	$\top$	$\phi \supset \phi$	$\to$	$\top$
$\neg\top$	$\to$	$\bot$	$\neg\bot$	$\to$	$\top$
$\phi \wedge \top$	$\to$	ϕ	$\phi \vee \top$	$\to$	$\top$
$\phi \wedge \bot$	$\to$	$\bot$	$\phi \vee \bot$	$\to$	ϕ
$\phi \supset \top$	$\to$	$\top$	$\top \supset \phi$	$\to$	ϕ
$\phi \supset \bot$	$\to$	$\neg\phi$	$\bot \supset \phi$	$\to$	$\top$
$\phi \equiv \top$	$\to$	ϕ	$\phi \equiv \bot$	$\to$	$\neg\phi$

$$\begin{array}{rcll} \forall x\, \phi & \to & \phi & \text{if } x \notin \mathit{free}(\phi) \\ \exists x\, \phi & \to & \phi & \text{if } x \notin \mathit{free}(\phi) \end{array}$$

Table 1: Simplification Rules

The standard CNF-translation algorithm is divided into seven subsequent steps represented by the Subsections 3.1–3.7 below.

3.1. Formula Simplification

The collection in Table 1 removes the logical constants $\top$, $\bot$ from positions below ϵ. Also it removes redundant quantifiers and performs some first-order simplifications like the application of idempotence of $\wedge$, $\vee$ or the removal of certain syntactic tautologies.

The application of the rules terminates, since every right hand side of a rule contains fewer symbols than its left hand side. All these rules are equivalence preserving. After an exhaustive application of the collection to a formula resulting in ϕ, either $\phi \in \{\top, \bot\}$ or ϕ does not contain $\top$ nor $\bot$ and every quantifier in ϕ binds a variable that occurs freely in its argument formula.

If a formula is reduced to $\top$ or $\bot$ by the above rules, the CNF translation stops here and we are already done. Hence, for further considerations we assume that formulae contain neither $\top$ nor $\bot$.

3.2. Negation Normal Form

A formula is in *negation normal form* (NNF), if it does not contain implication or equivalence symbols, and every negation symbol occurs directly in front of an atom. A transformation into NNF can be performed with the help of the rules shown in Table 2

$\psi[\phi_1 \equiv \phi_2]_\pi$	$\to$	$\psi[\pi/(\phi_1 \supset \phi_2) \wedge (\phi_2 \supset \phi_1)]$	if $pol(\psi,\pi) = 1$
$\psi[\phi_1 \equiv \phi_2]_\pi$	$\to$	$\psi[\pi/(\phi_1 \wedge \phi_2) \vee (\neg\phi_1 \wedge \neg\phi_2)]$	if $pol(\psi,\pi) = -1$

$\neg(\phi \wedge \psi)$	$\to$	$\neg\phi \vee \neg\psi$	$\neg(\phi \vee \phi)$	$\to$	$\neg\phi \wedge \neg\psi$
$\neg(\forall x\, \phi)$	$\to$	$\exists x\, \neg\phi$	$\neg(\exists x\, \phi)$	$\to$	$\forall x\, \neg\phi$
$\phi \supset \psi$	$\to$	$\neg\phi \vee \psi$	$\neg\neg\phi$	$\to$	ϕ

Table 2: Transformation into NNF

The rules terminate, since they either decrease the number of equivalence or implication symbols or they decrease the depth of formulae that start with a negation symbol. All these rules are equivalence preserving. They transform any formula ϕ into a NNF equivalent to ϕ.

Note that for the elimination of equivalences, we need not consider positions with polarity 0, since we can always apply these rules at a position with minimal length. The formulae at such positions always have polarity 1 or -1. The suggested transformation is called *polarity dependent* elimination of equivalence symbols. This elimination avoids generating redundant clauses that are hardly recognizable once the CNF is built: Assume that we always apply the first variant $\phi_1 \equiv \phi_2 \to (\phi_1 \supset \phi_2) \wedge (\phi_2 \supset \phi_1)$ to replace equivalence symbols, independently from the polarity of the formula. Then consider the following transformation of $\neg(\phi_1 \equiv \phi_2)$

$$
\begin{gathered}
\neg(\phi_1 \equiv \phi_2) \\
\downarrow \\
\neg((\phi_1 \supset \phi_2) \wedge (\phi_2 \supset \phi_1)) \\
\downarrow \\
\neg((\neg\phi_1 \vee \phi_2) \wedge (\neg\phi_2 \vee \phi_1)) \\
\downarrow \\
\neg(\neg\phi_1 \vee \phi_2) \vee \neg(\neg\phi_2 \vee \phi_1) \\
\downarrow \\
(\phi_1 \wedge \neg\phi_2) \vee (\phi_2 \wedge \neg\phi_1) \\
\downarrow \\
(\phi_1 \vee \phi_2) \wedge (\phi_1 \vee \neg\phi_1) \wedge (\neg\phi_2 \vee \phi_2) \wedge (\neg\phi_2 \vee \neg\phi_1)
\end{gathered}
$$

where we used the rules for NNF and finally the distributive law to obtain the correct shape of a CNF (see also Section 3.6). Now the disjuncts $(\phi_1 \vee \neg\phi_1)$ and $(\neg\phi_2 \vee \phi_2)$ are tautologies (see Section 3.1) and can therefore be removed. However, if ϕ_1 and ϕ_2 are complex formulae this may not be easily detectable, e.g., $\neg\phi_1$ is transformed by the above rules into a formula that is no longer syntactically the negation of ϕ_1. This problem is completely avoided by the polarity dependent elimination suggested above, since it will not generate any of these tautologies. We shall make use of this particular transformation in our section on formula renaming (Section 4).

3.3. Miniscoping

In Section 3.5 below, we shall get rid of existential quantifiers via the introduction of Skolem functions. The arity of a Skolem function is determined by the number of universally quantified variables that appear in the argument formula of the existential quantifier to be removed. The aim of the rules here is to minimize the arity of Skolem functions by moving quantifiers as far inwards as possible. So this step is only for Skolem function argument number optimization purposes.

$$
\begin{array}{rcll}
\exists x\,(\phi \wedge \psi) & \to & \exists x\,\phi \wedge \psi & \text{if } x \notin \mathit{free}(\psi)\\
\exists x\,(\phi \vee \psi) & \to & \exists x\,\phi \vee \psi & \text{if } x \notin \mathit{free}(\psi)\\
\forall x\,(\phi \wedge \psi) & \to & \forall x\,\phi \wedge \psi & \text{if } x \notin \mathit{free}(\psi)\\
\forall x\,(\phi \vee \psi) & \to & \forall x\,\phi \vee \psi & \text{if } x \notin \mathit{free}(\psi)\\
\\
\forall x\,(\phi \wedge \psi) & \to & \forall x\,\phi \wedge \forall x\,\psi & \text{if } x \in \mathit{free}(\phi) \text{ and } x \in \mathit{free}(\psi)\\
\exists x\,(\phi \vee \psi) & \to & \exists x\,\phi \vee \exists x\,\psi & \text{if } x \in \mathit{free}(\phi) \text{ and } x \in \mathit{free}(\psi)
\end{array}
$$

All rules are equivalence preserving. They terminate, since they always reduce the depth of a formula starting with a quantifier.

Here is an example that demonstrates the potential of this transformation:

$$
\begin{array}{c}
\forall x\,\exists y\,\forall z\,(R(x,x) \wedge (P(y) \vee R(x,y) \vee Q(z)))\\
\downarrow\\
\forall x\,\exists y\,(R(x,x) \wedge \forall z\,(P(y) \vee R(x,y) \vee Q(z)))\\
\downarrow\\
\forall x\,\exists y\,(R(x,x) \wedge (P(y) \vee R(x,y) \vee \forall z\,Q(z)))\\
\downarrow\\
\forall x\,(R(x,x) \wedge \exists y\,(P(y) \vee R(x,y) \vee \forall z\,Q(z)))\\
\downarrow\\
\forall x\,(R(x,x) \wedge (\exists y\,P(y) \vee \exists y\,R(x,y) \vee \forall z\,Q(z)))\\
\downarrow\\
\forall x\,R(x,x) \wedge \forall x\,(\exists y\,P(y) \vee \exists y\,R(x,y) \vee \forall z\,Q(z))\\
\downarrow\\
\forall x\,R(x,x) \wedge (\exists y\,P(y) \vee \forall x\,\exists y\,R(x,y) \vee \forall z\,Q(z))
\end{array}
$$

In Section 3.5 we continue the example showing that this transformation enables the Skolemization rule to generate Skolem functions with fewer arguments.

3.4. Variable Renaming

Miniscoping duplicates quantifiers. Since we want to eventually move all remaining universal quantifiers completely outwards, we have to rename variables such that different occurrences of quantifiers bind different variables. Moreover, our Skolemization rule does not work properly if different quantifiers bind the same variable. Variable renaming is well-known to be an error-prone subject. Therefore, we provide a formal definition and a formal proof for the properties enjoyed by the variable renaming procedure.

Let ψ be a formula and $\overline{y} = y_1, y_2, \ldots$ be an infinite sequence of different variables that do not occur in ψ. Then we recursively define the variable renaming function $ren(\psi) = ren(\psi, \overline{y}, 1)$ by

$$ren(\psi,\overline{y},n) = \begin{cases} P(t_1,\ldots,t_n) & \text{if } \psi = P(t_1,\ldots,t_n) \\ ren(\phi_1,\overline{y},n) \circ ren(\phi_2,\overline{y},n+nq(\phi_1)) & \text{if } \psi = \phi_1 \circ \phi_2 \\ \neg ren(\phi,\overline{y},n) & \text{if } \psi = \neg\phi \\ \exists y_n\, ren(\phi\{x \mapsto y_n\},\overline{y},n+1) & \text{if } \psi = \exists x\,\phi \\ \forall y_n\, ren(\phi\{x \mapsto y_n\},\overline{y},n+1) & \text{if } \psi = \forall x\,\phi \end{cases}$$

where for the second case $\circ \in \{\supset, \equiv, \wedge, \vee\}$. The function nq counts the number of quantifiers in a formula: $nq(\psi) = |\{\pi \mid \psi|_\pi = \exists x\,\phi \text{ or } \psi|_\pi = \forall x\,\phi\}|$.

The renaming of variables is a subtle issue. Therefore, we now prove in detail that the variable renaming function *ren* generates the desired result. In order to prove the logical equivalence between a formula and a renamed variant, we make use of the lemma below.

3.1. LEMMA. *Let σ be a variable renaming and let ϕ be a formula with $cdom(\sigma) \cap vars(\phi) = \emptyset$ Moreover, let ν_1 and ν_2 be two variable valuations satisfying $\nu_1(x) = \nu_2(x\sigma)$ for all $x \in free(\phi)$. Then*

$$\langle \mathcal{D},\mathcal{I},\nu_1\rangle \models \phi \text{ iff } \langle \mathcal{D},\mathcal{I},\nu_2\rangle \models \phi\sigma$$

PROOF. By induction on the structure of formulae. Note that if $cdom(\sigma) \cap vars(\phi) = \emptyset$ then $cdom(\sigma) \cap vars(\psi) = \emptyset$ and $cdom(\sigma) \cap vars(t) = \emptyset$ for any subformula ψ of ϕ and any term t occurring in ϕ. For the purpose of this proof we say that two valuations ν_1 and ν_2 agree modulo a variable renaming σ, if $\nu_1(x) = \nu_2(x\sigma)$ for all x. First, we prove by structural induction that $\langle \mathcal{D},\mathcal{I},\nu_1\rangle(t) = \langle \mathcal{D},\mathcal{I},\nu_2\rangle(t\sigma)$ for all terms t. If t is a variable y, then $\langle \mathcal{D},\mathcal{I},\nu_1\rangle(y) = \nu_1(y) = \nu_2(y\sigma) = \langle \mathcal{D},\mathcal{I},\nu_2\rangle(y\sigma)$ since y occurs free in y. If t is a constant c, then $\langle \mathcal{D},\mathcal{I},\nu_1\rangle(c) = \mathcal{I}(c) = \langle \mathcal{D},\mathcal{I},\nu_2\rangle(c\sigma)$. Finally, if t is a compound term $f(t_1,\ldots,t_n)$ then since ν_1 and ν_2 agree on the free variables of t modulo σ, the two valuations agree on all variables of the t_i modulo σ. Therefore, we conclude $\langle \mathcal{D},\mathcal{I},\nu_1\rangle(f(t_1,\ldots,t_n)) = \mathcal{I}(f)(\langle \mathcal{D},\mathcal{I},\nu_1\rangle(t_1),\ldots,\langle \mathcal{D},\mathcal{I},\nu_1\rangle(t_n)) = \mathcal{I}(f)(\langle \mathcal{D},\mathcal{I},\nu_2\rangle(t_1\sigma),\ldots,\langle \mathcal{D},\mathcal{I},\nu_2\rangle(t_n\sigma)) = \langle \mathcal{D},\mathcal{I},\nu_2\rangle(f(t_1,\ldots,t_n)\sigma)$ by definition and by applying the induction hypothesis to the t_i.

Second, we prove the main conjecture by structural induction on formulae. For an atom $P(t_1,\ldots,t_n)$, if ν_1 and ν_2 agree on the free variables of $P(t_1,\ldots,t_n)$ modulo σ, they agree on all free variables of the t_i modulo σ. We conclude $\langle\mathcal{D},\mathcal{I},\nu_1\rangle \models P(t_1,\ldots,t_n)$ iff $\mathcal{I}(P)(\langle\mathcal{D},\mathcal{I},\nu_1\rangle(t_1),\ldots,\langle\mathcal{D},\mathcal{I},\nu_1\rangle(t_n))$ iff $\mathcal{I}(P)(\langle\mathcal{D},\mathcal{I},\nu_2\rangle(t_1\sigma),\ldots,\langle\mathcal{D},\mathcal{I},\nu_2\rangle(t_n\sigma))$ iff $\langle\mathcal{D},\mathcal{I},\nu_2\rangle \models P(t_1,\ldots,t_n)\sigma$ by definition and by induction hypothesis applied to the t_i. The cases for all first-order operators except for the quantifiers are similar and are therefore omitted. For the quantifiers we concentrate on the existential quantifier. The treatment of the universal quantifier works analogous and is also omitted. We have to show that $\langle\mathcal{D},\mathcal{I},\nu_1\rangle \models \exists x\,\phi$ iff $\langle\mathcal{D},\mathcal{I},\nu_2\rangle \models (\exists x\,\phi)\sigma$. If $x \notin dom(\sigma)$, then $(\exists x\,\phi)\sigma = \exists x\,\phi\sigma$. For any $a \in \mathcal{D}$ the valuations $\nu_1[x/a]$ and $\nu_2[x/a]$ agree on all free variables of ϕ since they agree on all free variables of $\exists x\,\phi$ and obviously they agree on x. Hence, for any $a \in \mathcal{D}$ we have $\langle\mathcal{D},\mathcal{I},\nu_1[x/a]\rangle \models \phi$ iff $\langle\mathcal{D},\mathcal{I},\nu_2[x/a]\rangle \models \phi\sigma$ by induction hypothesis, implying the conjecture for the case $x \notin dom(\sigma)$. If $x \in dom(\sigma)$, where $x\sigma = y$, then by definition $(\exists x\,\phi)\sigma = \exists y\,\phi\sigma$. If ν_1, ν_2 agree on the free variables of $\exists x\,\phi$, the valuations $\nu_1[x/a]$, $\nu_2[y/a]$ agree on the free variables of ϕ modulo σ for any $a \in \mathcal{D}$ because $y \notin vars(\exists x\,\phi)$ and σ is injective. Therefore, by the induction hypothesis, we get $\langle\mathcal{D},\mathcal{I},\nu_1[x/a]\rangle \models \phi$ iff $\langle\mathcal{D},\mathcal{I},\nu_2[y/a]\rangle \models \phi\sigma$. This implies the main conjecture and we are done. $\square$

3.2. LEMMA. *The function ren terminates, there are no two quantifiers in ren*(Ψ) *binding the same variable and the formulae ren*(Ψ) *and* Ψ *are logically equivalent.*

PROOF. First, all recursive calls of *ren* apply to proper subformulae. This guarantees termination. Second, the function renames exactly the variables bound by quantifiers. The cases four and five of the definition introduce a fresh variable y_n that will not be used in the recursive call, since n is incremented. Using an induction argument on the definition of *ren*, this property is preserved. In particular, for the second case where *ren* distributes over a first-order operator, in the recursive call for ϕ_2 the parameter n is incremented by the number of quantifier occurrences in ϕ_1 ensuring that different y_i are introduced for the two subformulae. Third, logical equivalence follows by an inductive argument on the definition of *ren* using Lemma 3.1. $\square$

Renaming the variables of

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

(see Section 3.3) with respect to the sequence $y_1, y_2, \ldots$ results in

$$\forall y_1\, R(y_1,y_1) \wedge (\exists y_2\, P(y_2) \vee \forall y_3\, \exists y_4\, R(y_3,y_4) \vee \forall y_5\, Q(y_5))$$

3.5. Standard Skolemization

In this step we get rid of the existential quantifiers. Note that the rule below eliminates existential quantifiers in an outermost way, since the position of the subformula under consideration is always required to be of minimal length. In addition, the function f is assumed to be new with respect to the symbols occurring in the overall formula ϕ.

$$\begin{array}{|lll|}\hline \phi[\exists x\,\psi]_\pi & \to & \phi[\pi/\psi\{x\mapsto f(y_1,\ldots,y_n)\}] \\ & & \text{if } \mathit{free}(\exists x\,\psi)=\{y_1,\ldots,y_n\},\ f \text{ new},\ |\pi| \text{ minimal} \\ \hline\end{array}$$

Skolemization terminates because every rule application removes one of the existential quantifiers. The result of a rule application is not logically equivalent to the formula the rule is applied to, but preserves satisfiability. In Section 5 we will prove this property for two advanced techniques that both subsume the above Skolemization rule.

Applying the Skolemization rule twice to our example results in the formula

$$\forall y_1\, R(y_1,y_1) \wedge (P(a) \vee \forall y_3\, R(y_3, f(y_3)) \vee \forall y_5\, Q(y_5))$$

where a is a Skolem constant and f a Skolem function. Note that without miniscoping (Section 3.3) we would have obtained $P(f(y_1))$ instead of $P(a)$. This example nicely shows that it is a priori not known whether miniscoping is a useful operation in the sense that it supports the search of an afterwards applied proof search procedure, like resolution. If only a is used in a proof, then the miniscoping might have reduced proof complexity, given the Herbrand complexity measures introduced in [Baaz et al. 2001] (Chapter 5 of this Handbook), since we could replace a one place function by a constant. If both a and f are needed for a proof, then miniscoping might have increased proof complexity, since it caused the generation of two Skolem functions where without miniscoping only one is generated.

3.6. Clause Normal Form

Finally, we move all universal quantifiers outwards and compute a conjunctive normal form.

$$\begin{array}{|rcll|}\hline \forall x\,\phi \wedge \psi & \to & \forall x\,(\phi\wedge\psi) & \text{if } x\notin \mathit{free}(\psi) \\ \forall x\,\phi \vee \psi & \to & \forall x\,(\phi\vee\psi) & \text{if } x\notin \mathit{free}(\psi) \\ \phi\vee(\psi_1\wedge\psi_2) & \to & (\phi\vee\psi_1)\wedge(\phi\vee\psi_2) & \\ \hline\end{array}$$

All three rules are equivalence preserving and terminating since they either decrease the length of subformulae positions starting with a quantifier or the number of conjunctions that occur below the position of a disjunction.

For our running example this means

$$\begin{array}{c} \forall y_1\, R(y_1,y_1) \wedge (P(a) \vee \forall y_3\, R(y_3, f(y_3)) \vee \forall y_5\, Q(y_5)) \\ \downarrow \\ \forall y_1, y_3, y_5\, (R(y_1,y_1) \wedge (P(a) \vee R(y_3, f(y_3)) \vee Q(y_5))) \\ \downarrow \\ \forall y_1, y_3, y_5\, ((R(y_1,y_1) \vee P(a)) \wedge (R(y_1,y_1) \vee R(y_3, f(y_3))) \wedge (R(y_1,y_1) \vee Q(y_5))) \end{array}$$

where we left out some intermediate steps.

3.7. Clause Simplification

Finally, we simply remove the universal quantifiers and turn our conjunction of disjunctions of literals into a clause set. We always assume that all variables are universally quantified and that different clauses do not share any variables.

The final formula of the previous step yields the clause set

$$\{\{R(y_1, y_1), P(a)\}, \{R(y_6, y_6), R(y_3, f(y_3))\}, \{R(y_7, y_7), Q(y_5)\}\}$$

where we renamed occurrences of y_1 with y_6 and y_7, respectively.

There is a bunch of well-known simplifications on the clause level, see [Bachmair and Ganzinger 2001, Nieuwenhuis and Rubio 2001, Weidenbach 2001] (Chapters 2, 7 and 27 of this volume). Standard techniques are the removal of subsumed clauses and tautologies. A clause is called a (syntactic) *tautology* if it contains both some atom and its negation. Subsumption as well as tautology deletion are equivalence preserving and can only be applied finitely often to a finite clause set.

4. Formula Renaming

Let us illustrate the problem with the help of a simple example. Consider the formula

$$\phi_1 \vee \forall x\, \phi_2$$

where we assume that x is the only free variable in ϕ_2. If n is the number of clauses generated by ϕ_1 and m is the number of clauses generated by $\forall x\, \phi_2$ then the above formula generates nm clauses. Thus, a nesting of such disjunctions easily leads to an exponential increase in the number of clauses. The same holds for nested implications and equivalences. The reason for this exponential explosion is the (exponential) duplication of subformulae obtained by the exhaustive application of the distributivity law. A solution to this problem is *Formula Renaming*, i.e., the replacement of subformulae using new predicates. For the above formula, a renaming of ϕ_2 is

$$[\phi_1 \vee \forall x\, P(x)] \wedge \forall x\, (P(x) \supset \phi_2)$$

where P is a new one-place predicate. The renamed formula is not logically equivalent to the first formula, but preserves satisfiability and this suffices for resolution-based theorem proving. Furthermore, the renamed formula generates only $n + m$ clauses. Thus, using a renaming technique, the worst case exponential explosion of the number of clauses generated by a formula can be avoided.

4.1. DEFINITION *(Formula Renaming).* Let ψ be a formula and let $\phi = \psi|_\pi$ be the subformula of ψ we want to rename. Let $x_1, \ldots, x_n$ be the free variables in ϕ and let R be a predicate symbol with arity n which is new to ψ. Then the formula

$$\psi[\pi / R(x_1, \ldots, x_n)] \wedge \mathit{def}(\pi, \psi, R)$$

is a *formula renaming* of ψ at position π. The formula $def(\pi, \psi, R)$ is a polarity dependent definition of the new predicate R:

$$def(\pi, \psi, R) = \begin{cases} \forall x_1, \dots, x_n\, [R(x_1, \dots, x_n) \supset \phi] & \text{if } pol(\psi, \pi) = 1 \\ \forall x_1, \dots, x_n\, [\phi \supset R(x_1, \dots, x_n)] & \text{if } pol(\psi, \pi) = -1 \\ \forall x_1, \dots, x_n\, [R(x_1, \dots, x_n) \equiv \phi] & \text{if } pol(\psi, \pi) = 0 \end{cases}$$

Formula renaming is a satisfiability preserving operation. This can be proved by choosing appropriate interpretations for the introduced predicates.

Now recall, that we only want to rename a formula if the number of clauses generated from the renamed formula is smaller than the number of clauses generated from the original formula. The first step towards this aim is to calculate the number of clauses generated by a standard CNF without any reduction. This is done by the function p defined in Table 3, where $\bar{p}(\psi)$ stands for $p(\neg\psi)$ for any formula ψ. The first column shows the form of ψ and the next two columns the corresponding recursive, top-down calculations for $p(\psi)$ and $\bar{p}(\psi)$, respectively.

ψ	$p(\psi)$	$\bar{p}(\psi)$
$\phi_1 \wedge \phi_2$	$p(\phi_1) + p(\phi_2)$	$\bar{p}(\phi_1)\bar{p}(\phi_2)$
$\phi_1 \vee \phi_2$	$p(\phi_1)p(\phi_2)$	$\bar{p}(\phi_1) + \bar{p}(\phi_2)$
$\phi_1 \supset \phi_2$	$\bar{p}(\phi_1)p(\phi_2)$	$p(\phi_1) + \bar{p}(\phi_2)$
$\phi_1 \equiv \phi_2$	$p(\phi_1)\bar{p}(\phi_2) + \bar{p}(\phi_1)p(\phi_2)$	$p(\phi_1)p(\phi_2) + \bar{p}(\phi_1)\bar{p}(\phi_2)$
$\forall x\, \phi_1$	$p(\phi_1)$	$\bar{p}(\phi_1)$
$\exists x\, \phi_1$	$p(\phi_1)$	$\bar{p}(\phi_1)$
$\neg\phi_1$	$\bar{p}(\phi_1)$	$p(\phi_1)$
$P(t_1, \dots, t_n)$	1	1

Table 3: Calculating the number of clauses

Note that the calculation for equivalences assumes a polarity dependent elimination of equivalences. Recall that this elimination avoids generating redundant clauses that are hardly recognizable once the CNF is built (see page 342).

Second, in order to check whether a renaming yields fewer clauses, we need to calculate the difference between the number of clauses generated with and without a renaming. So let us assume that we want to rename a subformula at position π within a formula ψ. The condition to be checked is

$$p(\psi) \geq p(\psi[\pi / R(x_1, \dots, x_n)]) + p(def(\pi, \psi, R))$$

The obvious problem with this condition is that the function p cannot be efficiently computed in general, for it grows exponentially in the size of the input formula. Moreover, a straightforward top-down implementation of p following Table 3 results in an algorithm with exponential time complexity. The exponential complexity can be avoided using a dynamic programming idea: we simply store intermediate results for subformulae. Nevertheless, because p grows exponentially, computing p requires arbitrary precision arithmetic. It turns out that this can hardly be afforded in practice. The rest of this section is therefore concerned with a solution to this problem, i.e., we shall show that it is not necessary to compute p at all.

Obviously, the formulae ψ and $\psi[\pi/R(x_1,\ldots,x_n)]$ differ only at position π, the other parts of the formulae remain identical. We make use of this fact by an abstraction of those parts of ψ that do not influence the changed position. To this end we introduce the notion of a coefficient as shown in Table 4. The coefficients

π	$\psi\|_\tau$	a^ψ_π	b^ψ_π
$\tau.i$	$\phi_1 \wedge \phi_2$	a^ψ_τ	$b^\psi_\tau \prod_{j\neq i} \bar{p}(\phi_j)$
$\tau.i$	$\phi_1 \vee \phi_2$	$a^\psi_\tau \prod_{j\neq i} p(\phi_j)$	b^ψ_τ
$\tau.1$	$\phi_1 \supset \phi_2$	b^ψ_τ	$a^\psi_\tau p(\phi_2)$
$\tau.2$	$\phi_1 \supset \phi_2$	$a^\psi_\tau \bar{p}(\phi_1)$	b^ψ_τ
$\tau.1$	$\phi_1 \equiv \phi_2$	$a^\psi_\tau \bar{p}(\phi_2) + b^\psi_\tau p(\phi_2)$	$a^\psi_\tau p(\phi_2) + b^\psi_\tau \bar{p}(\phi_2)$
$\tau.2$	$\phi_1 \equiv \phi_2$	$a^\psi_\tau \bar{p}(\phi_1) + b^\psi_\tau p(\phi_1)$	$a^\psi_\tau p(\phi_1) + b^\psi_\tau \bar{p}(\phi_1)$
$\tau.1$	$\neg\phi_1$	b^ψ_τ	a^ψ_τ
$\tau.1$	$\forall x\, \phi_1$	a^ψ_τ	b^ψ_τ
$\tau.1$	$\exists x\, \phi_1$	a^ψ_τ	b^ψ_τ
ϵ	ψ	1	0

Table 4: Calculating the coefficients

determine how often a particular subformula and its negation are duplicated in the course of a standard CNF translation. The coefficient a^ψ_π is the factor for the multiplication of $p(\psi|_\pi)$ in the CNF whereas the factor b^ψ_π is responsible for the multiplication of $\bar{p}(\psi|_\pi)$. The first column of Table 4 shows the form of π, the second column the form of ψ directly above position π (ψ itself if $\pi = \epsilon$). The next two columns demonstrate the corresponding recursive bottom-up calculations for a^ψ_π and b^ψ_π, respectively. Applied to our starting example formula $\psi = \phi_1 \vee \forall x\, \phi_2$ where we renamed position 2.1, i.e., the subformula ϕ_2, the coefficients are $a^\psi_{2.1} = p(\phi_1)$ (Table 4, eighth, second and last row, first column) and $b^\psi_{2.1} = 0$ (eighth, second and last row, second column). Note that a^ψ_π (b^ψ_π) is always 0 if $pol(\psi,\pi) = -1$ ($pol(\psi,\pi) = 1$).

Using the notion of a coefficient, the previously stated condition can be reformulated as

$$a^\psi_\pi p(\phi) + b^\psi_\pi \bar{p}(\phi) \geq a^\psi_\pi + b^\psi_\pi + p(def(\pi,\psi,R))$$

where we still assume that $\phi = \psi|_\pi$. Note that, since ϕ is replaced by an atom, the coefficients a^ψ_π, b^ψ_π are multiplied by 1 in the renamed version. Depending on the polarity of $\psi|_\pi$ the inequality is equivalent to one of the three inequalities:

$$\begin{aligned} a^\psi_\pi p(\phi) &\geq a^\psi_\pi + p(\phi) && \text{if } pol(\psi,\pi) = 1 \\ b^\psi_\pi \bar{p}(\phi) &\geq b^\psi_\pi + \bar{p}(\phi) && \text{if } pol(\psi,\pi) = -1 \\ a^\psi_\pi p(\phi) + b^\psi_\pi \bar{p}(\phi) &\geq a^\psi_\pi + b^\psi_\pi + p(\phi) + \bar{p}(\phi) && \text{if } pol(\psi,\pi) = 0 \end{aligned}$$

By simple arithmetical transformations, we can group all occurrences of factors a^ψ_π, b^ψ_π and all occurrences of $p(\phi)$ and $\bar{p}(\phi)$, respectively:

$$\begin{aligned} (a^\psi_\pi - 1)(p(\phi) - 1) &\geq 1 && \text{if } pol(\psi,\pi) = 1 \\ (b^\psi_\pi - 1)(\bar{p}(\phi) - 1) &\geq 1 && \text{if } pol(\psi,\pi) = -1 \\ (a^\psi_\pi - 1)(p(\phi) - 1) + (b^\psi_\pi - 1)(\bar{p}(\phi) - 1) &\geq 2 && \text{if } pol(\psi,\pi) = 0 \end{aligned}$$

Let us abbreviate the product $(a^\psi_\pi - 1)(p(\phi) - 1)$ with p_a and $(b^\psi_\pi - 1)(\bar{p}(\phi) - 1)$ with p_b. Since neither p_a nor p_b can become negative, in any of the cases where they appear, the first inequality holds if $p_a \geq 1$, the second inequality holds if $p_b \geq 1$ and the third inequality holds if (i) $p_a \geq 2$ or (ii) $p_b \geq 2$ or (iii) $p_a \geq 1$ and $p_b \geq 1$. In order to check these conditions, it suffices to test whether the coefficients a^ψ_π, b^ψ_π and the number of clauses $p(\phi)$, $\bar{p}(\phi)$ are strictly greater than 1, 2 or 3, respectively. This can always be checked in linear time with respect to the size of ψ. The condition $p(\phi) > 1$ holds iff there exists a position π such that $\phi[\phi_1 \equiv \phi_2]_\pi$ or $\phi[\phi_1 \wedge \phi_2]_\pi$ and $pol(\phi,\pi) = 1$ or $\phi[\phi_1 \circ \phi_2]_\pi$ with $pol(\phi,\pi) = -1$ and $\circ \in \{\vee, \supset\}$. The computations for the boolean conditions $p(\phi) > 2$ and $p(\phi) > 3$ are depicted in Table 5. We only considered the case of a conjunction and disjunction, since all other cases can be reduced to these cases. The computation of the conditions for $\bar{p}$ works accordingly, following Table 3.

ψ	$p(\psi) > 2$
$\phi_1 \wedge \phi_2$	$p(\phi_1) > 1$ or $p(\phi_2) > 1$
$\phi_1 \vee \phi_2$	$p(\phi_i) > 2$ or $p(\phi_1) > 1$ and $p(\phi_2) > 1$

ψ	$p(\psi) > 3$
$\phi_1 \wedge \phi_2$	$p(\phi_i) > 2$
$\phi_1 \vee \phi_2$	$p(\phi_i) > 3$ or $[p(\phi_i) > 2$ and $p(\phi_j) > 1, i \neq j]$

Table 5: The Boolean Conditions for p

As for the factors, Table 6 shows how to compute $a^\psi_\pi > 1$ and, following Table 4, this can be extended to the other cases for the a factor and the corresponding conditions for the b factor.

π	$\psi\|_\tau$	$a^\psi_\pi > 1$
$\tau.i$	$\phi_1 \wedge \phi_2$	$a^\psi_\tau > 1$
$\tau.i$	$\phi_1 \vee \phi_2$	$a^\psi_\tau > 1$ or $p(\phi_j) > 1$ for some j

Table 6: The Boolean Conditions for a

Hence we turned a test that required the computation of exponentially growing functions into a boolean condition that does not require any arithmetic calculation at all.

4.2. Theorem (Formula Renaming). *Formula Renaming preserves satisfiability and can be computed in polynomial time.*

In order to further reduce the number of eventually generated clauses it may still be useful to rename a formula, even if the above considerations do not apply. For example, renaming the formula $P_1 \vee (Q_1 \wedge Q_2)$ at position 2 results in three clauses, whereas a standard CNF translation of the original formula yields two clauses. This calculation also applies if this situation is repeated, as in

$$[P_1 \vee (Q_1 \wedge Q_2)] \wedge [P_2 \vee (Q_1 \wedge Q_2)] \wedge \ldots [P_n \vee (Q_1 \wedge Q_2)]$$

where our renaming criterion does not apply. But now a simultaneous renaming of all occurrences $(Q_1 \wedge Q_2)$ may pay off. It results in $n+2$ clauses whereas the standard CNF translation yields $2n$ clauses. Hence, it is useful to search for multiple occurrences of the same subformula. The problem here is to find an appropriate "equality" or "instance" relation between subformulae. In our example syntactic equality was sufficient to detect all such occurrences. In general, a matching process – probably with respect to the commutativity, associativity of some logical operators or even logical implication – may be needed to obtain a suitable renaming result. So we run here into a tradeoff between compact CNFs and computational complexity to achieve these CNFs. For example, if we slightly modify the above formula to

$$[P_1 \vee (Q_1(a_1) \wedge Q_2(a_1))] \wedge [P_2 \vee (Q_1(a_2) \wedge Q_2(a_2))] \wedge \ldots [P_n \vee (Q_1(a_n) \wedge Q_2(a_n)]$$

syntactic equality is not sufficient to simultaneously rename all occurrences of conjuncts $(Q_1(a_i) \wedge Q_2(a_i))$. Here we need a generalization. We add the definition $\forall x, y\, (R(x,y) \supset (Q_1(x) \wedge Q_2(y)))$ and replace the i^{th} occurrence of the conjunct by $R(x,y)\{x \mapsto a_i, y \mapsto a_i\}$ where $\{x \mapsto a_i, y \mapsto a_i\}$ is the matcher between $(Q_1(x) \wedge Q_2(y))$ and $(Q_1(a_i) \wedge Q_2(a_i))$.

Repeated application of formula renamings prevents the explosion in the number of clauses in the course of a CNF transformation. Furthermore, it is well-known that the introduction of new predicate symbols that abbreviate formulae, can cause an exponential decrease in proof length of an afterwards found resolution proof. On the other hand, the introduction of renamings may also increase proof (search) complexity. Consider a formula of the form $P \equiv Q \equiv P \equiv Q \ldots$ with n occurrences of P (and Q) where we assume that $\equiv$ binds to the left. The formula will be renamed for $n > 1$ where we introduce new propositional variables, linearly many in n, if we perform every renaming of a subformula that results in a smaller clause normal form with respect to our calculations. Hence, the number of truth assignments of the renamed formula grows exponentially in the size of the input formula. The problem is that the above formula is highly redundant since there are at most four different, non-redundant clauses in the sense of Section 3.7 with respect to two propositional variables. Redundancy is not considered by the calculations we developed in this section and hence the renaming method may produce worse results in this case. On the other hand, the standard CNF algorithm, we presented in Section 3 would generate exponentially many clauses after Section 3.6 which finally collapse to at most four clauses in Section 3.7. To prevent the intermediate exponential blowup, a much more sophisticated CNF procedure is needed. The point we want to make here is that formula renaming does not necessarily support proof search and that the design of a good CNF procedure is far from straightforward. Nevertheless, our experience shows that formula renaming, as we stated here, is a very useful mechanism (see Section 7 and Section 8).

Finally, it should be noted that after application of formula renamings and CNF generation, the "original" standard CNF can be reproduced by resolving on the literals that have been newly introduced in the renaming process. This may be an undesired behavior that can be inhibited by using ordered resolution with an appropriate ordering where literals built from the newly introduced predicate symbols are "small," see [Weidenbach 2001] (Chapter 27 of this Handbook).

5. Skolemization

One of the most important parts in the process of clause normal form transformation is *Skolemization*. It is used to get rid of existentially quantified variables and that by replacing each such occurrence with a Skolem function application. This, ultimately, leads to formulae in which all quantifications are universal (see Section 3).

Usually, there are two kinds of Skolemization techniques mentioned in the relevant literature on automated theorem proving which we call *Inner* and *Outer Skolemization* in this paper. The two differ mainly in the choice of the arguments for the Skolem functions. An existential quantifier to be replaced by some Skolem function lies within the scope of some universally quantified variables but also has some free variables within its own scope. Outer Skolemization takes the former variables as arguments for Skolem functions, whereas Inner Skolemization utilizes the latter variables. For both kinds of Skolemization it has been shown to be sometimes valuable to initially transform the given problem into antiprenex normal form.[1]

Here and in the sequel we assume uniqueness of quantified subformulae, i.e., no

[1] A formula is said to be in *antiprenex normal form* if its quantifiers are moved inwards as far as possible. There is one exception, however, existential quantifiers are moved outwards over disjunctions.

variable symbol is bound twice. Evidently, this can always easily be achieved by a suitable variable renaming.

5.1. DEFINITION *(Standard Skolemizations).* Let ϕ be a sentence in negation normal form and let $\exists x\, \psi = \phi|_\pi$ for some $\pi \in pos(\phi)$. Moreover, let $\{\overline{y}_n\} = \mathit{free}(\exists x\, \psi)$ and $\{\overline{z}_m\} = \{z \mid \forall z\, \xi = \phi|_\tau, \pi > \tau\}$. Then from ϕ we can obtain

- $\phi[\pi/\psi\{x \mapsto f(\overline{y}_n)\}]$ by an *inner Skolemization step* (see also 3.5), and
- $\phi[\pi/\psi\{x \mapsto f(\overline{z}_m)\}]$ by an *outer Skolemization step*,

where f is a new n-place (m-place respectively) function symbol.

Standard Skolemizing a formula means to apply standard Skolemization steps until all existential quantifiers are eliminated. Although the final result is in general not logically equivalent to the original formula, we know that preserves satisfiability.

Not much effort had been spent until recently to improve these standard Skolemization techniques. Nevertheless, even Skolemization leaves possibilities for further improvements. In this article we want to present two such proposals, the optimized Skolemization and the strong Skolemization. Both of these two new Skolemization methods are generalizations of Inner Skolemization in the sense that their Skolemization result subsumes the one of Inner Skolemization. This often allows for considerable simplifications after generating normal forms for automated theorem proving and may thus lead to a reduction of both search space and proof length.

5.2. DEFINITION *(Optimized Skolemization).* Let ϕ be a sentence in negation normal form, let $\exists \overline{x}_k\, (\psi_1 \wedge \psi_2) = \phi|_\pi$ with $\{\overline{y}_n\} = \mathit{free}(\phi|_\pi)$, and let $f_1, \ldots, f_k$ be new (Skolem) function symbols. If $\models \phi \supset \forall \overline{y}_n\, \exists \overline{x}_k\, \psi_1$ then

$$\forall \overline{y}_n\, \psi_1\{\ldots, x_i \mapsto f_i(\overline{y}_n), \ldots\} \wedge \phi[\pi/\psi_2\{\ldots, x_i \mapsto f_i(\overline{y}_n), \ldots\}]$$

can be obtained by an *optimized Skolemization step* from ϕ.

5.3. LEMMA. *Optimized Skolemization preserves satisfiability.*

PROOF. According to our definition of satisfiability preservation from page 339 we have to show that a formula is satisfiable if and only if the result of an optimized Skolemization step on this formula is satisfiable. Let us first consider the direction from right to left, i.e., assume that $\mathcal{M} \models \forall \overline{y}_n\, \psi_1\{\ldots, x_i \mapsto f_i(\overline{y}_n), \ldots\} \wedge \phi[\pi/\psi_2\{\ldots, x_i \mapsto f_i(\overline{y}_n), \ldots\}]$. Then $\mathcal{M} \models \forall \overline{y}_n\ (\psi_2\{\ldots, x_i \mapsto f_i(\overline{y}_n), \ldots\} \supset \exists \overline{x}_k(\psi_1 \wedge \psi_2))$ and so $\mathcal{M} \models \phi$. Hence, if ϕ is unsatisfiable then the result of an optimized skolemization step must be unsatisfiable as well for otherwise, as the above shows, we would be able to find a model for ϕ.

For the other direction of the proof assume that $\mathcal{M} \models \phi$. We have to show that there exists an interpretation $\mathcal{M}'$ that satisfies the result of an optimized skolemization step on ϕ. We do so by constructing such an interpretation; in fact we show that there exists a suitable $\mathcal{M}'$ that differs from $\mathcal{M}$ only in the interpretation of the new function symbols $f_1, \ldots, f_k$, i.e., $\mathcal{M}' = \langle \mathcal{D}, \mathcal{I}', \nu\rangle$ where $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \nu\rangle$ and $\mathcal{I}'$ extends $\mathcal{I}$ by suitable interpretations for $f_1, \ldots, f_k$. Now, consider an arbitrary sequence $\overline{a}_n$ of domain values. If $\mathcal{M}[\overline{y}_n/\overline{a}_n] \models \exists \overline{x}_k\,(\psi_1 \wedge \psi_2)$ then we choose the f_i such that $\mathcal{I}'(f_i)(\overline{a}_n) = b_i$, where the b_i are witnesses for the above existentially quantified variables x_i. If, on the other hand, $\mathcal{M}[\overline{y}_n/\overline{a}_n] \not\models \exists \overline{x}_k\,(\psi_1 \wedge \psi_2)$ then we choose the f_i such that $\mathcal{I}'(f_i)(\overline{a}_n) = c_i$ where $\mathcal{M}[\overline{y}_n/\overline{a}_n, \overline{x}_k/\overline{c}_k] \models \psi_1$. Note that according to our assumption that $\models \phi \supset \forall \overline{y}_n\, \exists \overline{x}_k\, \psi_1$ such $\overline{c}_k$ always exist. Now, the f_i have been constructed such that $\langle \mathcal{D}, \mathcal{I}', \nu\rangle \models \phi \wedge (\exists \overline{x}_k\,(\psi_1 \wedge \psi_2) \supset \psi_2\{x_i \mapsto f_i(\overline{y}_n)\})$ and so $\langle \mathcal{D}, \mathcal{I}', \nu\rangle \models \phi[\pi/\psi_2\{x_i \mapsto f_i(\overline{y}_n)\}]$. Moreover, by construction, $\langle \mathcal{D}, \mathcal{I}', \nu\rangle \models \forall \overline{y}_n \exists \overline{x}_k \psi_1 \wedge (\exists \overline{x}_k \psi_1 \supset \psi_1\{x_i \mapsto f_i(\overline{y}_n)\})$ and therefore $\langle \mathcal{D}, \mathcal{I}', \nu\rangle \models \forall \overline{y}_n \psi_1\{x_i \mapsto f_i(\overline{y}_n)\}$ and we are done. □

Note that optimized Skolemization in general requires a theorem prover on its own in order to prove the preliminary condition $\models \phi \supset \forall \overline{y}_n\, \exists \overline{x}_k\, \psi_1$.
As an example consider the subformula

$$\forall x, y, z\,(R(x, y) \wedge R(x, z) \supset \exists u\,(R(y, u) \wedge R(z, u)))$$

of some sentence ϕ and assume that $\forall x\, \exists y\, R(x, y)$ is provable from ϕ. With standard Inner Skolemization we would obtain the clauses

$$\begin{array}{l} \neg R(x, y) \vee \neg R(x, z) \vee R(y, f(y, z)) \\ \neg R(x, y) \vee \neg R(x, z) \vee R(z, f(y, z)) \end{array}$$

With optimized Skolemization, however, we would end up with the clause set

$$\begin{array}{l} R(y, f(y, z)) \\ \neg R(x, y) \vee \neg R(x, z) \vee R(z, f(y, z)) \end{array}$$

which is definitely superior to the standard result. Intuitively, the effect of optimized Skolemization compared to standard (Inner) Skolemization is that some of the literals in the standard result are deleted.

Strong Skolemization differs from optimized Skolemization in many respects. First of all, just like standard (Inner) Skolemization, it is a local method, i.e., it applies only to the subformula under consideration and does not take the whole problem into account. Moreover, it does not require a theorem prover on its own to perform its task. Now, before we define what strong Skolemization is about, let us first introduce the notion of a *free variable splitting.*

5.4. DEFINITION *(Free Variable Splitting).* Let ϕ be a sentence in negation normal form and let $\phi|_\pi = \exists \overline{x}_k\,(\psi_1 \wedge \ldots \wedge \psi_n)$ be a subformula of ϕ. Moreover, let

$$\begin{array}{rcl} \{\overline{z_{1l_1}}\} & = & \mathit{free}(\psi_1) \setminus \{\overline{x}_k\} \\ \{\overline{z_{il_i}}\} & = & (\mathit{free}(\psi_i) \setminus \bigcup_{1 \le j < i} \mathit{free}(\psi_j)) \setminus \{\overline{x}_k\} \text{ for } 1 < i \le n. \end{array}$$

We call $\langle \{\overline{z_{1l_1}}\}, \ldots, \{\overline{z_{nl_n}}\}\rangle$ the *free variable splitting* of $\phi|_\pi$.

Intuitively, each $\{\overline{z_{il_i}}\}$ contains the free variables of ψ_i (without the existentially quantified $\overline{x}_k$) that did not yet occur in ψ_j, $j < i$. Such a free variable splitting has some interesting features. For instance, its elements are pairwise disjoint and their overall union covers exactly the free variables of the formula $\exists \overline{x}_k\ (\psi_1 \wedge \ldots \wedge \psi_n)$. An important property which we will make use of in the sequel is the following: if $\langle\{\overline{z_1}_{l_1}\},\ldots,\{\overline{z_n}_{l_n}\}\rangle$ is a free variable splitting of $\exists\,\overline{x}_k\ (\psi_1 \wedge \ldots \wedge \psi_n)$[2] then $\langle\{\overline{z_1}_{l_1},\overline{z_2}_{l_2}\},\{\overline{z_3}_{l_3}\},\ldots,\{\overline{z_n}_{l_n}\}\rangle$ is a free variable splitting of $\exists\,\overline{v}_k\ (\psi_2 \wedge \ldots \wedge \psi_n)\{x_i \mapsto f_i(\overline{z_1}_{l_1},\overline{v}_k)\}$ provided the variables $\overline{v}_k$ are new to $\exists\,\overline{x}_k\ (\psi_1 \wedge \ldots \wedge \psi_n)$.

With this definition standard Inner Skolemization could be described as follows. Replace any occurrence of a subformula $\exists\,\overline{x}_k\ (\psi_1 \wedge \ldots \wedge \psi_n)$ with

$$\begin{array}{l}
\psi_1\{x_i \mapsto f_i(\overline{z_1}_{l_1},\ldots,\overline{z_n}_{l_n})\}\ \wedge\\
\psi_2\{x_i \mapsto f_i(\overline{z_1}_{l_1},\ldots,\overline{z_n}_{l_n})\}\ \wedge\\
\qquad\vdots\\
\psi_n\{x_i \mapsto f_i(\overline{z_1}_{l_1},\ldots,\overline{z_n}_{l_n})\}
\end{array}$$

where $\langle\{\overline{z_1}_{l_1}\},\ldots,\{\overline{z_n}_{l_n}\}\rangle$ is the free variable splitting of $\exists\,\overline{x}_k\ (\psi_1 \wedge \ldots \wedge \psi_n)$.

The effect of strong Skolemization lies in replacing some of these variable sequences with sequences of fresh universally quantified variables. The following definition specifies this.

5.5. DEFINITION *(Strong Skolemization).* Let ϕ be a sentence in negation normal form with $\phi|_\pi = \exists\overline{x}_k\ (\psi_1 \wedge \ldots \wedge \psi_n)$. A strong Skolemization step yields $\phi[\pi/\xi]$ with $\xi =$

$$\begin{array}{rl}
\forall\overline{w_2},\overline{w_3},\overline{w_4},\ldots,\overline{w_n} & \psi_1\{x_i \mapsto f_i(\overline{z_1},\overline{w_2},\overline{w_3},\overline{w_4},\ldots,\overline{w_n})\}\ \wedge\\
\forall\overline{w_3},\overline{w_4},\ldots,\overline{w_n} & \psi_2\{x_i \mapsto f_i(\overline{z_1},\overline{z_2},\overline{w_3},\overline{w_4},\ldots,\overline{w_n})\}\ \wedge\\
\vdots & \\
\forall\overline{w_n} & \psi_{n-1}\{x_i \mapsto f_i(\overline{z_1},\overline{z_2},\ldots,\overline{z_{n-1}},\overline{w_n})\}\ \wedge\\
 & \psi_n\{x_i \mapsto f_i(\overline{z_1},\overline{z_2},\ldots,\overline{z_{n-1}},\overline{z_n})\}
\end{array}$$

where $\langle\{\overline{z_1}\},\ldots,\{\overline{z_n}\}\rangle$ is the free variable splitting of $\exists\overline{x}_k\,(\psi_1 \wedge \ldots \wedge \psi_n)$. Each variable sequence $\overline{w_i}$ has length equal to the length of the variable sequence $\overline{z_i}$, and the $f_i, 1 \le i \le k$, are new (Skolem) function symbols.

In the proof that strong Skolemization behaves as desired we shall make use of some preliminary definitions as well as an important key lemma.

5.6. DEFINITION *(Selector Functions, Projections).* A *selector function* for a non-empty set $\mathcal{S} \subseteq \mathcal{D}^k$ is a total function $h : \mathcal{D}^k \to \mathcal{D}^k$ such that its co-domain is just $\mathcal{S}$, i.e., $cdom(h) = \mathcal{S}$.
As usual, the *projection functions* $proj_i$ allow us to refer to the i^{th} element of a given sequence, i.e., $proj_i(e_1,\ldots,e_i,\ldots,e_n) = e_i$.

5.7. LEMMA. *If* $\exists\,\overline{x}_k\ \ (\phi \wedge \psi)$ *is satisfiable then* $\forall\,\overline{w}_k\ \ \phi\{x_i \mapsto h_i(\overline{y}_n,\overline{w}_k)\} \wedge \exists\,\overline{v}_k\ \psi\{x_i \mapsto h_i(\overline{y}_n,\overline{v}_k)\}$ *is satisfiable as well, where* $\{\overline{y}_n\} = free(\phi) \setminus \{\overline{x}_k\}$.

[2] We assume that each of the ψ_i contains at least one of the variables from $\overline{x}_k$. Otherwise, such a ψ_i could be shifted out of the scope of the existential quantification (see also the simplification rules in Section 3.3).

PROOF. Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \nu \rangle$ be an interpretation and define

$$\mathcal{S}^\phi(\overline{b}_n) = \{\overline{a}_k \mid \mathcal{M}[\overline{y}_n/\overline{b}_n, \overline{x}_k/\overline{a}_k] \models \phi\}$$

Since $\mathcal{M} \models \exists\, \overline{x}_k\ (\phi \wedge \psi)$, we know that $\mathcal{S}^\phi(\nu(\overline{y}_n)) \neq \emptyset$, where $\nu(\overline{y}_n)$ abbreviates the sequence $\nu(y_1), \ldots, \nu(y_n)$. Thus we know that

$$\begin{aligned} \mathcal{M}[\overline{x}_k/\overline{a}_k] &\models \phi \quad \text{for all } \overline{a}_k \in \mathcal{S}^\phi(\nu(\overline{y}_n)) \text{ and} \\ \mathcal{M}[\overline{x}_k/\overline{a}_k] &\models \psi \quad \text{for some } \overline{a}_k \in \mathcal{S}^\phi(\nu(\overline{y}_n)). \end{aligned}$$

Now, for arbitrary $\overline{c}_n \in \mathcal{D}^n$, let $h^\phi(\overline{c}_n) \colon \mathcal{D}^k \to \mathcal{D}^k$ be a *selector function* for $\mathcal{S}^\phi(\overline{c}_n) \subseteq \mathcal{D}^k$. Then

$$\begin{aligned} \mathcal{M}[\overline{x}_k/h^\phi(\nu(\overline{y}_n))(\overline{a}_k)] &\models \phi \quad \text{for all } \overline{a}_k \in \mathcal{D}^k \text{ and} \\ \mathcal{M}[\overline{x}_k/h^\phi(\nu(\overline{y}_n))(\overline{a}_k)] &\models \psi \quad \text{for some } \overline{a}_k \in \mathcal{D}^k \end{aligned}$$

$h^\phi(\nu(\overline{y}_n))$ returns sequences of domain elements. Thus, in order to represent a substitute for each of the x_i, we have to refer to the respective projections on these sequences, i.e.,

$$\begin{aligned} \mathcal{M}[\ldots, x_i/\mathit{proj}_i(h^\phi(\nu(\overline{y}_n))(\overline{a}_k)), \ldots] &\models \phi \quad \text{for all } \overline{a}_k \in \mathcal{D}^k \text{ and} \\ \mathcal{M}[\ldots, x_i/\mathit{proj}_i(h^\phi(\nu(\overline{y}_n))(\overline{a}_k)), \ldots] &\models \psi \quad \text{for some } \overline{a}_k \in \mathcal{D}^k \end{aligned}$$

This immediately leads to a suitable interpretation for the new function symbols $h_1, \ldots, h_k$ which extend $\mathcal{M}$ to $\mathcal{M}'$ by

$$\mathcal{M}'(h_i)(\overline{b}_n, \overline{a}_k) = \mathit{proj}_i(h^\phi(\overline{b}_n)(\overline{a}_k)).$$

With this we get

$$\begin{aligned} \mathcal{M}'[\overline{w}_k/\overline{a}_k] &\models \phi\{x_i \mapsto h_i(\overline{y}_n, \overline{w}_k)\} \quad \text{for all } \overline{a}_k \in \mathcal{D}^k \text{ and} \\ \mathcal{M}'[\overline{v}_k/\overline{a}_k] &\models \psi\{x_i \mapsto h_i(\overline{y}_n, \overline{v}_k)\} \quad \text{for some } \overline{a}_k \in \mathcal{D}^k \end{aligned}$$

which finally can be simplified to

$$\mathcal{M}' \models \forall\, \overline{w}_k\ \phi\{x_i \mapsto h_i(\overline{y}_n, \overline{w}_k)\} \wedge \exists\, \overline{v}_k\ \psi\{x_i \mapsto h_i(\overline{y}_n, \overline{v}_k)\}$$

and this completes the proof. □

5.8. LEMMA. *Strong Skolemization preserves satisfiability.*

PROOF. We have to show that a formula ϕ is satisfiable if and only if the result of a strong Skolemization step on ϕ is satisfiable. The direction from right to left follows immediately from the fact that strong Skolemization generalizes standard Inner Skolemization. As for the other direction, assume that $\mathcal{M} \models \phi$ with $\phi|_\pi = \exists \overline{x}_k\, (\psi_1 \wedge \ldots \wedge \psi_n)$. Without loss of generality we assume that each of the

ϕ_i contains at least one of the variables from $\overline{x}_k$. We prove by induction on the structure of ϕ that there exists an interpretation $\mathcal{M}'$ that differs from $\mathcal{M}$ only in a suitable interpretation for the new Skolem function symbols $f_1, \ldots, f_k$ and that satisfies the strong Skolemization result on the indicated subformula.
The base case (where $\phi = \exists \overline{x}_k\ (\psi_1 \wedge \ldots \wedge \psi_n)$) is in fact the crucial part of the proof and requires another induction; this time on the number of conjuncts in the subformula under consideration (n in the formula above).
For the case $n = 1$ strong Skolemization does not differ from standard Inner Skolemization and we are done. For the induction step consider for $n > 1$ that $\mathcal{M} \models \exists \overline{x}_k\ (\psi_1 \wedge \ldots \wedge \psi_n)$. Then, by Lemma 5.7, we know that there exists an interpretation $\mathcal{M}'$ such that

$$\mathcal{M}' \models \forall \overline{w}_k\ \psi_1\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{w}_k)\} \wedge \exists \overline{v}_k\ (\psi_2 \wedge \ldots \wedge \psi_n)\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}$$

where $\mathcal{M}'$ is like $\mathcal{M}$ except for the additional interpretation for the new function symbols h_i. The second conjunct of this new formula is a sequence of existential quantifiers followed by a conjunction that consists of $n-1$ conjuncts. This means that we can apply the induction hypothesis on this subformula. Now, in order to apply the induction hypothesis on $\exists \overline{v}_k\ (\psi_2 \wedge \ldots \wedge \psi_n)\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}$ recall that the corresponding free variable splitting is given by $\langle \{\overline{z_1}_{l_1}, \overline{z_2}_{l_2}\}, \{\overline{z_3}_{l_3}\}, \ldots, \{\overline{z_n}_{l_n}\}\rangle$. Thus, an application of strong Skolemization leads to the satisfiability of

$$\begin{array}{rl}
\forall \overline{w}_k & \psi_1\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{w}_k)\} \wedge \\
\forall \overline{w_3}_{l_3}, \ldots, \overline{w_n}_{l_n} & \psi_2\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}\{v_j \mapsto g_j(\overline{z_1}_{l_1}, \overline{z_2}_{l_2}, \overline{w_3}_{l_3}, \ldots, \overline{w_n}_{l_n})\} \wedge \\
\forall \overline{w_4}_{l_4}, \ldots, \overline{w_n}_{l_n} & \psi_3\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}\{v_j \mapsto g_j(\overline{z_1}_{l_1}, \ldots, \overline{z_3}_{l_3}, \overline{w_4}_{l_4}, \ldots, \overline{w_n}_{l_n})\} \wedge \\
 & \vdots \\
\forall \overline{w_n}_{l_n} & \psi_{n-1}\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}\{v_j \mapsto g_j(\overline{z_1}_{l_1}, \ldots, \overline{z_{n-1}}_{l_{n-1}}, \overline{w_n}_{l_n})\} \wedge \\
 & \psi_n\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}\{v_j \mapsto g_j(\overline{z_1}_{l_1}, \overline{z_2}_{l_2}, \ldots, \overline{z_n}_{l_n})\}
\end{array}$$

Instantiating the variables w_i in the first conjunct with $g_i(\overline{z_1}_{l_1}, \overline{u_2}_{l_2}, \ldots, \overline{u_n}_{l_n})$ – where the elements in each $\overline{u_j}_{l_j}$ are fresh universally quantified variables – then reveals the satisfiability of

$$\begin{array}{rl}
\forall \overline{u_2}_{l_2}, \ldots, \overline{u_n}_{l_n} & \psi_1\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}\{v_j \mapsto g_j(\overline{z_1}_{l_1}, \overline{u_2}_{l_2}, \ldots, \overline{u_n}_{l_n})\} \wedge \\
\forall \overline{w_3}_{l_3}, \ldots, \overline{w_n}_{l_n} & \psi_2\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}\{v_j \mapsto g_j(\overline{z_1}_{l_1}, \overline{z_2}_{l_2}, \overline{w_3}_{l_3}, \ldots, \overline{w_n}_{l_n})\} \wedge \\
 & \vdots \\
\forall \overline{w_n}_{l_n} & \psi_{n-1}\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}\{v_j \mapsto g_j(\overline{z_1}_{l_1}, \ldots, \overline{z_{n-1}}_{l_{n-1}}, \overline{w_n}_{l_n})\} \wedge \\
 & \psi_n\{x_i \mapsto h_i(\overline{z_1}_{l_1}, \overline{v}_k)\}\{v_j \mapsto g_j(\overline{z_1}_{l_1}, \overline{z_2}_{l_2}, \ldots, \overline{z_n}_{l_n})\}
\end{array}$$

Now let $f_i(\overline{u_1}_{l_1}, \ldots, \overline{u_n}_{l_n}) = h_i(\overline{u_1}_{l_1}, g_1(\overline{u_1}_{l_1}, \ldots, \overline{u_n}_{l_n}), \ldots, g_k(\overline{u_1}_{l_1}, \ldots, \overline{u_n}_{l_n}))$ for arbitrary $\overline{u_1}_{l_1}, \ldots, \overline{u_n}_{l_n}$. This then leads to the satisfiability of

$$\begin{array}{rl} \forall\, \overline{w_2}_{l_2}, \overline{w_3}_{l_3}, \ldots, \overline{w_n}_{l_n} & \psi_1\{x_i \mapsto f_i(\overline{z_1}_{l_1}, \overline{w_2}_{l_2}, \ldots, \overline{w_n}_{l_n})\} \wedge \\ \forall\, \overline{w_3}_{l_3}, \ldots, \overline{w_n}_{l_n} & \psi_2\{x_i \mapsto f_i(\overline{z_1}_{l_1}, \overline{z_2}_{l_2}, \overline{w_3}_{l_3}, \ldots, \overline{w_n}_{l_n})\} \wedge \\ & \vdots \\ \forall\, \overline{w_n}_{l_n} & \psi_{n-1}\{x_i \mapsto f_i(\overline{z_1}_{l_1}, \ldots, \overline{z_{n-1}}_{l_{n-1}}, \overline{w_n}_{l_n})\} \wedge \\ & \psi_n\{x_i \mapsto f_i(\overline{z_1}_{l_1}, \overline{z_2}_{l_2}, \ldots, \overline{z_n}_{l_n})\} \end{array}$$

and we are done with the base case of the outer induction proof.
The induction steps then follow trivially from the induction hypothesis. □

Evidently, the strong Skolemization result subsumes what we could possibly get from standard Inner Skolemization (simply instantiate the $\overline{w_i}$ with the corresponding $\overline{z_i}$). Intuitively, the effect of strong Skolemization compared to standard Inner Skolemization is that some of the arguments of the new Skolem functions are replaced with new, universally quantified variables.

Let us again have a look at the example

$$\forall x, y, z\, (R(x, y) \wedge R(x, z) \supset \exists u\, (R(y, u) \wedge R(z, u))).$$

Strong Skolemization would lead us to the clause set

$$\begin{array}{l} \neg R(x, y) \vee \neg R(x, z) \vee R(y, f(y, w)) \\ \neg R(x, y) \vee \neg R(x, z) \vee R(z, f(y, z)) \end{array}$$

which is almost identical to the standard Skolemization outcome, with one exception however, the new variable w in the first clause. In fact, this tiny change allows us to perform a condensation step on the first two literals, so that we finally end up with

$$\begin{array}{l} \neg R(x, y) \vee R(y, f(y, w)) \\ \neg R(x, y) \vee \neg R(x, z) \vee R(z, f(y, z)) \end{array}$$

This result is not quite as strong as the one we obtained from optimized Skolemization. But notice that it did not require to prove the goal $\forall x\, \exists y\, R(x, y)$ which might actually be hard to prove, provided it is at all provable. In fact, proving such preliminary lemmata can be arbitrarily complicated and so, in case of optimized Skolemization, suitable restrictions are necessary to ensure that such intermediate steps ultimately terminate.

Now recall the example from above. It shows the superiority of both strong and optimized Skolemization over standard Skolemization. Also, it suggests to prefer optimized Skolemization provided the intermediate goal $\forall y, z\, \exists u\; R(y, u)$ is (more or less easily) provable. For this reason, it might be argued to first perform an optimized Skolemization step and only if this was not successful to proceed with a strong Skolemization step. Such a heuristics has shown to be sensible in practice, although there are examples where an opposite direction would be more effective. For instance, consider the formula $\forall x, y\; (R(x, y) \supset \exists z\; (P(z) \wedge Q(x, z) \wedge S(x, y, z)))$ and suppose that the relation R is provably non-empty. Standard Skolemization would then lead us to the clause set

$$\begin{array}{l}\neg R(x,y) \lor P(f(x,y)) \\ \neg R(x,y) \lor Q(x,f(x,y)) \\ \neg R(x,y) \lor S(x,y,f(x,y))\end{array}$$

whereas strong Skolemization would come up with

$$\begin{array}{l}\neg R(x,y) \lor P(f(v,w)) \\ \neg R(x,y) \lor Q(x,f(x,w)) \\ \neg R(x,y) \lor S(x,y,f(x,y)).\end{array}$$

Because of the assumption that R is provably non-empty, we eventually might be able to derive $P(f(v,w))$ from the first clause.

In case of optimized Skolemization the attempt to prove that $\exists z\, P(z)$ would obviously be successful and so we would finally end up with the clause set

$$\begin{array}{l}P(f(x,y)) \\ \neg R(x,y) \lor Q(x,f(x,y)) \\ \neg R(x,y) \lor S(x,y,f(x,y))\end{array}$$

which is a little less general than what we obtained from strong Skolemization (note the fresh variable w in the literal $Q(x,f(x,w))$ of the strong Skolemization result).

Thus there are examples which suggest to first try an optimized Skolemization step and then a strong Skolemization step (if the former was not successful in proving any of the intermediate goals), but also there are examples which propose the other way round. There is some evidence that a more tight combination of the two approaches is possible and valuable.

6. Simplification

Depending on the application area there usually exists a bunch of simplification techniques specific to that area. Often these techniques become available by particular semantic properties of the considered domain. For example, in a program verification context, syntactically different constructor terms generating a data type are usually interpreted by different domain elements. This implies that the domain of any model contains at least as many elements as there are constructor terms. This information can be exploited by simplification techniques (see Section 6.1) which falsify formulae that only hold in a domain of smaller size. As this example already suggests, advanced simplification techniques are usually so closely related to their specific domain that there cannot be a complete set of such techniques. Furthermore, such a presentation would not even be useful, since these techniques are often completely useless in other contexts. In this section we present three formula simplification techniques in the above sense. They should be considered as mere examples to motivate the investigation of such techniques in domains of interest.

6.1. Equality

The equality relation $\approx$ is an important relation in automated reasoning. It is interpreted as the identity on the domain $\mathcal{I}(\approx) = \{(a,a) \mid a \in \mathcal{D}\}$ and serves, for example, as a means to encode the behavior of functions. The equality relation can be directly, finitely axiomatized in first-order logic. However, it turns out that the generated axioms result in a proliferation of the search space of an afterwards applied automated reasoning system. Therefore, it is now commonly agreed that equality must be supported by specific reasoning facilities [Nieuwenhuis and Rubio 2001] (Chapter 7 of this volume). Here we want to show that already on the formula level, the equality relation can be employed for simplifications. The two simplifications we suggest should be considered as examples for such simplifications. There are many more of such simplifications possible. However, their usefulness highly depends on the respective application domain. We therefore concentrate on the two examples below.

$$\boxed{\begin{array}{lcll} \exists x\,[x \approx t \wedge \psi] & \to & \psi\{x \mapsto t\} & \text{if } x \notin \mathit{vars}(t) \\ \forall x\,[x \approx t \supset \psi] & \to & \psi\{x \mapsto t\} & \text{if } x \notin \mathit{vars}(t) \end{array}}$$

The rules terminate, since they reduce the number of equational atoms occurring in a formula. Furthermore, both rules are equivalence preserving.

6.1. LEMMA. *For any term t, variable x ($x \notin \mathit{vars}(t)$) and any formula ψ,*

$$\begin{array}{lcl} \mathcal{M} \models \exists x\,[x \approx t \wedge \psi] & \textit{iff} & \mathcal{M} \models \psi\{x \mapsto t\} \\ \mathcal{M} \models \forall x\,[x \approx t \supset \psi] & \textit{iff} & \mathcal{M} \models \psi\{x \mapsto t\} \end{array}$$

PROOF. We only show the first equivalence. The proof for the second is analogous. So, let us assume that $\mathcal{M} \models \exists x\,[x \approx t \wedge \psi]$. Then, by definition, there is some $a \in \mathcal{D}$ such that $\mathcal{M}[x/a] \models x \approx t$ and $\mathcal{M}[x/a] \models \psi$. The first part holds if we select an a with $a = \mathcal{M}(t)$. This can always be done, because $x \notin \mathit{vars}(t)$. Using $\mathcal{M}[x/a] \models \psi$ and an inductive argument on the structure of ψ, this implies $\mathcal{M} \models \psi\{x \mapsto t\}$. On the other hand, assume $\mathcal{M} \models \psi\{x \mapsto t\}$. Since $x \notin \mathit{vars}(t)$, the formula $\exists x\, x \approx t$ is valid witness the assignment $[x/\mathcal{M}(t)]$. We conclude $\mathcal{M} \models \exists x\,[x \approx t \wedge \psi]$ by an inductive argument on the structure of ψ. $\square$

Note that the above two rules together with the rules of Section 3.1 and the rule

$$\boxed{t \approx t \quad \to \quad \top}$$

are already strong enough to reduce all equational axioms mentioned above to $\top$. For example, a congruence axiom for some function f with arity two can be reduced as follows

$$\begin{array}{c} \forall x,y,z\,[x \approx y \supset f(x,z) \approx f(y,z)] \\ \downarrow \\ \forall x,y,z\,[f(y,z) \approx f(y,z)] \\ \downarrow \\ \top \end{array}$$

where we omitted some intermediate steps. The next set of simplification rules exploits properties of the domain of any interpretation satisfying the formula

$\psi[\forall y\ t \not\approx y]_\pi$	$\rightarrow$	$\psi[\pi/\bot]$	if $y \notin vars(t)$
$\psi[\forall y\ t \approx y]_\pi$	$\rightarrow$	$\psi[\pi/\bot]$	if $y \notin vars(t)$ and $\mathcal{M} \models \psi$ implies $\lvert\mathcal{D}\rvert > 1$
$\psi[\forall y\ t \approx y]_\pi$	$\rightarrow$	$\psi[\pi/\top]$	if $y \notin vars(t)$ and $\mathcal{M} \models \psi$ implies $\lvert\mathcal{D}\rvert = 1$
$\psi[\exists y\ t \approx y]_\pi$	$\rightarrow$	$\psi[\pi/\top]$	if $y \notin vars(t)$
$\psi[\exists y\ t \not\approx y]_\pi$	$\rightarrow$	$\psi[\pi/\top]$	if $y \notin vars(t)$ and $\mathcal{M} \models \psi$ implies $\lvert\mathcal{D}\rvert > 1$
$\psi[\exists y\ t \not\approx y]_\pi$	$\rightarrow$	$\psi[\pi/\bot]$	if $y \notin vars(t)$ and $\mathcal{M} \models \psi$ implies $\lvert\mathcal{D}\rvert = 1$

Since the rules introduce the logical constants $\bot$, $\top$ they should again be combined with the rules of Section 3.1. Moreover, explicit or implicit miniscoping (Section 3.3) may be necessary to achieve a suitable form for the above rules. The rules terminate, because the right hand side of each rule contains less symbols than its left hand side. The rules are equivalence preserving, e.g., for any domain $\mathcal{D}$ with $|\mathcal{D}| > 1$ the formula $\exists y\ t \not\approx y$ is obviously true and the formula $\forall y\ t \approx y$ is obviously false if $y \notin vars(t)$.

Four of the six transformation rules require cardinality properties of domains of models to be known. These are of course not decidable in general. For many applications, however, such properties are inherently given or can easily, even syntactically, be detected. For example, if some formula ψ contains as a conjunct the subformula $t \not\approx s$, then any model satisfying ψ has a domain with at least two elements.

6.2. *Atom Definitions*

An atom definition is a sentence of the form

$$\forall x_1, \ldots, x_m\, [P(t_1, \ldots, t_n) \equiv \psi]$$

where $free(P(t_1, \ldots, t_n)) = \{x_1, \ldots, x_m\}$. As we are only interested in sentences this implies $free(\psi) \subseteq free(P(t_1, \ldots, t_n))$. In case $P(t_1, \ldots, t_n)$ has the form $P(x_1, \ldots, x_n)$ and the predicate P does not occur in ψ, such a definition may be used to completely eliminate the predicate P and the definition from a formula.

$$\boxed{\begin{array}{l} \phi[P(s_1, \ldots, s_n)]_\pi \wedge \theta \quad \rightarrow \quad \phi[\pi/\psi\sigma] \wedge \theta \\ \qquad \text{if } \theta \text{ implies the atom definition } \forall x_1, \ldots, x_m\, [P(t_1, \ldots, t_n) \equiv \psi] \\ \qquad \text{and } P(t_1, \ldots, t_n)\sigma = P(s_1, \ldots, s_n) \end{array}}$$

The transformation is equivalence preserving, but in general exhaustive application of the reduction rule may not terminate, e.g., if ψ contains the predicate P. If ψ does not contain P and all t_i are different variables, then exhaustive application of the rule using a single atom definition terminates. The usefulness of an application of the rule highly depends on the context and the atom definition. There are application domains, e.g., mathematical theories or software verification where the expansion of atom definitions can be indispensable for the successful application of an automated reasoning system.

The notion of an atom definition can be generalized to sentences of the form

$$\forall x_1, \ldots, x_m \, [\phi \supset (P(t_1, \ldots, t_n) \equiv \psi)]$$

where we again require $\mathit{free}(P(t_1, \ldots, t_n)) = \{x_1, \ldots, x_m\}$ implying $\mathit{free}(\psi) \subseteq \mathit{free}(P(t_1, \ldots, t_n))$ and $\mathit{free}(\phi) \subseteq \mathit{free}(P(t_1, \ldots, t_n))$. For an application of this extended form, the formula ϕ must be valid in the context of the atom to be replaced. Of course, the applicability of such a definition is undecidable, in general. But there are domains where such definitions frequently occur and their applicability can be decided. For example, in a software verification domain, the formula ϕ may express sort/type restrictions that are usually decidable in that context.

As a transformation, the expansion of atom definitions and formula renaming (Section 4) seem to be inverse to each other. However, recall that we only apply formula renaming if this results in a smaller CNF and that the expansion of atom definitions makes no assumptions on the size. Its applicability should be decided on the basis of specific application domains or problems. This can lead to a combination where atom definitions are considered first and then the resulting formulae are considered for formula renamings, as it is done in SPASS, see [Weidenbach 2001] (Chapter 27 of this Handbook).

6.3. Binary Decision Diagrams

Another way to simplify formulae to obtain small clause normal forms is based on so-called *binary decision diagrams, or BDDs*, see [Clarke and Schlingloff 2001] (Chapter 24 of this Handbook). Such BDDs usually serve as rather compact representations of propositional formulae, they also gain more and more attention in the case of first-order logic, though.

The main idea behind BDDs is as follows. Given an arbitrary propositional formula $\phi[\psi]_\pi$, i.e., ψ is the sub-formula of ϕ at position π, we know that ϕ is equivalent to the conjunction of $\psi \supset \phi[\pi/\top]$ and $\neg\psi \supset \phi[\pi/\bot]$. This fact is often illustrated by a diagram of the following kind:

$$\begin{tikzcd} & \psi \arrow[dl, "+"'] \arrow[dr, "-"] & \\ \phi[\pi/\top] & & \phi[\pi/\bot] \end{tikzcd}$$

The +-arrow indicates the positive case, whereas the other one is responsible for the negative case. The above diagram is thus to be read as:

$$\texttt{if } \psi \texttt{ then } \phi[\pi/\top] \texttt{ else } \phi[\pi/\bot].$$

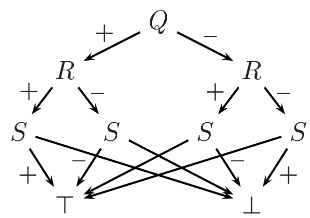
A formula is to be transformed such that each of its atomic and each of its quantified sub-formulae is split as indicated above. This way the bottom leaves will be just $\top$ or $\bot$. After that specific simplification rules as for example

$$\texttt{if } \chi \texttt{ then } \xi \texttt{ else } \xi \quad \to \quad \xi$$

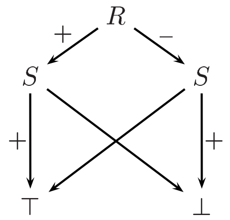
are exhaustively applied. Note that, given an ordering on the atoms and performing the above transformation with respect to this ordering, reduced BDDs form a unique normal form for propositional logic. In particular, each valid (unsatisfiable) formula gets thus transformed into the unique reduced and ordered BDD $\top$ ($\bot$ respectively). As a simple example consider the propositional formula

$$(Q \equiv R) \equiv (Q \equiv S)$$

A corresponding BDD could be



Now note that the two sub-graphs below the node labeled Q are identical, thus we may simplify the graph to



from which we can immediately read off the clause normal form as

$$(\neg R \vee S) \wedge (R \vee \neg S).$$

This approach can be generalized to the first-order level. Then it not only allows us to detect simplifications as the one described above but also helps to detect *false dependencies* of existentially quantified variables. In this case these approaches follows a similar aim as the special Skolemization methods described in Section 5.

7. Bibliographic Notes

Eliminating equivalences depending on their polarity was first suggested by Henschen, Lusk, Overbeek, Smith, Veroff, Winker and Wos [1980] in order to solve Peter Andrew's famous "Challenge Problem 1" [Andrews 1979].

The renaming idea goes back to Tseitin [1968] who showed that the introduction of new propositional symbols can result in exponentially shorter proofs. In the context of CNF translation important contributions have been made by Eder [1992], Plaisted and Greenbaum [1986] and Boy de la Tour [1992]. These approaches differ in their respective criteria to decide which subformulae are to be replaced by new predicates. Whereas Plaisted and Greenbaum basically suggested to rename all subformulae up to literals, Boy de la Tour only replaces a subformula if this would decrease the number of eventually generated clauses. Plaisted and Greenbaum [1986] also discuss the generalized formula renaming suggested at the end of Section 4. Since our goal is to few clauses, we followed the approach of Boy de la Tour [1992] where we presented his ideas using the notion of positions [Rock 1995]. Furthermore, we showed how to turn the approach into an efficient algorithm [Nonnengart, Rock and Weidenbach 1998]. This paper also contains an impressive experiment demonstrating the benefits of formula renaming and advanced Skolemization techniques. Boy de la Tour [1992] also proved that if a formula is checked top-down for possible renamings and does not contain equivalence symbols, then the renaming criterion in Section 4 produces an optimal[3] CNF among all renamings. In this case Boy de la Tour [1992] showed that we can even strengthen the greater equal test for a renaming to take place to a strictly greater test[4] and still obtain an optimal renaming. If an input formula contains equivalence symbols, the number of clauses of a CNF of a completely renamed formula grows linearly in the size of the input formula. As an upper bound for the number of symbols a CNF of a completely renamed formula ψ we obtain $|\psi|^3$ in case of the approach presented here and we obtain $|\psi|^2$ if the renaming criterion of Plaisted and Greenbaum [1986] is applied. Of course, all results abstract from potential effects that may be caused by redundant formulae (see page 352).

As for Skolemization issues, [Skolem 1920] is certainly worth a look at and that not only for historical reasons. More recent, though still classical references on this area can be found in [Andrews 1981, Loveland 1978, Chang and Lee 1973]. These cover essentially the standard approaches presented in this article. Refinements such as antiprenex transformation and miniscoping are described in [Egly 1994]. More information on the special Skolemization techniques introduced in this article, i.e., on strong as well as on optimized Skolemization, can be found in [Ohlbach and Weidenbach 1995, Nonnengart 1996, Nonnengart et al. 1998]. The reader who is rather interested in proof complexity issues is referred to the work of Baaz and Leitsch [1994] for an overview. Goubault [1995] introduces an approach of utilizing BDDs for the reduction of first-order formulae. This method is particularly interesting for it directly combines the reduction of BDDs with Skolemization. For the application of BDDs in first-order theorem proving Goubault and Posegga [1994] introduced a BDD-based approach and compared it with semantic tableaux.

Several techniques are not touched in this paper that seem to be valuable for further improvements. This includes, for example, potential reuse of Skolem functions and/or predicate symbols introduced by renaming, as described by Egly and Rath [1995].

8. Implementation Notes

Our considerations in Section 4 are motivated by the implementation of SPASS ([Weidenbach, Afshordel, Brahm, Cohrs, Engel, Keen, Theobalt and Topic 1999], see also [Weidenbach 2001] (Chapter 27 of this Handbook). Combining the renam-

[3] With respect to the number of finally generated clauses.

[4] See the disequations on page 348 and page 350 ff.

ing criteria developed in Section 4 with a top down traversal of the input formula provides the basis for an efficient implementation. As for data structures, it is important that the direct arguments of a formula as well as its direct super term can be efficiently accessed. A large number of experiments gave rise to the proposition that the renaming technique suggested here is in general superior to other renaming techniques and also to standard CNF translation [Nonnengart et al. 1998]. Concerning the renaming of multiple formula occurrences we mentioned on page 351, our experience shows that the following definition of matching is a good compromise between precision and complexity. We simple say that a formula ϕ is an instance of a formula ψ, if there exists a substitution σ with $\psi\sigma = \phi$.[5] This can be checked in linear time with respect to the size of ψ and ϕ and furthermore, it allows us to use the size of a formula as a prefilter for this condition. Hence, it is useful to store formulae together with their size.

Acknowledgments

We want to thank Thierry Boy de la Tour, Uwe Egly and Enno Keen for their valuable comments.

Bibliography

Andrews P. B. [1979], A challenge problem for automated theorem provers. The problem was posed at the fourth workshop on automated deduction, Austin.

Andrews P. B. [1981], 'Theorem proving via general matings', *Journal of the ACM* **28**(2), 193–214.

Baaz M., Egly U. and Leitsch A. [2001], Normal form transformations, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 5.

Baaz M. and Leitsch A. [1994], 'On skolemization and proof complexity', *Fundamenta Informaticae* **20**(4).

Bachmair L. and Ganzinger H. [2001], Resolution theorem proving, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 2, pp. 19–99.

Boy de la Tour T. [1992], 'An optimality result for clause form translation', *Journal of Symbolic Computation* **14**, 283–301.

Chang C.-L. and Lee R. C.-T. [1973], *Symbolic Logic and Mechanical Theorem Proving*, Computer Science and Applied Mathematics, Academic Press.

Clarke E. and Schlingloff H. [2001], Model checking, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 24.

Eder E. [1992], *Relative Complexities of First Order Calculi*, Artificial Intelligence, Vieweg.

Egly U. [1994], On the value of antiprenexing, *in* 'Logic Programming and Automated Reasoning, 5th International Conference, LPAR'94', Vol. 822 of *LNAI*, Springer, pp. 69–83.

Egly U. and Rath T. [1995], 'The halting problem: An automatically generated proof', *AAR Newsletter* **30**, 10–16.

Fermüller C., Leitsch A., Hustadt U. and Tammet T. [2001], Resolution decision procedures, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 25.

[5] Recall the application of substitutions to formulae, page 339.

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

GOUBAULT J. AND POSEGGA J. [1994], BDDs and automated deduction, *in* 'Proceedings of the 8th Int. Symp. on Methodologies for Intelligent Systems', Springer Verlag, LNAI.

HENSCHEN L., LUSK E., OVERBEEK R., SMITH B., VEROFF R., WINKER S. AND WOS L. [1980], 'Challenge problem 1', *SIGART Newsletter* **72**, 30–31.

LOVELAND D. W. [1978], *Automated Theorem Proving: A Logical Basis*, Vol. 6 of *Fundamental Studies in Computer Science*, North-Holland.

NIEUWENHUIS R. AND RUBIO A. [2001], Paramodulation-based theorem proving, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 7.

NONNENGART A. [1996], Strong skolemization, Research Report MPI-I-96-2-010, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany.

NONNENGART A., ROCK G. AND WEIDENBACH C. [1998], On generating small clause normal forms, *in* '15th International Conference on Automated Deduction, CADE-15', Vol. 1421 of *LNAI*, Springer, pp. 397–411.

OHLBACH H. J. AND WEIDENBACH C. [1995], 'A note on assumptions about skolem functions', *Journal of Automated Reasoning* **15**(2), 267–275.

PLAISTED D. A. AND GREENBAUM S. [1986], 'A structure-preserving clause form translation', *Journal of Symbolic Computation* **2**, 293–304.

ROCK G. [1995], Transformations of first-order formulae for automated reasoning, Diplomarbeit, Max-Planck-Institut für Informatik, Saarbrücken, Germany. Supervisors: H.J. Ohlbach, C. Weidenbach.

SKOLEM T. [1920], 'Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theoreme über dichte Mengen', *Skrifter utgit av Videnskappsellkapet i Kristiania* **4**, 4–36.

TSEITIN G. [1968], On the complexity of derivations in propositional calculus, *in* A. Slisenko, ed., 'Studies in Constructive Mathematics and Mathematical Logic'. Reprinted in [Tseitin 1983].

TSEITIN G. [1983], On the complexity of derivations in propositional calculus, *in* J. Siekmann and G. Wrightson, eds, 'Automation of Reasoning: Classical Papers on Computational Logic', Vol. 2, Springer, pp. 466–483.

WEIDENBACH C. [2001], Combining superposition, sorts and splitting, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 27.

WEIDENBACH C., AFSHORDEL B., BRAHM U., COHRS C., ENGEL T., KEEN E., THEOBALT C. AND TOPIC D. [1999], System description: Spass version 1.0.0, *in* H. Ganzinger, ed., '16th International Conference on Automated Deduction, CADE-16', Vol. 1632 of *LNAI*, Springer, pp. 314–318.

Index