Chapter 25

Resolution Decision Procedures

Christian G. Fermüller Alexander Leitsch

Ullrich Hustadt Tanel Tammet

Contents

HANDBOOK OF AUTOMATED REASONING
Edited by Alan Robinson and Andrei Voronkov

1. Introduction

The design of an algorithmic method which decides the truth of sentences is an old and important problem of mathematics and philosophy. It was Gottfried Wilhelm Leibniz who formulated already in the seventeenth century the fascinating vision of a *calculus ratiocinator* [Leibniz 1923], which reduces the solution of arbitrary problems to purely "mechanical" computation, once they have been formulated in an adequate formalism. About two centuries later Gottlob Frege [1884] developed the formalism of (what we now call) predicate logic, the first formal system with a strong potential of general knowledge representation. This allowed Leibniz's vision to materialize in the more moderate though also more concrete project to find a decision algorithm for the validity of formulas in predicate logic. Already Löwenheim [1915] presented results that (at least with hindsight) can be understood to define, on the one hand, a decision procedure for monadic first-order logic with equality, and on the other hand reduce the decision problem for full first-order logic to one with binary predicate symbols only. However, Hilbert should be credited for clear formulations (in his lectures around 1920 and most prominently in [Hilbert and Ackermann 1928]) of the *Entscheidungsproblem*, i.e. the problem to "decide the validity (respectively satisfiability) of a given logical expression by a finite number of operations". The importance of the decision problem is emphasized by the list of prominent scientists who contributed to it (by finding solvable subclasses): Löwenheim, Skolem, Ackermann, Bernays, Schönfinkel and Gödel *inter alii*. A. Church [1936] and—independently, in a more convincing form—A. Turing [1936/37] succeeded to prove the undecidability of the problem. Despite the unsolvability in its general form, the problem retained its vigor and importance. With the development of computers and automated theorem proving in the 1960s the implementation of a small but real *calculus ratiocinator* eventually became reality. In particular the invention of the unification principle and of resolution by J. A. Robinson [Robinson 1965*b*] paved the way for efficient ATP-programs. As theorem provers are essentially semi-decision procedures for first-order logic (i.e. proofs for all valid sentences can be effectively constructed) the challenge consists in guaranteeing *termination* of proof search procedures on decidable fragments of full first-order logic. The first prover of this type was designed by S. Y. Maslov [1968]; the underlying calculus combines sequent calculus and unification in a clever way. An interesting decision procedure for the so-called Gödel class was described in [Kallik 1969]. It is related to Robinson's resolution rule, but does not use unification. Somewhat later J. H. Joyner [1973] showed in his thesis that resolution can be turned into a decision procedure for many classes of clause sets that correspond to classical solvable classes.

On the basis of Joyner's approach we present resolution as decision procedure; that means we systematically investigate resolution refinements with respect to their potential as decision procedures. In Section 2 we introduce notations and basic definitions used throughout the paper. In Section 3 we investigate ordering refinements from the point of view of the decision problem. In some interesting cases we have to augment the ordering restrictions by saturation mechanisms in order to

guarantee termination of the proof search or use nonliftable orderings that are not complete in general. Section 4 is dedicated to hyperresolution and not only presents classes of clause sets decidable by this refinement, but also shows how to extend the resolution prover to an automated model building procedure. In Section 5, we give an application of resolution decision procedures to a problem in description logic. In particular we show that the satisfiability problem for general knowledge bases over the description logic $\mathcal{ALB}$, an extension of $\mathcal{ALC}$, can be reduced to that of formulas in various solvable fragments of first-order logic, which in turn can be solved by an appropriate resolution refinement. We conclude with some hints to related work in Section 6.

2. Notation and definitions

We provide definitions for the basic notions of clause logic and also introduce some more special terminology. Although we assume the reader to be familiar with the concept of resolution we review the fundamental definitions for sake of clarity. Additional terminology will be introduced in later sections whenever this assists precise formulations.

Concerning the language of clause logic we assume that there is an infinite supply of *variable symbols* V, *constant symbols* CS, *function symbols* FS, and *predicate symbols* PS. As usual each function and predicate symbol is associated with some fixed *arity* which we denote by $arity(F)$ for $F \in PS$ or FS. We call a predicate or function symbol *unary* if it is of arity 1; *binary* if the arity is 2; and, in general, *n-place* for arity n. If S is some set of expressions, clauses or clause sets then $CS(S)$, $FS(S)$, and $PS(S)$, refers to the set of constant, function and predicate symbols, respectively, that occur in S.

We define the notions *term*, *atom*, *literal*, *expression* and *clause* formally:

2.1. Definition. A *term* is defined inductively as follows:

1. Each variable and each constant is a term.
2. If $t_1, \ldots, t_n$ are terms and f is an n-place function symbol, then $f(t_1, \ldots, t_n)$ is also a term.

If a term t is of form $f(t_1, \ldots, t_n)$, $n \geq 1$, we call it *functional*; the set of arguments of t, $args(t)$, is $\{t_1, \ldots, t_n\}$;

2.2. Definition. If $t_1, \ldots, t_n$ are terms and P is an n-place predicate symbol then $A = P(t_1, \ldots, t_n)$ is an *atom*; $args(A)$ is the set $\{t_1, \ldots, t_n\}$.

2.3. Definition. A *literal* is either an atom or an atom preceded by a negation sign.

2.4. Definition. An *expression* is either a term or a literal.

2.5. Definition. A *clause* is a finite set of literals. The *empty clause* is denoted by $\Box$.

Throughout this chapter we shall speak of classes of clause sets, by this we always mean sets of finite sets of clauses.

2.6. DEFINITION. If a literal L is unnegated, i.e. if it is identical with an atom A, then the *dual* of L—L^d—equals $\neg A$. Otherwise, if L is of the form $\neg A$ then $L^d = A$. For a set of literals $C = \{L_1, \ldots, L_n\}$ we define $C^d = \{L_1^d, \ldots, L_n^d\}$.

Additionally we introduce the following notation:

2.7. DEFINITION. C_+ is the set of *positive literals* (unnegated atoms) of a clause C, analogously C_- denotes the set of *negative literals* (negated atoms) in C. A clause C is called *positive* if $C_+ = C$ and *negative* if $C_- = C$.

If S is a set of clauses then $P(S)$ denotes the subset of all positive clauses in S.

2.8. DEFINITION. C is a *Horn clause* if it contains at most one positive literal, i.e. $|C_+| \leq 1$.

2.9. DEFINITION. The *term depth* of a term t, $\tau(t)$, is defined as follows:

1. If t is a variable or a constant, then $\tau(t) = 0$.
2. If $t = f(t_1, \ldots, t_n)$, where f is an n-place function symbol, then $\tau(t) = 1 + \max\{\tau(t_i) \mid 1 \leq i \leq n\}$.

The term depth of a literal L is defined as $\tau(L) = \max\{\tau(t) \mid t \in \mathit{args}(L)\}$. The term depth of a clause C is defined as $\tau(C) = \max\{\tau(L) \mid L \in C\}$. For a set S of clauses we define $\tau(S) = \max\{\tau(C) \mid C \in S\}$.

The *set of all variables* occurring in E is denoted by $V(E)$; if C is a clause, then $V(C)$ is the union over all $V(P_i)$ for all atoms P_i in C.

2.10. DEFINITION. Expressions or clauses E_1 and E_2 are called *variable disjoint* if $V(E_1) \cap V(E_2) = \emptyset$.

An expression or a clause is called *ground* if no variables occur in it. We call it *constant free* if no constants occur in it, and *function free* if it does not contain function symbols.

2.11. DEFINITION. $\tau_{\min}(t, s)$ is defined as the *minimal depth of occurrence* of a term t within a term s; i.e.

1. If $t = s$ then $\tau_{\min}(t, s) = 0$.
2. Otherwise, if $s = f(s_1, \ldots, s_n)$, where f is an n-place function symbol, then $\tau(t, s) = 1 + \min\{\tau(t, s_i) \mid t \text{ occurs in } s_i\}$.

(If t does not occur in s then $\tau_{\min}(t, s)$ remains undefined.)

For a literal $L = {}^{(\neg)}P(t_1, \ldots, t_n)$, $\tau_{\min}(t, L) = \min\{\tau_{\min}(t, t_i) \mid t \text{ occurs in } t_i\}$. If C is a clause, then $\tau_{\min}(t, C)$ denotes the minimum of $\tau_{\min}(t, P_i)$ for all atoms P_i of C.

The *maximal depth of occurrence* $\tau_{\max}$ is defined analogously.

2.12. EXAMPLE. Let $P_1 = P(x, f(f(y)))$, $P_2 = Q(f(x))$ and $C = \{P_1, \neg P_2\}$. Then $\tau(P_1) = 2$, $\tau(P_2) = 1$, $\tau(C) = 2$, $\tau_{\min}(x, C) = 0$, $\tau_{\max}(x, C) = 1$, $\tau_{\min}(y, C) = \tau_{\max}(y, C) = 2$.

2.13. DEFINITION. The *maximal variable depth* of an expression E is defined as $\tau_v(E) = \max\{\tau_{\max}(x, E) \mid x \in V(E)\}$.
For clauses C we define $\tau_v(C) = \max\{\tau_v(L) \mid L \in C\}$;
analogously for clause sets S $\tau_v(S) = \max\{\tau_v(C) \mid C \in S\}$.

The *number of occurrences* of a variable x in a set of expressions is written as $OCC(x, E)$.

2.14. DEFINITION. Let V be the set of variables and T be the set of terms. A *substitution* is a mapping $\sigma : V \to T$ such that $\sigma(x) = x$ almost everywhere. We call the set $\{x \mid \sigma(x) \neq x\}$ *domain* of σ and denote it by $dom(\sigma)$, $\{\sigma(x) \mid x \in dom(\sigma)\}$ is called *range* of σ ($rg(\sigma)$). σ is called a *ground substitution* if $V(rg(\sigma)) = \emptyset$.

A *(variable) renaming* is an injective substitution σ such that $rg\,\sigma \subseteq V$.

We occasionally specify a substitution as a (finite) set of expressions of the form $x_i \leftarrow t_i$ with the intended meaning $\sigma(x_i) = t_i$. Compositions of substitutions are written in postfix notation, i.e. $\sigma\vartheta$ stands for $\vartheta \circ \sigma$; if E is a set of expressions then $E\sigma$ represents the set $\sigma(E) = \{\sigma(e) \mid e \in E\}$.

2.15. DEFINITION. We say that a substitution σ is *based on a clause set* S if no other constant and functions symbols besides that in $CS(S)$ and $FS(S)$, respectively, occur in the terms of $rg(\sigma)$.

2.16. DEFINITION. An expression or set of expressions E_1 is an *instance* of another expression (or set of expressions) E_2 if there exists a substitution σ such that $E_1 = E_2\sigma$. If $V(E_2\sigma) = \emptyset$ then $E_2\sigma$ is called a *ground instance*.

We may compare expressions, substitutions and clauses using the following ordering relation.

2.17. DEFINITION. Let E_1 and E_2 be expressions, then $E_1 \leq_s E_2$—read: E_1 *is more general than* E_2—if there exists a substitution σ such that $E_1\sigma = E_2$. For substitutions ρ and θ we define analogously: $\rho \leq_s \theta$ if there exists a substitution σ such that $\rho\sigma = \theta$. Similarly, if C and D are clauses, then $C \leq_{ss} D$ if there exists a substitution σ such that $C\sigma \subseteq D$. In this case we also say, in accordance with the usual resolution terminology, that C *subsumes* D.

2.18. DEFINITION. A set of expressions M is *unifiable* by a substitution σ (called *unifier* of M) if $E_i\sigma = E_j\sigma$ for all $E_i, E_j \in M$. σ is called *most general unifier (mgu)* of M if for every other unifier ρ of M: $\sigma \leq_s \rho$.

We shall also say that E_1 *is unifiable with* E_2 if $\{E_1, E_2\}$ is unifiable.

Remember that any two different mgus of a set of expressions only differ in the names of the variables.

2.19. DEFINITION. A *factor* of a clause C is a clause $C\theta$, where θ is an mgu of some $C' \subseteq C$.

2.20. DEFINITION. A clause C is called *condensed* if there exists no factor of C which is a proper subclause of C. If C' is a condensed factor of C such that $C' \subseteq C$ then C' is called a *condensation* of C; condensations are unique up to renaming [Joyner 1976].

By the *condensation rule* we mean the principle that—during proof-search—every clause is immediately replaced by its condensation.

2.21. EXAMPLE. $\{P(x,y), P(y,x)\}$ is condensed. $\{P(x,y), P(x,a)\}$ is not condensed; its condensation is $\{P(x,a)\}$.

For the *resolvent* we retain the original definition of Robinson [1965*b*], which combines factorization and (binary) resolution.

2.22. DEFINITION. If C and D are variable disjoint clauses and M and N are subsets of C and D respectively, such that $N^d \cup M$ is unifiable by the mgu θ, then $E = (C - M)\theta \cup (D - N)\theta$ is a *(Robinson-)resolvent* of C and D.

If M and N are singleton sets then E is called *binary resolvent* of C and D.

The atom A of $(N^d \cup M)\theta$ is called the *resolved atom*. We also say that E is *generated via* A. The elements of N and M are called the *literals resolved upon*.

2.23. DEFINITION. For a clause set S we define $Res(S)$ as the set of Robinson-resolvents of S. Additionally we define:

$$R(S) = S \cup Res(S)$$

$$R^0(S) = S, \quad R^{i+1}(S) = R(R^i(S)), \qquad \text{and}$$

$$R^*(S) = \bigcup_{i \geq 0} R^i(S).$$

We say that a clause C is *derivable from* a clause set S if $C \in R^*(S)$.

In the following sections we use various *refinements* of Robinson's resolution procedure. By a refinement (call it x) of resolution we mean a computable operator ρ_x such that

$$\rho_x(S) \subseteq R^*(S)$$

for all clause sets S. We define

$$R_x(S) = S \cup \rho_x(S).$$

R_x^i and the deductive closure R_x^* are defined in the obvious way. Particularly we write $R_<$ for operators based on an A-ordering $<$ and R_H for hyperresolution.

In contrast to resolution refinements we shall also define *variants of resolution*: For resolution variants we allow ordinary resolvents to be replaced by certain instances of it. This technique is also called *saturation*.

In Section 3, we sometimes refer to the well known splitting rule. To make the argumentation more precise we present a formal definition.

2.24. Definition. Let L and L' be literals in a clause C; we define $L \sim_v L'$ if $V(L) \cap V(L') \neq \emptyset$ and $\sim_{v*}$ as the transitive closure of $\sim_v$. An equivalence class under $\sim_{v*}$ is called *component* of C. A clause is called *decomposed* if it consists of only one component and *disconnected* if every component consists of one literal only.

2.25. Example. Let $C_1 = \{P(x), R(x,y), Q(y)\}$, $C_2 = \{P(x), Q(y)\}$ and $C_3 = \{P(x), R(x,y), Q(z)\}$. Then C_1 is decomposed (note that $P(x) \sim_v R(x,y)$, $R(x,y) \sim_v Q(y)$ and thus $P(x) \sim_{v*} Q(y)$). C_2 is disconnected and C_3 is neither decomposed nor disconnected.

2.26. Definition. For any clause set S let $SPLIT(S)$ denote the set of clause sets obtained by *splitting* all members of S as far as possible. More accurately we define recursively:

1. $\Sigma_0 = \{S\}$.
2. If for all $S' \in \Sigma_i$ and all $C \in S'$ C is decomposed then
$$\Sigma_{i+1} = \Sigma_i.$$
3. If there is some $S' \in \Sigma_i$ such that for some $C \in S'$ C is can be decomposed into C' and C'' then
$$\Sigma_{i+1} = (\Sigma_i - \{S'\}) \cup ((S' - \{C\}) \cup \{C'\}) \cup ((S' - \{C\}) \cup \{C''\}).$$
(A proper C is chosen nondeterministically)

Now we may define $SPLIT(S) = \Sigma_k$ where $k = \min\{l \mid \Sigma_l = \Sigma_{l+1}\}$.

The *splitting rule* says that we have to apply resolution to all members of $SPLIT(S)$ separately to test for the unsatisfiability of S. (Recall that S is unsatisfiable iff S' is unsatisfiable for all $S' \in SPLIT(S)$.)

For the semantic of clause logic we refer, as usual, to the terminological machinery inspired by Jaques Herbrand. We review the basic definitions:

2.27. Definition. The *Herbrand universe* $\mathbf{H}_S$ of a clause set S is the set of all ground terms built up from $CS(S)$ and $FS(S)$. (If $CS(S)$ is empty we introduce a special constant symbol to prevent $\mathbf{H}_S$ from being empty).

2.28. Definition. An *Herbrand instance* of an atom or a clause C in S is a ground instance $C\theta$ of C such that θ is based on S.

2.29. DEFINITION. The *Herbrand base* $\hat{\mathbf{H}}_S$ is the set of all Herbrand instances of atoms appearing in clauses of S.

2.30. DEFINITION. An *Herbrand interpretation* $\mathbf{HI}_S$ for a clause set S is a subset of $\hat{\mathbf{H}}_S$ with the intended meaning that the truth value **true** is assigned to all elements of $\mathbf{HI}_S$ and the truth value **false** is assigned to all atoms in $\hat{\mathbf{H}}_S - \mathbf{HI}_S$.

We shall make use of the following naming conventions: For variable symbols we use letters from the end of the alphabet (u, v, w, x, y, z); for constant symbols, letters a, b, c are used; function symbols are denoted by f, g or h; as metavariables for terms we use t or s; capital letters will denote atoms, literals, clauses or certain sets of expressions. Whenever needed these letters are augmented by indices.

2.31. DEFINITION. The set of *first-order formulas* over a signature (CS, FS, PS) and a set of variables V is inductively defined as follows: (i) every atom is a first-order formula, (ii) if φ and ψ are formulas and x is a variable, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \to \psi$, $\varphi \leftrightarrow \psi$, $(\forall x)\,\varphi$, and $(\exists x)\,\varphi$ are first-order formulas.

The notions of scope of a universal quantifier $\forall$ and an existential $\exists$, and free and bound variables are defined in the usual way. We use $(Qx)\varphi$ to denote either $(\forall x)\,\varphi$ or $(\exists x)\,\varphi$. The *matrix* of a formula φ is the formula ψ obtained from φ by deleting all occurrences of quantifiers. By $\varphi[x/t]$ we denote the formula obtained from φ by replacing every free occurrence of the variable x by the term t.

2.32. DEFINITION. A formula is *rectified* if every quantifier binds a different variable and free variables are different from bound ones. It is easy to see that every first-order formula is logically equivalent to a rectified one.

A formula φ is in *prenex form* if $\varphi = (Q_1x_1)\ldots(Q_nx_n)\psi$ where $Q_1, \ldots, Q_n$ are quantifiers and the formula ψ is quantifier free. The sequence of quantifiers $(Q_1x_1)\ldots(Q_nx_n)$ is the *prefix* of φ. A formula in prenex form is also called *prenex formula*. For every formula φ there exists an equivalent formula *prenex*(φ) in prenex form.

A formula φ is in *negation normal form* if for every subformula $\neg\psi$ of φ, ψ is an atomic formula and φ contains no occurrences of the binary operators $\to$ and $\leftrightarrow$. For every formula φ there exists an equivalent formula *nnf*(φ) in negation normal form.

Let φ be a rectified formula in negation normal form. The *structural Skolem form* *sSkf*(φ) of φ is obtained by exhaustively applying the following rule to φ:

$$(\exists x)\,\psi \Rightarrow \psi[x/f(y_1, \ldots, y_n)]$$

where $y_1, \ldots, y_n$ are the universally quantified variables in the scope of which $(\exists x)\,\psi$ occurs and f is a new function symbol.

2.33. EXAMPLE. The structural Skolem form of $(\forall x)\,(\exists x_2)\,(\forall z_1)\,(((\neg p(x) \vee q(x_2)) \wedge (\neg q(x_2) \vee p(x))) \wedge \neg r(x, z_1))$ is $(\forall x)\,(\forall z_1)\,(((\neg p(x) \vee q(f(x))) \wedge (\neg q(f(x)) \vee p(x))) \wedge \neg r(x, z_1))$.

The structural Skolem form is only one of the possible Skolem normal forms of first-order formula. For an in-depth discussion of various different skolemization techniques see [Baaz et al. 2001] (Chapter 5 of this Handbook).

2.34. Definition. Let φ be a rectified first-order formula. The *standard clausal form* of φ is given by $cls(cnf(sSkf(prenex(nnf(\varphi)))))$ where *cnf* denotes a syntax preserving transformation of first-order formulas to conjunctive normal form based on the rule of distributivity, and *cls* is defined by

$$cls(\psi) = \{\{L_1, \ldots, L_n\} \mid L_1 \vee \ldots \vee L_n \text{ is a conjunct in } \psi\}.$$

2.35. Example. The standard clausal form of $(\forall x)\,(\forall z_1)\,(((\neg p(x) \vee q(f(x))) \wedge (\neg q(f(x)) \vee p(x))) \wedge \neg r(x, z_1))$ is $\{\{\neg p(x), q(f(x))\}, \{\neg q(f(x)), p(x)\}, \{\neg r(x, z_1)\}\}$.

Note that the size of $cnf(\psi)$ and therefore the size of the standard clausal form of ψ can be exponential in the size of ψ, due to the duplication of subformulas in applications of the rule of distributivity. One solution is the use of *structural transformations* [Baaz, Fermüller and Leitsch 1994, Plaisted and Greenbaum 1986] in the process of transforming formulas into conjunctive normal form. The use of structural transformations in the computation of the clausal form of a first-order formula is discussed in detail in [Baaz et al. 2001, Nonnengart and Weidenbach 2001] (Chapters 5 and 6 of this Handbook).

In the following we describe a particular instance of structural transformations, namely, the computation of the *definitional form* of a first-order formula.

2.36. Definition. Let $\mathrm{Pos}(\varphi)$ be the set of positions of a first-order formula φ. If λ is a position in φ, then $\varphi|_\lambda$ denotes the subformula of φ at position λ and $\varphi[\lambda \leftarrow \psi]$ is the result of replacing the subformula of φ at position λ by ψ.

We associate with each element λ of $\Lambda \subseteq \mathrm{Pos}(\varphi)$ a new predicate symbol Q_λ and a new literal $Q_\lambda(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are the free variables of $\varphi|_\lambda$. Let

$$\begin{aligned}
\mathrm{Def}^+_\lambda(\varphi) &= (\forall x_1) \ldots (\forall x_n)\ \ (Q_\lambda(x_1, \ldots, x_n) \rightarrow \varphi|_\lambda) \quad \text{and} \\
\mathrm{Def}^-_\lambda(\varphi) &= (\forall x_1) \ldots (\forall x_n)\ \ (\varphi|_\lambda \rightarrow Q_\lambda(x_1, \ldots, x_n)).
\end{aligned}$$

The *definition* of Q_λ is the formula

$$\mathrm{Def}_\lambda(\varphi) = \begin{cases} \mathrm{Def}^+_\lambda(\varphi) & \text{if } \varphi|_\lambda \text{ has positive polarity,} \\ \mathrm{Def}^-_\lambda(\varphi) & \text{if } \varphi|_\lambda \text{ has negative polarity,} \\ \mathrm{Def}^+_\lambda(\varphi) \wedge \mathrm{Def}^-_\lambda(\varphi) & \text{otherwise.} \end{cases}$$

Now, define $\mathrm{Def}_\Lambda(\varphi)$ inductively by:

$$\begin{aligned}
\mathrm{Def}_\emptyset(\varphi) &= \varphi \quad \text{and} \\
\mathrm{Def}_{\Lambda \cup \{\lambda\}}(\varphi) &= \mathrm{Def}_\lambda(\varphi) \wedge \mathrm{Def}_\Lambda(\varphi[\lambda \leftarrow Q_\lambda(x_1, \ldots, x_n)])),
\end{aligned}$$

where λ is maximal in $\Lambda \cup \{\lambda\}$ with respect to the prefix ordering on positions. The corresponding clauses will be called *definitional clauses*. A *definitional form* of φ is $\mathrm{Def}_\Lambda(\varphi)$, where Λ is a subset of positions of subformulas (usually, nonatomic or nonliteral subformulas).

If a formula φ contains no occurrences of equivalence, then for any position λ, $\varphi|_\lambda$ will have either positive or negative polarity. In this case the third case in the definition of $\mathrm{Def}_\lambda(\varphi)$ is obsolete.

2.37. EXAMPLE. Consider the first-order formula

$$\varphi = (\exists x)\,(\forall y)\,(\neg R(x,y) \vee (\neg q(y) \vee (\exists z)\,(R(y,z) \wedge q(z))))$$

Let λ_1, λ_2, and λ_3 denote the positions of the subformulas $(\exists z)\,(R(y,z) \wedge q(z))$, $(\neg q(y) \vee (\exists z)\,(R(y,z) \wedge q(z)))$, and $(\forall y)\,(\neg R(x,y) \vee (\neg q(y) \vee (\exists z)\,(R(y,z) \wedge q(z))))$ in φ, respectively. Then

$$\begin{aligned}
\mathrm{Def}_{\{\lambda_1,\lambda_2,\lambda_3\}}(\varphi) &= \mathrm{Def}_{\lambda_1}(\varphi) \wedge \mathrm{Def}_{\{\lambda_2,\lambda_3\}}(\varphi[\lambda_1 \leftarrow Q_{\lambda_1}(y)]) \\
&= (\forall x)\,(Q_{\lambda_1}(x) \to (\exists z)\,(R(y,z) \wedge q(z))) \wedge \\
&\quad \mathrm{Def}_{\{\lambda_2,\lambda_3\}}((\exists x)\,(\forall y)\,(\neg R(x,y) \vee (\neg q(y) \vee Q_{\lambda_1}(y)))) \\
&= (\forall y)\,(Q_{\lambda_1}(y) \to (\exists z)\,(R(y,z) \wedge q(z))) \wedge \\
&\quad \mathrm{Def}_{\lambda_2}((\exists x)\,(\forall y)\,(\neg R(x,y) \vee (\neg q(y) \vee Q_{\lambda_1}(y)))) \wedge \\
&\quad \mathrm{Def}_{\{\lambda_3\}}((\exists x)\,(\forall y)\,(\neg R(x,y) \vee Q_{\lambda_2}(y))) \\
&= (\forall y)\,(Q_{\lambda_1}(y) \to (\exists z)\,(R(y,z) \wedge q(z))) \wedge \\
&\quad (\forall y)\,(Q_{\lambda_2}(y) \to (\neg q(y) \vee Q_{\lambda_1}(y))) \wedge \\
&\quad \mathrm{Def}_{\{\lambda_3\}}((\exists x)\,(\forall y)\,(\neg R(x,y) \vee Q_{\lambda_2}(y))) \\
&= (\forall y)\,(Q_{\lambda_1}(y) \to (\exists z)\,(R(y,z) \wedge q(z))) \wedge \\
&\quad (\forall y)\,(Q_{\lambda_2}(y) \to (\neg q(y) \vee Q_{\lambda_1}(y))) \wedge \\
&\quad \mathrm{Def}_{\lambda_3}((\exists x)\,(\forall y)\,\neg R(x,y) \vee Q_{\lambda_2}(y)) \wedge \\
&\quad \mathrm{Def}_{\emptyset}((\exists x)\,Q_{\lambda_3}(x)) \\
&= (\forall x)\,(Q_{\lambda_1}(y) \to (\exists z)\,(R(y,z) \wedge q(z))) \wedge \\
&\quad (\forall x)\,(Q_{\lambda_2}(y) \to (\neg q(y) \vee Q_{\lambda_1}(y))) \wedge \\
&\quad (\forall x)\,(Q_{\lambda_3}(x) \to (\forall y)\,(\neg R(x,y) \vee Q_{\lambda_2}(y))) \wedge \\
&\quad (\exists x)\,Q_{\lambda_3}(x).
\end{aligned}$$

2.38. PROPOSITION. *Let φ be a first-order formula and let Λ be a subset of* $\mathrm{Pos}(\varphi)$.

1. *φ is satisfiable iff* $\mathrm{Def}_\Lambda(\varphi)$ *is satisfiable.*
2. $\mathrm{Def}_\Lambda(\varphi)$ *can be computed in polynomial time.*
3. *If Λ contains all positions of non-literal subformulas of φ, then the size of the standard clausal form of* $\mathrm{Def}_\Lambda(\varphi)$ *is linear in the size of φ.*

The p-definitional form of a closed first-order formula defined in [Baaz et al. 2001, page 310] (Chapter 5 of this Handbook) Chapter 5 is a special case of Definition 2.36. The length-optimizing form of [Boy de la Tour 1992] is based on a similar

principle, as well as the formula renaming technique described in [Nonnengart and Weidenbach 2001] (Chapter 6 of this Handbook).

3. Decision procedures based on ordering refinements

3.1. Motivation

Very different types of resolution refinements can be turned into effective means for deciding satisfiability of certain classes of clause sets. In Section 4 hyperresolution will be investigated from this point of view. However, the ground breaking work of William H. Joyner [Joyner 1973, Joyner 1976] and Russian scholars—in particular S. J. Maslov (see [Maslov 1968] and [Zamov 1989])—that introduced the paradigm of "resolution as a decision procedure" (or, equivalently, "inverse method as a decision procedure") was based on ordering refinements, sometimes augmented by saturation mechanisms. In this section we review some generalizations of these results.

Traditionally, investigations into the decision problem concentrated on prefix classes of first-order formulas without function symbols. It turns out that certain (completeness preserving) ordering strategies allow to control effectively the nesting of terms in resolvents of clauses corresponding (via skolemization) to formulas with certain types of quantifiers prefixes. In particular the initially extended Ackermann class (prefix $\exists^*\forall\exists^*$), the initially extended Gödel class (prefix $\exists^*\forall\forall\exists^*$), the Skolem class (see Subsection 3.5) and the related class $\mathcal{K}$ of Maslov can be shown decidable by such methods. Once the role of ordering restrictions is understood it is easy to generalize these classes to classes of clause sets that allow for more complex term structures than those obtained by skolemization of function free formulas.

As a motivating example consider the *monadic class*, i.e. the class of all first-order formulas (without function symbols) that contain only monadic predicate symbols. Obviously, the atoms of clauses corresponding to such formulas are all of form $P(x)$ or $P(f(x_1,\ldots,x_n))$, where f denotes a skolem function. To define a proof search procedure which generates only finitely many different resolvents from such sets of clauses we have to prevent the derivation of "deep" resolvents. Observe that unrestricted resolution allows to derive all clauses of form

$$\{P(f^n(x))\} \quad \text{where } n \geq 1$$

from the clause set

$$S = \big\{\{\neg P(x), P(f(x))\},\ \{P(x)\}\big\}$$

corresponding to the monadic formula $\forall x\exists y([P(x) \supset P(y)] \wedge P(x))$. However, suppose we order atoms (and correspondingly literals) by defining

$$P(f(x_1,\ldots,x_n))\sigma \succ Q(x_i)\sigma \quad (1 \leq i \leq n), \text{ and}$$

$$P(f(x_1,\ldots,x_n))\sigma \succ Q(g(x_1,\ldots,x_m))\sigma \quad \text{if } n > m$$

for all monadic predicate symbols P, Q, function symbols f, g, and substitutions σ. If we allow to resolve only upon maximal literals in clauses then no clause at all is derivable from S. It is easy to see that no clauses with nested function symbols are derivable from any clause set corresponding to a monadic formula if we employ the mentioned resolution refinement, which is well known to preserve completeness (see below). Still, clauses of arbitrary length could be derivable. But this can be prevented either by decomposing clauses into their variable disjoint subclauses or by applying the condensation rule mentioned in Section 2, above. Summarizing we get that $R_x^*(S)$ is finite for all clause sets S corresponding to a formula in the monadic class, where R_x is a properly defined resolution operator. By the completeness of R_x, we have just described a decision procedure for the monadic class since we only have to check whether $\Box \in R_x^*(S)$.

Traditionally, decidability results like that for monadic class have been solved by quite different methods (see [Dreben and Goldfarb 1979]). E.g., it can be shown that any model of a monadic formula F can be mapped into a model of F of cardinality $\leq 2^p$, where p is the number of different predicate symbols in F. Clearly, the resolution based decision procedure is more feasible than an exhaustive search for a model among all possible interpretations of cardinality $\leq 2^p$ (unless p is extremely small). Moreover, it has the advantage that any resolution based prover (e.g., OTTER [McCune 1995], SPASS [Weidenbach, Gaede and Rock 1996], or Gandalf [Tammet 1997]) can be used for that purpose, provided it allows to define simple ordering restrictions as mentioned above.

3.2. A-orderings

The ordering that we have used for the monadic class in the example above is just a special case of the well known concept of A-orderings introduced in Kowalski and Hayes [1969]. We present a definition, that is slightly more general than that of Kowalski and Hayes.

3.1. Definition. An A-ordering $<$ is a binary relation on atoms such that

1. $<$ is irreflexive,
2. $<$ is transitive,
3. for all atoms A,B and all substitutions θ: $A < B$ implies $A\theta < B\theta$.

A subtle point concerns the definition of the resolvents that obey the A-ordering restriction. Whereas Kowalski and Hayes demand that the atoms to be resolved upon are maximal with respect to the ordering, we use the following definition which is due to Joyner [1976].

3.2. Definition. For any clause set S and any A-ordering $<$: $E \in R_<(S)$ if E is a (Robinson-)resolvent of clauses in S such that for no atom B in E: $A < B$, where A is the resolved atom.

This means that we use an *a posteriori criterion* as opposed to an *a priori criterion* for the applicability of the resolution rule.

It is interesting to observe that in order to guarantee the termination of proof search we have to insist on the stronger a posteriori criterion in some cases.

The completeness of A-ordering resolution is readily shown by a semantic tree argument. The importance of this proof technique lies in the fact that additional techniques like condensation, subsumption and certain saturation techniques can be shown to preserve completeness by the same argument. For a proof of this fact and the theorems and lemmas below, we refer to [Fermüller, Leitsch, Tammet and Zamov 1993].

3.3. Covering terms and atoms

Instead of investigating "classical" decision classes (i.e., the prefix classes described in [Dreben and Goldfarb 1979]) in a case by case manner we first introduce some additional terminology that reveals common features of the term structure in corresponding clauses and, more importantly, leads to natural generalizations of those classes in the context of clause logic.

Throughout the next sections we speak of terms, atoms and literals characterized by special properties which we call *covering* and *weakly covering*, respectively.

3.3. Definition. A functional term t is called *covering* if for all functional subterms s occurring in t we have $V(s) = V(t)$. An atom or literal A is called covering if each argument of A is either a constant or a variable or a covering term t such that $V(t) = V(A)$.

3.4. Definition. A functional term t is called *weakly covering* if for all non ground, functional subterms s of t: $V(s) = V(t)$. Similarly to Definition 3.3, an atom or literal A is called *weakly covering* if each argument of A is either a ground term or a variable or a weakly covering term t such that $V(t) = V(A)$.

3.5. Example. $g(h(x), a, f(x, x))$ is a covering term; $f(x, g(h(a), f(y, x), y))$ is weakly covering but not covering; $g(f(x, y), x, h(x))$ is neither covering nor weakly covering. $P(h(x), x)$ is a covering and $Q(f(x, f(x, y)), f(y, x), h(a))$ a weakly covering atom; $P(f(a, h(b)), h(c))$, like any ground atom, is covering, too.
$P(h(x), y)$ and $P(h(x), f(x, y))$ are not weakly covering.

Clearly, all covering atoms or terms are also weakly covering, but the converse does not hold. Covering atoms originate for instance by skolemization of (function free) prenex formulas where the existential quantifiers are in the scope of all or none of the universal quantifiers. Also observe that any atom or term that contains at most one variable is weakly covering.

The essential facts needed to establish a bound on the term depth as well as the length of resolvents for certain classes of clause sets, are summarized in the following lemmas.

3.6. LEMMA. *Let θ be an mgu of two covering atoms A and B. Then the following properties hold for $A\theta$ $(= B\theta)$:*

1. *$A\theta$ is covering,*
2. $\tau(A\theta) = \max(\tau(A), \tau(B))$
3. $|V(A\theta)| \leq \max(|V(A)|, |V(B)|)$.

Similar properties can be shown to hold for weakly covering atoms as well.

The following property of covering atoms is essential, too:

3.7. LEMMA. *Let A and B be two (weakly) covering atoms such that $V(A) = V(B)$. For any substitution θ: If $A\theta$ is (weakly) covering then $B\theta$ is (weakly) covering, too.*

3.4. Classes $\mathcal{E}_1$ and $\mathcal{E}^+$

The concept of (weakly) covering literals permits concise definitions of some decision classes. Consider, e.g., the following class of clause sets:

3.8. DEFINITION *($\mathcal{E}_1$).* A clause set S belongs to $\mathcal{E}_1$ if the following holds for all clauses C in S:

1. All literals in C are covering, and
2. for all literals $L, M \in C$ either $V(L) = V(M)$ or $V(L) \cap V(M) = \emptyset$.

3.9. EXAMPLE. Let $C_1 = \{P(f(x,y),a),\ Q(y,x,x)\}$, $C_2 = \{P(f(x,f(x,a))), Q(z,y,a)\}$, $C_3 = \{P(x,f(a))\}$ and $C_4 = \{Q(x,y,a),\ P(x,x)\}$. Then $\{C_1, C_2\} \in \mathcal{E}_1$, but any clause set containing C_3 or C_4 is not in $\mathcal{E}_1$.

$\mathcal{E}_1$ may be regarded as an extension of the initially extended Ackermann class (characterized by the prefix type $\exists^*\forall\exists^*$).

The class $\mathcal{E}_1$ has the pleasant property that for every clause set S in $\mathcal{E}_1$ and every resolvent or factor C of clauses in S, the number of variables in any split component of C will not exceed the maximal number of variables in its parent clauses. However, there is no bound on the term depth of clauses in $R(S)$. To guarantee a term depth limit for resolvents of clauses in $\mathcal{E}_1$, a simple A-ordering strategy can be used.

3.10. DEFINITION. Let A and B be two atoms, then $A <_d B$ if

1. $\tau(A) < \tau(B)$, and
2. for all $x \in V(A)$: $\tau_{\max}(x, A) < \tau_{\max}(x, B)$ (implying $V(A) \subseteq V(B)$).

It is easy to show that $<_d$ is an A-ordering. It can also be proved that, for every $S \in \mathcal{E}_1$, every resolvent in $R_{<_d}(S)$ fulfills the defining conditions of class $\mathcal{E}_1$ and is smaller or equal with respect to term depth than its parent clauses. This is mainly a consequence of Lemma 3.6 and Lemma 3.7. To get a decision procedure we have to bound the length of resolvents, too. For this we may either use the splitting rule (see Definition 2.26) or the condensation rule (see Definition 2.20).

3.11. Theorem. *The class $\mathcal{E}_1$ is decidable; $R_{<_d}$ combined with the splitting or condensation rule provides a decision procedure.*

$\mathcal{E}_1$ remains decidable if we slightly generalize it as follows:

3.12. Definition *($\mathcal{E}^+$).* A clause set S belongs to $\mathcal{E}^+$ if, for all clauses C in S, the following properties hold:

1. All literals in C are weakly covering, and
2. for all literals $L, M \in C$ either $V(L) = V(M)$ or $V(L) \cap V(M) = \emptyset$.

3.13. Example. Let $C_1 = \{P(f(x,y),a),\ Q(y,x,x)\}$, $C_2 = \{P(f(x,f(x,a)),\ Q(z,y,a)\}$, $C_3 = \{P(x,f(a))\}$ and $C_4 = \{Q(x,y,a), P(x,x)\}$. Then $\{C_1, C_2, C_3\} \in \mathcal{E}^+$, but any clause set containing C_4 is not in $\mathcal{E}^+$.

$\mathcal{E}^+$ clearly is a strict superset of $\mathcal{E}_1$, because arbitrary ground terms are now additionally allowed to occur everywhere in the clauses. $\mathcal{E}^+$ also contains the class of clause sets only consisting of clauses C such that $|V(C)| \leq 1$. This subclass has first been proved decidable by Y. Gurevich [1973] and is often called class $\mathcal{E}$, which motivated the name $\mathcal{E}^+$ for the class defined above.

$R_{<_d}$ combined with the splitting rule or condensation also provides a decision procedure for $\mathcal{E}^+$ [de Nivelle 1998*a*]. However, unlike $\mathcal{E}_1$, the term depth as well as the maximal variable depth of a resolvent can be greater than the term depth and maximal variable depth of its parent clauses.

3.14. Example. Consider the clauses $C_1 = \{\neg P(x, s(s(0))),\ P(s(x), s(s(0)))\}$ and $C_2 = \{\neg P(y, s(s(0))),\ P(s(y), s(s(0)))\}$. Clearly, $\{C_1, C_2\} \in \mathcal{E}^+$. The resolvent of C_1 and C_2 using the most general unifier $\sigma = \{y \leftarrow s(x)\}$ is $C_3 = \{\neg P(x, s(s(0)))),\ P(s(s(x)), s(s(0)))\}$. Note that in $C_1\sigma = C_1$ and in $C_2\sigma = \{\neg P(s(x), s(s(0))),\ P(s(s(x)), s(s(0)))\}$ both literals are maximal with respect to $<_d$. The maximal variable depth of C_3 is greater than the maximal variable depth of C_1 and C_2.

The resolvent $\{Q(s(0), y, f(s(0), y))\}$ of $\{P(x, y, f(x, y)),\ Q(x, y, f(x, y))\}$ and $\{\neg P(s(0), y, z)\}$ has a greater term depth than its parent clauses.

It can be proved that for every $S \in \mathcal{E}^+$, every resolvent C in $R_{<_d}(S)$ fulfills the defining conditions of class $\mathcal{E}^+$, and the maximal variable depth and the term depth of C does not exceed $\tau_v(S) + \tau(S)$ [de Nivelle 1998*a*].

3.15. Theorem. *The class $\mathcal{E}^+$ is decidable; $R_{<_d}$ combined with the splitting or condensation rule provides a decision procedure.*

It is mandatory that $<_d$ is used as an a posteriori criterion as the following example illustrates.

3.16. EXAMPLE. Consider the clauses $C_1 = \{\neg P(x, s(s(0))),\ P(s(x), s(s(0)))\}$ and $C_4 = \{P(0, s(s(0)))\}$. We have already observed in the previous example, that both literals in C_1 are maximal with respect to $<_d$. Thus, if $<_d$ is used as an a priori criterion, then we obtain an unbounded sequence of clauses $\{P(s(0), s(s(0)))\}$, $\{P(s(s(0)), s(s(0)))\}, \ldots, \{P(s^i(0)), s^2(0)\}, \ldots$ by resolving on the first literal of C_1.

There are various alternative decision procedures for the class $\mathcal{E}^+$. In Section 3.7 we present a resolution refinement based on a nonliftable ordering which is able to decide $\mathcal{E}^+$. A decision procedure based on a combination of ordering refinements with a saturation mechanism has been used in [Fermüller et al. 1993, chapter 5]. We illustrate this approach in the next section for the more interesting case of the Skolem class.

3.5. An extension of the Skolem class

The (initially extended) Skolem class is the class of prenex formulas with a prefix of the form $(\exists z_1)\cdots(\exists z_l)(\forall y_1)\cdots(\forall y_m)(\exists x_1)\cdots(\exists x_n)$ such that each atom of the matrix has among its arguments either

1. at least one of the x_i, or
2. at most one of the y_i, or
3. all of $y_1, \ldots y_m$.

In this section we consider a class of clause sets which strongly generalizes the initially extended Skolem class.

3.17. DEFINITION *($\mathcal{S}^+$).* A clause set S belongs to $\mathcal{S}^+$ if for all clauses C in S and all literals L in C:

1. If t is a functional term occurring in C then $V(t) = V(C)$.
2. $|V(L)| \leq 1$ or $V(L) = V(C)$.

Class $\mathcal{S}^+$ contains the clause forms of all formulas in the initially extended Skolem class, which in turn contains the initially extended Gödel class (i.e. the prefix class with quantifier prefix type $\exists^*\forall\forall\exists^*$).

$\mathcal{S}^+$ is related to $\mathcal{E}_1$. In fact we have:

$$E \in R_{<_d}(\{C, D\}) \qquad \text{implies} \qquad \tau(E) \leq \tau(\{C, D\})$$

if $\{C, D\} \in \mathcal{S}^+$. The only problem is that in general $R_{<_d}(S) \notin \mathcal{S}^+$ for clause sets $S \in \mathcal{S}^+$. Atoms may be generated which, besides covering terms, have arbitrary variables as arguments.

To be able to argue more accurately we define:

3.18. DEFINITION. An atom or literal A is called *essentially monadic on a term t* if t is an argument of A and each other argument is either equal to t or a constant. A is called *almost monadic on t* if t is functional and, besides t and constants, also at least one variable that is not in $V(t)$ occurs among the arguments of A.

3.19. Example. The literal $P(g(x), b, g(x))$ is essentially monadic on $g(x)$. The literal $P(f(f(z)), x, a, f(f(z)))$ is almost monadic on $f(f(z))$. In contrast, $P(x, y, a, x)$ and $Q(f(x), f(y))$ are neither essentially monadic nor almost monadic (on any term).

As mentioned above $R_{<_d}$ provides a term depth limit for the resolvents of $\mathcal{S}^+$. But by resolving such clauses, almost monadic atoms may be generated besides covering ones.

3.20. Example. Let $C = \{P(x), Q(x, y)\}$ and $D = \{\neg P(f(z)), Q(g(z), z)\}$. The only $R_{<_d}$-resolvent of C and D is $E = \{Q(f(z), y), Q(g(z), z)\}$. The first literal of E is almost monadic on $f(z)$; the second literal is covering and essentially monadic.

Observe that, for the class $\mathcal{S}^+$, noncovering (but almost covering) atoms are generated by $R_{<_d}$ only if one of the parent clauses is function free, and the atom(s) resolved upon in this clause contain(s) one variable only, whereas other atoms must also have additional variables as arguments. In all other cases all atoms of a resolvent are covering. It is an easy task to refine the ordering $<_d$ in a way such that it becomes sufficiently restrictive to terminate on all inputs from $\mathcal{S}^+$ (and many other classes). We just have to add

4. $V(A) \subset V(B)$ implies $A <_d B$ for function free atoms A and B

to the defining conditions for $<_d$ (Definition 3.10). Unfortunately, the resulting resolution variant is not an A-ordering. Still, using techniques developed by Hans de Nivelle one can show that completeness is preserved if the inputs are from $\mathcal{S}^+$ [de Nivelle n.d.]. However, one looses compatibility with subsumption and other deletion strategies. Here, we describe another useful method to guarantee termination *and* completeness, namely *saturation*.

For any almost monadic atom we define a corresponding set of essentially monadic atoms.

3.21. Definition. Let A be almost monadic on some functional term t and CS be some set of constants. The *monadization* $MON(A, \mathrm{CS})$ consists of atoms that are like A except for replacing each variable that occurs as argument of A (but not of t) by t or some constant in CS. More exactly: Let $\Sigma_{t,\mathrm{CS}}$ be the set of all substitutions of the form $\{x_i \leftarrow t_i \mid x_i \in V(A) - V(t)\}$ where $t_i = t$ or $t_i \in \mathrm{CS}$ then $MON(A, \mathrm{CS}) = \{A\sigma \mid \sigma \in \Sigma_{t,\mathrm{CS}}\}$.

We extend the definition of MON to clauses and clause sets: If all almost monadic atoms A of a clause C are almost monadic on the same functional term t then $MON(C, \mathrm{CS}) = \{C\sigma' \mid \sigma' \in \Sigma'_{t,\mathrm{CS}}\}$ where Σ'_t is the set of substitutions $\{x_i \leftarrow t_i \mid x_i \in V(C) - V(t)\}$, such that $t_i = t$ or $t_i \in \mathrm{CS}$.

If C is function free and there is one and only one variable x, such that all literals $L \in C$ with $|V(L)| \geq 2$ contain x then $MON(C, \mathrm{CS}) = \{C\sigma \mid \sigma \in \Sigma_{x,\mathrm{CS}}\}$ where $\Sigma_{x,\mathrm{CS}}$ is the set of substitutions $\{x_i \leftarrow t_i \mid x_i \in V(C) - V(t)\}$, such that $t_i = x$ or $t_i \in \mathrm{CS}$.

In all other cases $MON(C, \mathrm{CS}) = \{C\}$.

For any clause set S: $MON(S) = \bigcup_{C \in S} MON(C, CS(S))$ (where $CS(S)$ is the set of all constants occurring in clauses of S).

3.22. REMARK. As we only use the monadization operator in the context of finite clause sets S we will implicitly assume that the set of constants occurring in S is always used for the monadization of atoms or clauses. For sake of readability we therefore suppress the second argument and write $MON(A)$ and $MON(C)$ in stead of $MON(A, \mathrm{CS})$ and $MON(C, \mathrm{CS})$, respectively. Observe that, in our context, $MON(A)$ and $MON(C)$ are always finite.

3.23. EXAMPLE. For all examples we assume that there is just one constant a. Let $A = P(f(x,y),z)$ then $MON(A) = \{P(f(x,y),f(x,y)),\ P(f(x,y),a)\}$. Let $C = \{Q(u,x,f(x,a)),\ P(u,f(x,a))\}$ then $MON(C) = \{\{Q(f(x,a),x,f(x,a)),\ P(f(x,a),f(x,a))\},\ \{Q(a,x,f(x,a)),\ P(a,f(x,a))\}$. For $D = \{P(x,x), Q(u,a,x),\ P(x,v)\}$ we have $MON(D) = \{\{P(x,x),\ Q(x,a,x)\},\ \{P(x,x),\ Q(x,a,x), P(x,a)\},\ \{P(x,x),\ Q(a,a,x)\},\ \{P(x,x),\ Q(a,a,x), P(x,a)\}\}$.

We may now define a new resolution variant R_m which is based on $R_{<_d}$.

3.24. DEFINITION. For any clause set S:

$$R_m(S) = MON(R_{<_d}(S)).$$

The members of $R_m(S)$ are called R_m-resolvents.

It can be shown that for any $\{C, D\} \in \mathcal{S}^+$ and $E \in R_m(\{C, D\})$ we have $\tau(E) \leq \tau(\{C, D\})$. Moreover, the subclauses of E resulting from splitting fulfill the defining conditions of $\mathcal{S}^+$.

We thus arrive at the following result:

3.25. THEOREM. *$\mathcal{S}^+$ is decidable; R_m combined with the splitting rule or condensation provides a decision procedure.*

3.6. Nonliftable orderings

We have seen above that refinements based on A-orderings can be used to define decision procedures for various classes of clause sets. However, other types of ordering restrictions are important, as well. In order to achieve new decidability results or more efficient decision procedures, it will be important to pay attention to these refinements. Therefore we conclude with a discussion of the particularly interesting case of the so-called *nonliftable orderings*.

Recall the Definition 3.1 of A-orderings. The binary relation $<_R$ is an A-ordering iff it is irreflexive, transitive and satisfies the *liftability* condition: for all atoms A,B and all substitutions θ: $A <_R B$ implies $A\theta <_R B\theta$.

Although sufficient, liftability is not always a necessary condition for an irreflexive and transitive relation $<_R$ to be complete. First of all the definition of liftability

can be weakened to deal not with arbitrary instances but only with ground instances appearing in an unsatisfiable set of ground instances. This observation is used in [de Nivelle 1998*a*] to define a resolution refinement which is complete and terminating on $\mathcal{E}^+$, but also liftable in the weaker sense. On $\mathcal{E}^+$ this ordering coincides with the A-ordering $<_d$ and thus yields an (a posteriori) A-ordering decision procedure for this class.

Moreover there exist a number of criteria which guarantee completeness for strong nonliftable orderings. A minimal requirement for a nonliftable ordering $\succ$ to be compatible with the resolution inference rule is that $\succ$ is invariant under renaming:

3.26. Definition. An ordering $\succ$ is invariant under renaming if for any two literals A and B such that $V(A) = V(B)$ and any renamings ϑ_1 and ϑ_2 with $rg(\vartheta_1) = rg(\vartheta_2)$, $A \succ B$ implies $A\vartheta_1 \succ B\vartheta_2$.

The renaming condition ensures that we can rename the variables of clauses C and D to ensure variable-disjointness before deriving a resolvent of C and D without affecting the $\succ$-maximality of literals:

3.27. Lemma. *Let $\succ$ be an ordering which is invariant under renaming. Let C be a clause and σ be a variable renaming. If a literal L is $\succ$-maximal in C, then $L\sigma$ is $\succ$-maximal in $C\sigma$.*

Now, the following question is an important open problem:
Does every ordering that is invariant under renaming define a complete resolution refinement?
Partial solutions to this problem have been presented by H. de Nivelle and T. Tammet. H. de Nivelle [1997] shows the completeness for orderings invariant under renaming on Krom clauses. A proof by Tammet [1996] shows that the following orderings $<_g$ and $<_{ng}$ are complete, if the a priori criterion is used to select the resolvent:

3.28. Definition. For any two literals A and B, $A <_g B$ if A is not ground and B is ground. Analogously, $A <_{ng} B$ if A is ground and B is not ground.

Completeness is preserved if $<_g$ is strengthened by arbitrary completeness-preserving orderings for the ground and nonground cases:

3.29. Definition. Let $<_s$ be a completeness-preserving ordering of nonground literals and $<_r$ be an ordering of ground literals. Then, for any two literals A and B, $A <_{gsr} B$ if either A is not ground and B is ground, both A and B are ground and $A <_r B$ or both A and B are nonground and $A <_s B$.

The resolution refinement based on $<_g$ is also complete if used as a posteriori criterion. However, this is not true for the ordering $<_{ng}$ as the following example shows.

3.30. Example. Let $C_1 = \{\neg P(a)\}$, $C_2 = \{\neg Q(a)\}$, and $C_3 = \{P(x), Q(y)\}$. The set $S = \{C_1, C_2, C_3\}$ is obviously unsatisfiable. There are only two resolution inference steps possible on S. We can either resolve C_1 and C_3 with most general unifier $\sigma_1 = \{x \leftarrow a\}$ or C_2 and C_3 with most general unifier $\sigma_2 = \{y \leftarrow a\}$. However, the literal $P(a)$ is not maximal in $C_3\sigma_1 = \{P(a), Q(y)\}$ with respect to $<_{ng}$ nor is $Q(a)$ maximal in $C_3\sigma_2 = \{P(x), Q(a)\}$. If we use $<_{ng}$ as a posteriori criterion, then none of these resolution inference steps by the resolution refinement based on $<_{ng}$ is possible. Hence, we obtain no refutation of S.

De Nivelle [1995] shows that any irreflexive and transitive relation $\succ$ which is invariant under renaming is complete for sets of *decomposed clauses*, i.e. clauses where all literals contain the same variables (the orderings are applied a priori).

Although nonliftable orderings are useful to obtain resolution decision procedures for a variety of classes, as exemplified in the following section, they are typically not compatible with the strategies of full subsumption and tautology removal. Moreover we cannot use the a posteriori but only the a priori criterion for selecting resolvents.

3.7. Classes decidable by nonliftable orderings

The book [Fermüller et al. 1993] presents a few orderings which are nonliftable, but are complete for some classes, for example the class $\mathcal{E}^+$ and the class $\mathcal{K}$ of Maslov.

It follows from the result of de Nivelle [1995] that completeness for $\mathcal{E}^+$ is preserved if the condition (1): $\tau(A) < \tau(B)$ is dropped from the previous Definition 3.10 of an A-ordering $<_d$, keeping only the condition (2): for all $x \in V(A)$: $\tau_{\max}(x, A) < \tau_{\max}(x, B)$. Although the ordering fulfilling just (2) does not obey the renaming condition in general, it does so on $\mathcal{E}^+$. There it coincides with the ordering $<_\tau$ defined as: $A <_\tau B$ iff $\tau_v(A) < \tau_v(B)$. The ordering $<_\tau$ is invariant under renaming and thus is complete on $\mathcal{E}^+$ but it is not an A-ordering. As discussed in Section 3.4, class $\mathcal{E}^+$ can be decided also by a liftable a posteriori ordering.

The class $\mathcal{K}$ was introduced by Maslov [1971]. The following definitions are based on [Fermüller et al. 1993, chapter 6].

3.31. Definition *($\mathcal{K}$).* Let φ be a closed formula in negation normal form without nonconstant function symbols such that no variable is bound by two different quantifier occurrences. Let ψ be a subformula of φ. The *φ-prefix* of the formula ψ is a sequence of quantifiers of the schema φ which bind the free variables of ψ.

Then φ belongs to the class $\mathcal{K}$ if there exist variables $x_1, \ldots, x_k$, $k \geq 0$, which are not in the scope of any existential quantifier, and for every atomic subformula ψ of φ the φ-prefix of φ

1. either has length less or equal to 1, or
2. ends with an existential quantifier, or
3. is of the form $(\forall x_1)(\forall x_2) \ldots (\forall x_k)$

It is straightforward to see that $\varphi \in \mathcal{K}$ iff $\mathit{cnf}(\mathit{prenex}(\varphi)) \in \mathcal{K}$, where *cnf* denotes a syntax preserving transformation of first-order formulas to conjunctive normal

form based on the rule of distributivity. Thus, without loss of generality we can restrict ourselves to formulas in prenex form whose matrix is in conjunctive normal form, that is, formulas in $\mathcal{K}$ have the form

$$(\forall x_1)\dots(\forall x_k)(Q_1 z_1)\dots(Q_l z_l)\bigwedge_{i=1,\dots,n}\bigvee_{j=1,\dots,m_i} L_{i,j} \tag{3.1}$$

where $m \geq 0$, $k \geq 0$, $l \geq 0$, $n > 0$, $m_i > 0$, and $L_{i,j}$ are literals. The class of clause sets obtained from formulas (3.1) in $\mathcal{K}$ is denoted by $\mathcal{KC}$.

We define an ordering on terms and literals as follows.

3.32. Definition. Let s and t be terms, then $s \lesssim_Z t$ if at least one of the following conditions is satisfied:

1. $t = f(t_1, \dots, t_n)$, $s = g(t_1, \dots, t_m)$, $n \geq m \geq 0$.
2. s is a variable and either $s \in \mathit{args}(t)$ or $s = t$.

Let A and B be two atoms, then $A \lesssim_Z B$ if for every nonconstant argument term s of B there exists a nonconstant argument term t of A such that $t \lesssim_Z s$.

We define $\sim_Z$ as $\lesssim_Z \cap \lesssim_Z^{-1}$ and $<_Z$ as $\lesssim_Z / \sim_Z$ on terms and atoms.

The ordering $<_Z$ is not liftable, but it is invariant under renamings. It follows from the results in [Fermüller et al. 1993] that any derivation from a clause set S in $\mathcal{KC}$ by a combination of $R_{<_Z}$ and splitting terminates. By the result of de Nivelle [1995], the refinement $R_{<_Z}$ is complete.

3.33. Theorem. *$\mathcal{KC}$ is decidable; $R_{<_Z}$ combined with the splitting rule provides a decision procedure.*

Recently, Hustadt and Schmidt [1999*b*] have shown that $\mathcal{KC}$ as well as the class $\mathcal{DKC}$ containing the clause sets obtained from conjunctions of formulas in $\mathcal{K}$ are decidable using a resolution refinement based on a liftable ordering.

A fragment of first-order logic which currently attracts much attention is the *guarded fragment* proposed by Andréka, van Benthem and Németi [1995, 1998]. It is an attempt to characterise a class of first-order formulas sharing properties like decidability, the finite model property and the tree model property with modal logics.

3.34. Definition *($\mathcal{GF}$).* The guarded fragment $\mathcal{GF}$ is a subclass of first-order logic without nonconstant function symbols which is inductively defined as follows:

1. $\top$ and $\bot$ are in $\mathcal{GF}$,
2. if A is an atom, then A is in $\mathcal{GF}$,
3. $\mathcal{GF}$ is closed under boolean combinations,
4. if φ is in $\mathcal{GF}$, A is an atom, and $\bar{x}$ is a sequence of variables, such that every free variable of φ is an argument of A, then $(\forall \bar{x})\,(A \to \varphi)$ and $(\exists \bar{x})\,(A \wedge \varphi)$ are in $\mathcal{GF}$. The atom A is called a *guard*.

Andréka, van Benthem and Németi [1995] show that every satisfiable formula in the guarded fragment has a model of finite size. Thus, the guarded fragment is decidable. The first resolution decision procedure for the guarded fragment has been described by de Nivelle [1998*b*].

Let φ be a guarded formula in negation normal form. Let Λ_φ be the set of positions of subformulas in φ of the form $\forall \bar{x}(\neg A \vee \psi)$. By $\Xi_{\mathcal{GF}}$ we denote the transformation taking φ to its definitional form $\mathrm{Def}_{\Lambda_\varphi}(\varphi)$.

3.35. EXAMPLE. The definitional form of the guarded formula

$$(\exists x)\,(Q(x) \wedge (\forall y)\,(\neg R(x,y) \vee (\forall z)\,(\neg P(x,z) \vee (\exists v)\,(R(v,z) \wedge (B(z,z) \vee C(v))))))$$

is

$$\begin{aligned}&(\exists x)\,(Q(x) \wedge Q_1(x))\\ \wedge\ &(\forall x)(Q_1(x) \rightarrow (\forall y)\,(\neg R(x,y) \vee Q_2(x)))\\ \wedge\ &(\forall x)(Q_2(x) \rightarrow (\forall z)\,(\neg P(x,z) \vee (\exists v)\,(R(v,z) \wedge (B(z,z) \vee C(v))))).\end{aligned}$$

The standard clausal form consists of the clauses

$$\begin{aligned}&\{Q(x)\}, \{Q_1(x)\},\\ &\{\neg Q_1(x), \neg R(x,y), Q_2(x)\},\\ &\{\neg Q_2(x), \neg P(x,z), R(f(x,z),z)\},\\ &\{\neg Q_2(x), \neg P(x,z), B(z,z), C(f(x,z))\}.\end{aligned}$$

Observe that $\Xi_{\mathcal{GF}}$ applied to a guarded formula φ yields again a guarded formula. Furthermore, all remaining universal quantifiers are outermost and any existential quantifier occurs in the scope of all universally quantified variables. The standard clausal form of $\Xi_{\mathcal{GF}}(\varphi)$ can then be characterized as follows.

3.36. DEFINITION *(Weakly guarded clause).* A clause C is *weakly guarded* if

1. all literals $L \in C$ are weakly covering,
2. if C is nonground, then there is a negative literal $L \in C$ with $\tau_v(L) = 0$ and $V(L) = V(C)$, and
3. if $\tau_v(L) > 0$, then $V(L) = V(C)$.

A literal L is *simple* if L is covering and the term depth of L is less than or equal to 1. If we replace the first condition above by the requirement that all literals L in C are simple, then C is a *guarded clause*. A clause set is *(weakly) guarded* if its clauses are (weakly) guarded. By $\mathcal{GC}$ and $\mathcal{WGC}$ we denote the class of all guarded and weakly guarded clause sets, respectively.

It is straightforward to see that the class $\mathcal{GC}$ is a subclass of $\mathcal{WGC}$.

3.37. THEOREM. *Let φ be a conjunction of guarded formulas. The standard clausal form of $\Xi_{\mathcal{GF}}(\varphi)$ is a guarded clause set.*

The resolution decision procedure of de Nivelle [1998*b*] is based on the following ordering $<_{\mathcal{GF}}$: $L_1 <_{\mathcal{GF}} L_2$ if $\tau_v(L_1) < \tau_v(L_2)$ or $V(L_1) \subset V(L_2)$. The ordering $<_{\mathcal{GF}}$ is not liftable, but is invariant under renaming. Every guarded clause C contains a literal L which is $<_{\mathcal{GF}}$-maximal in C and L contains all variables of C.

3.38. Theorem. *$\mathcal{WGC}$ is decidable; $R_{<_{\mathcal{GF}}}$ provides a decision procedure.*

Recently, Ganzinger and de Nivelle [1999] presented a resolution decision procedure for the class $\mathcal{GC}$ with equality based on the superposition calculus of Bachmair and Ganzinger [1994]. In contrast to the approach of de Nivelle [1998*b*], it makes use of a liftable ordering and an additional selection function on negative literals. Ganzinger and de Nivelle [1999] also show that the satisfiability problem for the class $\mathcal{WGC}$ with equality is undecidable.

4. Hyperresolution as decision procedure

4.1. Classes decidable by hyperresolution

Refinements based on A-orderings or on other types of orderings do not suffice to obtain decision procedures for all relevant decision classes. Particularly in the case of the Bernays-Schönfinkel class and of many functional clause classes hyperresolution is superior to ordering refinements. On the other hand hyperresolution is not suited as decision method for the one-variable or for the Ackermann class. Thus we will discuss some syntactic features, which characterize the applicability of decision methods of different types. First we give formal definitions of hyperresolution and the corresponding refinement operator:

4.1. Definition. Let C, D be condensed clauses, where D is positive. The condensation of a binary resolvent of C and a factor of D is called a *PRF-resolvent* (PRF abbreviates "positive, restricted factoring").

Remark. Throughout this section we assume that clauses always appear in condensed form, mostly without mentioning this fact explicitly.

4.2. Definition. Let C be a nonpositive clause and let the clauses D_i, for $1 \leq i \leq n$, be positive. Then the sequence $\Gamma = (C; D_1, \ldots, D_n)$ is called a *clash sequence*.

Let $C_0 = C$ and C_{i+1} be a PRF-resolvent of C_i and D_{i+1} for $i < n$. If C_n is positive then it is called a *clash (or hyper)resolvent* defined by Γ.

Hyperresolution is the oldest refinement of resolution [Robinson 1965*a*] and exemplifies the principle of macro inference. It only produces positive clauses or the empty clause $\Box$. In variance to the standard definition of hyperresolution we have included a restriction on factoring. The concept of "semi-factoring" is investigated in [Noll 1980], where—inter alia—it is shown that positive hyperresolution based on PRF-resolution is complete.

Below, we do not need to refer to hyperresolution deductions themselves but rather are interested in the set of derived clauses. For this purpose the following operator based description of hyperresolution seems most adequate.

4.3. DEFINITION. Let S be a set of clauses. By $\rho_H(S)$ we denote the set of all clash resolvents definable by clash sequences of clauses in S. The hyperresolution operator R_H and its closure R_H^* is defined by:

$$R_H(S) = S \cup \rho_H(S),$$

$$R_H^0(S) = S \text{ and } R_H^{i+1}(S) = R_H(R_H^i(S)) \text{ for } i \leq 0.$$

$$R_H^*(S) = \bigcup_{i \geq 0} R_H^i(S).$$

The satisfiability problem of the following class of first-order formulas is decidable, as the Herbrand universes of the corresponding skolemized forms are finite. Although decidability is easy to prove by model theoretic means, the behavior of resolution on this class is problematic.

4.4. DEFINITION. Let $\mathcal{BS}$ be the class of all closed formulas of the form

$$(\exists x_1) \ldots (\exists x_m)(\forall y_1) \ldots (\forall y_m) M,$$

where M is quantifier-free and all terms occurring in M are variables. Then $\mathcal{BS}$ is called the Bernays-Schönfinkel class. If, in addition, M is a conjunction of Horn clauses then we obtain the subclass $\mathcal{BSH}$.

Because the skolemization M' of a formula M in $\mathcal{BS}$ does not contain function symbols, all Herbrand interpretations of M' have a fixed finite domain. Thus a trivial decision procedure for satisfiability consists in evaluating M' over all its Herbrand interpretations. Here, however, we are interested in the termination behavior of resolution refinements on $\mathcal{BS}$ and $\mathcal{BSH}$. For this purpose we have to define clauses classes corresponding to $\mathcal{BS}$ and $\mathcal{BSH}$.

4.5. DEFINITION. $\mathcal{BS}^*$ is the class of all finite sets of clauses S such that for all $C \in S : \tau(C) = 0$. $\mathcal{BSH}^*$ is the subclass of $\mathcal{BS}^*$ containing sets of Horn clauses only.

The condition $\tau(C) = 0$ in Definition 4.5 guarantees that there are no function symbols in S. All constant symbols appearing in a set $S \in \mathcal{BS}^*$ can be thought to have been introduced by skolemization. Thus $\mathcal{BS}^*$ is exactly the clause class corresponding to $\mathcal{BS}$; similarly $\mathcal{BSH}^*$ is the clause class corresponding to $\mathcal{BSH}$. $\mathcal{BS}^*$ and $\mathcal{BSH}^*$ can be decided by (total) saturation:

Take a $S \in \mathcal{BSH}^*$, replace S by the set of all ground instances S' and then decide S' by a propositional method (e.g. propositional resolution or the Davis-Putnam method).

This method can be very inefficient due to the fact that S' may be much larger than S itself. It is also an interesting fact that $\mathcal{BS}$ is of highest computational complexity among the classical prefix classes [Denenberg and Lewis 1984].

We now investigate how A-ordering refinements behave on $\mathcal{BS}$ and $\mathcal{BSH}$.

4.6. Example. Let $S : \{C_1, C_2, C_3, C_4\}$ be the following set of clauses in $\mathcal{BSH}^*$: $C_1 = \{P(a,b)\},\ C_2 = \{P(x,y), \neg P(y,x)\},\ C_3 = \{P(x,z), \neg P(x,y), \neg P(y,z)\}$, $C_4 = \{\neg P(b,c)\}$.

C_2 represents symmetry and C_3 transitivity. S is satisfiable because $\{P(b,c)\}$ cannot be obtained from $\{P(a,b)\}$ using the rules of symmetry and transitivity. We show now that all A-ordering refinements are nonterminating on S, i.e. there exists no A-ordering refinement which decides $\mathcal{BSH}$ (note that S even is a set of Horn clauses).

Indeed $R^*_{<_A}(\{C_2, C_3\}) = R^*_{\emptyset}(\{C_2, C_3\})$ for every A-ordering $<$. The reason for this effect is the unifiability of resolved atoms with all atoms in the resolvent (for all R-derivable clauses); therefore the case $L < M$ can never occur and no resolvents can be excluded. Particularly $R^*_{<_A}(\{C_2, C_3\})$ contains the infinite sequence of clauses

$$C_n = \{P(x_1, x_n), \neg P(x_1, x_2) \ldots, \neg P(x_k, x_{k+1}) \ldots, \neg P(x_{n-1}, x_n)\}$$

for $n \geq 2$. These clauses cannot be removed by subsumption nor do they "collapse" by condensing (they are in fact condensed). Thus, even with subsumption, no A-ordering terminates on S. On the other hand we obtain

$$R^*_H(S) = S \cup \{\{P(b,a)\},\ \{P(a,a)\},\ \{P(b,b)\}\}$$

and thus R^*_H terminates on S. Moreover the set of unit clauses

$$\mathcal{M} = \{\{P(a,b)\},\ \{P(b,a)\},\ \{P(a,a)\},\ \{P(b,b)\}\}$$

defines an atomic representation of a Herbrand model of S.

We show now that termination of R^*_H can be guaranteed on $\mathcal{BSH}^*$.

4.7. Proposition. *Hyperresolution decides $\mathcal{BSH}^*$, i.e. $R^*_H(S)$ is finite for all S in $\mathcal{BSH}^*$.*

Proof. Let S be in $\mathcal{BSH}^*$. Then S is in Horn form and the positive clauses in $R^*_H(S)$ are all unit. There are no function symbols in S and no function symbols can be introduced by resolution. Therefore we obtain for the set $\mathcal{D}$ of atoms of positive clauses in $R^*_H(S)$:

$$\mathcal{D} \subseteq \{A \mid A \text{ is an atom over } \Sigma(S),\ \tau(A) = 0\}.$$

Moreover all unit clauses with atoms from $\mathcal{D}$ are (trivially) condensed and normalized with respect to renaming of variables; so we obtain that the set $\mathcal{D}$ is finite. □

4.8. EXAMPLE.

$$S = \{\{P(x_1, x_1, a)\},\ \{P(x, z, u), \neg P(x, y, u), \neg P(y, z, u)\},$$
$$\{P(x, y, u), P(y, z, u), \neg P(x, z, u)\},\ \{\neg P(x, x, b)\}\}.$$

S is non-Horn and even "essentially" non-Horn; that means there exists no sign renaming transforming S into a set of Horn clauses. In renaming P by $\neg P$ we only exchange the roles of a and b, otherwise S remains as it is. R_H^* neither terminates on S nor on the renamed form S'. R_H^* produces clauses of arbitrary length on S—even if we add subsumption (i.e. we replace R_H^* by R_{Hs}^*). Thus R_{Hs}^* + sign renaming does not terminate on S. That means hyperresolution cannot decide the Bernays-Schönfinkel class. Moreover none of the "standard" refinements terminates on S. There is, however, general semantic clash resolution over arbitrary models $\mathcal{M}$ as defined by J. Slagle [Slagle 1967]; in such a refinement only clauses which are false in $\mathcal{M}$ are derivable. So, in case S is satisfiable, we only have to choose a model of S; on such a model all clauses are true and thus semantic clash resolution does not produce any resolvents. This trick, however, can hardly be recommended as a method in resolution decision theory. Note that models should be the outcome of our procedures, not the starting point!

Of course there is the brute force method to decide $\mathcal{BS}^*$ by ground saturation. We will see later that, by an appropriate use of hyperresolution, saturation can be reduced considerably.

We show now how hyperresolution can be applied as decision procedure on functional clauses classes. These classes can be considered as generalizations of DATALOG [Ceri, Gottlob and Tanca 1990]. Formally DATALOG is a subclass of $\mathcal{BSH}$ such that all positive clauses are ground and $V(C_+) \subseteq V(C_-)$ for all other clauses.

4.9. DEFINITION. A set of clauses S belongs to $\mathcal{PVD}$ (positive variable dominated) if for all $C \in S$:

PVD-1) $V(C_+) \subseteq V(C_-)$ (C is ground for $C_- = \Box$),
PVD-2) $\tau_{\max}(x, C_+) \leq \tau_{\max}(x, C_-)$ for all $x \in V(C_+)$.

$\mathcal{PVD}$ corresponds to a subclass of a class named $\mathcal{PVD}$ in [Fermüller et al. 1993], where the properties above were "relativized" under settings. That means there might be some sign renaming γ such that $\gamma(S) \in \mathcal{PVD}$ even if S itself is not in $\mathcal{PVD}$. Take for example the set of clauses

$$S = \{\{P(x_1), Q(g(x_1, x_1))\},\ \{R(f(x_1), x_2)\},\ \{P(a)\},\ \{R(x, y), \neg Q(y)\},$$
$$\{\neg P(x), \neg P(f(x))\},\ \{\neg R(a, a), \neg R(f(b), a)\}\}$$

Obviously S is not in $\mathcal{PVD}$ (there are positive clauses containing variables and $\{R(x, y), \neg Q(y)\}$ violates PVD-1)).

But let γ be the sign renaming mapping Q to $\neg Q$, R to $\neg R$ and P remaining unchanged. Then

$$\gamma(S) = \{\{P(x_1), \neg Q(g(x_1, x_1))\},\ \{\neg R(f(x_1), x_2)\},\ \{P(a)\},\ \{Q(y), \neg R(x, y)\},$$
$$\{\neg P(x), \neg P(f(x))\},\ \{R(a, a), R(f(b), a)\}\}$$

and $\gamma(S) \in \mathcal{PVD}$.

The example above suggests the following generalization of $\mathcal{PVD}$:

4.10. Definition. A set of clauses S belongs to $\mathcal{PVD}_r$ if there exists a sign renaming γ such that $\gamma(S)$ belongs to $\mathcal{PVD}$.

The idea behind $\mathcal{PVD}$ is that the positive parts are always "smaller" than the negative ones. As hyperresolution produces positive clauses only, we may hope that the produced clauses are small too (i.e. small enough to achieve termination). Indeed the following theorem holds (see also [Leitsch 1993] and [Fermüller et al. 1993]):

4.11. Theorem. *Hyperresolution decides $\mathcal{PVD}$, i.e. for every $S \in \mathcal{PVD}$ the set $R^*_H(S)$ is finite.*

Proof. Let S be a set of clauses in $\mathcal{PVD}$. Obviously $R^*_H(S)$ is finite if the following conditions (a), (b) hold:

(a) $R^*_H(S) - S$ contains only (positive) ground clauses and

(b) $\tau(D) \leq r$ for all $D \in R^*_H(S) - S$ and $r = \max\{\tau(C_+) \mid C \in S\}$

Thus it remains to prove (a) and (b).

By Definition 4.9 $V(C_+) \subseteq V(C_-)$ for all $C \in S$; in particular $P(S)$ consists of ground clauses only. Now consider a clash sequence $\Gamma : (C; D_1, \ldots, D_n)$ for $C \in S$ and positive ground clauses $D_1, \ldots, D_n$; let E be a clash resolvent of Γ. Then $E = C_+\lambda$ where λ is the "total" substitution corresponding to the clash resolution (in fact λ is obtained by concatenation of the mgus corresponding to the single PRF-resolvents of Γ). Let L be an arbitrary (negative) literal in C_-; then, by definition of λ, $\lambda(L) = \neg A$ for some ground atom A in a clause D_i. Consequently, as all D_i are ground clauses, λ is a ground substitution. By $V(C_+) \subseteq V(C_-)$ it follows that the resolvent $E : C_+\lambda$ is a ground clause too. This proves (a).

For (b) consider, like above, a clash $\Gamma : (C; D_1, \ldots, D_n)$ and its corresponding total substitution λ. Let us assume that the D_i are all ground clauses with $\tau(D_i) \leq r$. By definition of λ we get $C_-\lambda \subseteq D_1 \cup \cdots \cup D_n$ and thus $\tau(C_-\lambda) \leq r$. We have to show $\tau(C_+\lambda) \leq r$.

If $\tau(C_+\lambda) = \tau(C_+)$ then, by definition of r, we get $\tau(C_+\lambda) \leq r$ and we are done.

If $\tau(C_+\lambda) > \tau(C_+)$ then there exists a variable $x \in V(C_+)$ with

$$\tau(C_+\lambda) = \tau_{\max}(x, C_+) + \tau(x\lambda).$$

By property PVD-2) $\tau_{\max}(x, C_+) \leq \tau_{\max}(x, C_-)$ and so

$$\tau(C_+\lambda) \leq \tau_{\max}(x, C_-) + \tau(x\lambda) \leq \tau(C_-\lambda).$$

But $\tau(C_-\lambda) \leq r$ and so $\tau(C_+\lambda) \leq r$.

□

The decision procedure for $\mathcal{PVD}$ can easily be modified to the following decision procedure for $\mathcal{PVD}_r$:

1. Search for a renaming γ such that $\gamma(S)$ is in $\mathcal{PVD}$ (if there is no such γ then $S \notin \mathcal{PVD}_r$).
2. Apply R_H^* to $\gamma(S)$.

Note that there are only finitely many sign renamings on a set of clauses and that the properties PVD-1) and PVD-2) are decidable. Thus we can always decide whether a set of clauses is in $\mathcal{PVD}_r$. Once we have found the right renaming γ, we replace S by $\gamma(S)$ and apply hyperresolution.

The class $\mathcal{PVD}$ is relatively "tight" with respect to undecidability: If we add the clause

$$T^- = \{P(x_1, x_2), P(x_2, x_3), \neg P(x_1, x_3)\}$$

(i.e. the transitivity of $\neg P$) we can encode the word problem of any equational theory (see [Fermüller et al. 1993] chapter 3.3). From the fact that there are equational theories with undecidable word problems (e.g. the theory of combinators [Stenlund 1971]) it follows that

$$\Gamma = \{S \cup \{T^-\} \mid S \in \mathcal{PVD}\}$$

is an undecidable class of clause sets.

The main point in the proof of Theorem 4.11 consists in showing that all positive clauses in $R_H^*(S)$ are ground and $\tau(R_H^*(S)) \leq d$ for some constant d. While the property PVD-1) is essential (note that T^- does not fulfil PVD-1)), PVD-2) can be replaced by a more general condition (term depth is only a specific complexity measure for literals and clauses). Particularly we obtain a more general decision class in replacing term depth by arbitrary *atom complexity measures* α fulfilling the following axioms:

1. $\alpha(A) \leq \alpha(A\theta)$ for substitutions θ and atom formulas A.
2. For all $k \in \mathbb{N}$ the set $\{A\theta \mid \theta \in \Theta_0, \alpha(A\theta) \leq k\}$ is finite, where Θ_0 is the set of all ground substitutions over a finite signature.
3. α is extended to literals by $\alpha(A) = \alpha(\neg A)$ and to clauses by $\alpha(\{L_1, \ldots, L_n\}) = \max\{\alpha(L_i) \mid 1 \leq i \leq n\}$.

For such an α we have to postulate that there exists a constant d such that for all ground substitutions θ either $\alpha(C_+\theta) \leq d$ or $\alpha(C_+\theta) \leq \alpha(C_-\theta)$ [Leitsch 1993].

If we relax the condition on $\mathcal{PVD}$, that positive clauses must be ground, we must add a somewhat stronger restriction on the behavior of the parts C_+ and C_- relative to each other. This idea leads to the class $\mathcal{OCCI}$ where the positive parts of clauses are "linear" (i.e. every variable occurs only once).

4.12. DEFINITION. $\mathcal{OCCI}$ is the set of all sets of clauses S such that for all $C \in S$:

1. $\mathrm{OCC}(x, C_+) = 1$ for all $x \in V(C_+)$ and

2. $\tau_{max}(x, C_+) \leq \tau_{min}(x, C_-)$ for all $x \in V(C_+) \cap V(C_-)$.

Like in the case of $\mathcal{PVD}$ we can define a class $\mathcal{OCCI}_r$ via renamings and reduce the decidability of $\mathcal{OCCI}_r$ to that of $\mathcal{OCCI}$.

4.13. Theorem. *Hyperresolution decides $\mathcal{OCCI}$, i.e. for every $S \in \mathcal{OCCI}$ the set $R^*_H(S)$ is finite.*

Proof. In [Fermüller et al. 1993]. □

While for deciding $\mathcal{PVD}$ condensing of clauses is not necessary, it is required for $\mathcal{OCCI}$ (otherwise the size of clauses diverges). Of course we can always apply an even stricter (but complete) refinement like hyperresolution with forward subsumption or replacement in order to decide $\mathcal{PVD}$ and $\mathcal{OCCI}$.

$\mathcal{BS}^*$ is not a subclass of $\mathcal{PVD}_r$. But we will define a method to transform $\mathcal{BS}^*$ into $\mathcal{BS}^* \cap \mathcal{PVD}$ under preservation of sat-equivalence. This method is more subtle and more efficient than complete ground saturation. The basic idea is the following:

Let S be in $\mathcal{BS}^*$. Search for a renaming γ such that $\gamma(S) \in \mathcal{PVD}$. If there is such a γ then apply R^*_H to $\gamma(S)$, else select some arbitrary γ and transform $\gamma(S)$ into a set of clauses $\mathcal{D} \in \mathcal{PVD}$ by partial saturation of the variables which violate PVD-1); afterwards apply R^*_H to $\mathcal{D}$. Let us call this procedure BSALG. Then BSALG is indeed a decision algorithm for $\mathcal{BS}^*$. For the actual performance of BSALG the right selection of a renaming is crucial; clearly one should try to select a γ for which the set $\mathcal{D}$ becomes minimal. In the next example we compare brute force saturation with BSALG. For this purpose we replace R^*_H by the more restrictive operator R^*_{Hs} (hyperresolution + forward subsumption). This leads to a further increase of efficiency, but without loss of correctness and termination (note that R^*_{Hs} is complete and $R^*_{Hs}(S) \subseteq R^*_H(S)$ for all sets of clauses S).

4.14. Example. We take the set of clauses from Example 4.8, i.e.

$$S = \{\{P(x_1, x_1, a)\},\ \{P(x, z, u), \neg P(x, y, u), \neg P(y, z, u)\},$$
$$\{P(x, y, u), P(y, z, u), \neg P(x, z, u)\},\ \{\neg P(x, x, b)\}\}.$$

We already know that R^*_H does not terminate on S. Clearly $S \notin \mathcal{PVD}$ but $S \in \mathcal{BS}^*$. We compute the set $\mathcal{D}$ (without renaming the predicate symbol P) and obtain

$$\mathcal{D} = \{\{P(a, a, a)\},\ \{P(b, b, a)\},\ \{P(x, z, u), \neg P(x, y, u), \neg P(y, z, u)\},$$
$$\{P(x, a, u), P(a, z, u), \neg P(x, z, u)\},$$
$$\{P(x, b, u), P(b, z, u), \neg P(x, z, u)\},\ \{\neg P(x, x, b)\}\}$$

$\mathcal{D} \in \mathcal{PVD}$ and $|\mathcal{D}| = |S| + 2 = 6$.

$$R^*_{Hs}(\mathcal{D}) = \mathcal{D} \cup \{\{P(a, b, a), P(b, a, a)\}\}.$$

Thus R^*_{Hs} terminates on $\mathcal{D}$ producing only one additional clause ($|R^*_{Hs}(\mathcal{D})| = 7$). Note that in the second generation of hyperresolvents we obtain the clauses

$$\{P(b,a,a), P(a,a,a), P(a,b,a)\}\ \{P(b,b,a), P(b,a,a), P(a,b,a)\}$$

which are both subsumed by the clauses of the first generation and therefore are deleted.

Using the brute force saturation method we obtain a set of ground clauses $\mathcal{D}'$ which contains 36 clauses. Moreover $\mathcal{D}'$ has still to be tested for satisfiability. Thus we see that BSALG may be much faster than the pure saturation method.

4.2. Model building by hyperresolution

If S is a set of Horn clauses then $P(R^*_H(S))$, the set of all positive clauses derivable from S by hyperresolution, defines a Herbrand model; this is a well-known result in the semantics of logic programming. Indeed, for Horn sets, $P(R^*_H(S))$ consists of unit clauses only which form the atomic representation of a Herbrand model. That means, in the Herbrand base of S, all instances of $P(R^*_H(S))$ are set to true and the other ground atoms to false. $P(R^*_H(S))$ is an atomic representation of an H-model even in the more general case that the nonpositive clauses in S may be arbitrary, but $P(R^*_H(S))$ consists of unit clauses only. By resolution decision theory we possess means to guarantee termination of R^*_H on certain clause classes. If S is a set of Horn clauses and R^*_H terminates on S then (clearly) we obtain a finite atom representation of a Herbrand model of S. Particularly we obtain such finite representations on the classes $\mathcal{PVD} \cap$ Hornlogic and $\mathcal{OCCI} \cap$ Hornlogic by the Theorems 4.11 and 4.13. Our main purpose here is to show, how we can extract atomic representations of Herbrand models from finite sets of clauses which are invariant under the operator of hyperresolution with replacement R_{Hr}; the positive clauses in this set need not be unit clauses. Particularly we will define a procedure which extracts a Herbrand model from every satisfiable set of clauses in $\mathcal{PVD} \cup \mathcal{OCCI}$; this procedure is free of backtracking and does not rely on search. Note that our basic resolution operator, however, will not be R_H (the standard operator of hyperresolution) but the reduction operator R_{Hr}. The choice of R_{Hr} is based on some particular mathematical properties of subsumption-reduced sets which turn out to be fruitful to model building.

For the remaining part of this section we identify unit clauses with atoms and write clauses as disjunctions; this is more natural for automated model building and makes the argumentation more transparent.

Hyperresolution + replacement is a reduction method rather than a deduction method; instead of deleting only clauses which are subsumed by clauses derived before, we subject the whole set of derived clauses to subsumption tests periodically. The formal definition is:

$$R_{Hr}(S) = \mathrm{sub}(R_H(S)), \quad R_H(S) = S \cup \rho_H(S)$$

where sub is a reduction operator which produces subsumption-minimal sets (i.e. there are no different clauses C and D such that $C \leq_{ss} D$) and $\rho_H(S)$ is the set of all hyperresolvents definable from clauses in S. Contrary to R_H the operator R_{Hr} is not monotonic and thus the union of the sets $R^i_{Hr}(S)$ does not describe the set of clauses "eventually" obtained. Note that, by the completeness of R^*_{Hr}, for unsatisfiable sets of clauses S there exists an iteration number i such that $\Box \in R^i_{Hr}(S)$ and therefore $R^i_{Hr}(S) = \{\Box\}$ ($\Box$ subsumes all other clauses). If the sequence of the $R^i_{Hr}(S)$ "converges" on a class of clause sets Γ then R^*_{Hr} is a decision procedure for Γ; in this case we have to compute $R^i_{Hr}(S)$ till we obtain a k such that $R^k_{Hr}(S) = R^{k+1}_{Hr}(S)$. The final set of clauses obtained that way is "stable" (sometimes also called saturated) i.e. it remains unchanged under further reductions (it is in fact in normal form with respect to R^*_{Hr}). We denote it by $R^*_{Hr}(S)$.

4.15. Definition. Let S be a set of condensed clauses. Then S is called (R_{Hr}-) *stable* if $R_{Hr}(S) = S$.

If a R_{Hr}-sequence $(R^i_{Hr}(S))_{i\in\mathbb{N}}$ converges to $R^k_{Hr}(S)$ then, by Definition 4.15, $R^k_{Hr}(S)$ is stable and a fixed point of the operator R_{Hr}. Let us assume that an R_{Hr}-sequence converges and yields a (stable) set S' such that all positive clauses in S' are unit. Then, by the following lemma, these clauses form an atom representation of a Herbrand model of S'; this lemma is a generalization of a well-known theorem in Horn logic. For the remaining part of this section we write "AR" for atomic representation and "stable" for R_{Hr}-stable.

4.16. Lemma. *Let S be a finite set of nonpositive condensed clauses and $\mathcal{A}$ be a finite set of atoms such that $S \cup \mathcal{A}$ is satisfiable and stable. Then $\mathcal{A}$ is an AR of a Herbrand model of $S \cup \mathcal{A}$ (over the signature of $S \cup \mathcal{A}$).*

Lemma 4.16 suggests the following strategy of finding a model: Suppose that a R_{Hr}-sequence converges to S such that $\Box \notin S$ (what is equivalent to $S \neq \{\Box\}$). Then search for a finite set of atoms $\mathcal{A}$ such that $(S - P(S)) \cup \mathcal{A}$ is finite, satisfiable and implies S. The resulting set $\mathcal{A}$ is an AR of a Herbrand model of $(S - P(S)) \cup \mathcal{A}$ which is also a model of S itself. Before we develop a method to obtain such a set of atoms $\mathcal{A}$ we have to investigate where R_{Hr}-reduction actually terminates. $\mathcal{PVD}$ and $\mathcal{OCCI}$ (see Definitions 4.10 and 4.12) are not only decision classes under R_H but also under R_{Hr}; a characteristic feature of both classes is the decomposed form of the positive clauses (and of all derivable positive clauses). The model building procedure to be described below works on all decision classes of hyperresolution with disconnected positive clauses.

4.17. Definition. We call S *positively disconnected* (and write $S \in \mathcal{PDC}$) if $R^*_H(S)$ is finite and all clauses in $P(R^*_H(S))$ are disconnected (see Definition 2.24).

There are technical reasons for using R^*_H instead of R^*_{Hr} in Definition 4.17 [Fermüller and Leitsch 1996].

The application of the operator R^*_{Hr} to $\mathcal{PDC}$ gives us finite R^*_{Hr}-stable sets and provides the raw material for our model building method. For every set $S \in \mathcal{PDC}$ we obtain a set $R^k_{Hr}(S)$ such that $R^k_{Hr}(S)$ is stable. If S is satisfiable then, by the logical equivalence of S and $R^k_{Hr}(S)$, every model of $R^k_{Hr}(S)$ is also a model of S. If the positive clauses in $R^k_{Hr}(S)$ are all unit then, by Lemma 4.16, we already have an AR of a Herbrand model of S.

The following lemma gives us the key technique for the transformation of a stable set into another stable set, where all positive clauses are unit. Essentially we show that, for stable sets of clauses, positive clauses can be replaced by proper subclauses under preservation of satisfiability.

4.18. LEMMA. *Let $S \in \mathcal{PDC}$ such that S is satisfiable and stable and let D be a positive nonunit clause in S. Let P be an (arbitrary) atom in D. Then*
(1) $(S - \{D\}) \cup \{P\}$ is satisfiable and
(2) $(S - \{D\}) \cup \{P\} \rightarrow S$ is valid.
Remark:
(1) and (2) together imply the existence of a model of $(S - \{D\}) \cup \{P\}$ which is also a model of S.

PROOF. (sketch) A detailed proof can be found in [Fermüller and Leitsch 1996].

(2) trivially holds since P implies D.

It remains to prove (1). We use proof by contradiction assuming that the set $S' : (S - \{D\}) \cup \{P\}$ is unsatisfiable. Then $\Box \in R^*_H(S')$; note that we use the monotonic operator R_H instead of R_{Hr}.

Let $D' = D - \{P\}$; then the clauses in $R^*_H(S')$ differ from those in $R^*_H(S)$ "at most by D'". More exactly: if $C \in R^*_H(S')$ then either $C \in R^*_H(S)$ or there exists a variable renaming η with $V(C) \cap V(D'\eta) = \emptyset$ such that the condensed form of $C \cup D'\eta$ is in $R^*_H(S)$. We express this relation by

$$(*) \quad R^*_H(S') \leq_{D'} R^*_H(S).$$

(*) is a nontrivial property and only holds because $S \in \mathcal{PDC}$ and by the condensed form of the clauses.

By $\Box \in R^*_H(S')$ we thus obtain $H \in R^*_H(S)$ where H is the condensed form of D' (note that $\Box \notin S$ as S is satisfiable!). Since $R^*_{Hr}(S)$ subsumes $R^*_H(S)$ and since S is R_{Hr}-stable there must be a clause $G \in S$ s.t. $G \leq_{ss} H$. By definition of D and D' we obtain $D' \leq_{ss} D$ and thus $H \leq_{ss} D$; moreover $H \neq D$—otherwise D is not condensed. Moreover D does not subsume H and thus does not subsume G (note that $\leq_{ss}$ is transitive). Therefore we obtain two clauses D and G in S with $G \leq_{ss} D$ and $D \not\leq_{ss} G$. This, however, contradicts the assumption that S is stable under subsumption. We conclude that S' is satisfiable. □

The validity of Lemma 4.18 is essentially based on the stability of the set of clauses S. It is very easy to see that the result becomes wrong for nonstable sets S: Just take $S = \{\{P(a), P(b)\}, \{\neg P(a)\}\}$.

Trivially S is satisfiable. But if we replace $\{P(a), P(b)\}$ by $\{P(a)\}$ we obtain the set of clauses $S_1 : \{\{P(a)\}, \{\neg P(a)\}\}$ which is unsatisfiable. But note that S is not stable; rather we have $R_{Hr}(S) = \{\{P(b)\}, \{\neg P(a)\}\}$ and the replacement sequence converges to the set $\{\{P(b)\}, \{\neg P(a)\}\}$.
The example also shows that the theorem becomes wrong if we replace the non-monotonic operator R_{Hr} by (the monotonic) R_H; thus we see that subsumption is necessary to guarantee the correctness of the reduction.

The transformation of S into $(S - \{D\}) \cup \{P\}$ can be described by an operator α which (deterministically) selects a clause D and a literal P in D; if $P(S)$ consists of unit clauses only we define $\alpha(S) = S$. Then we may iterate the application of α and R_{Hr}-closure on the new sets of clauses. Note that $(S - \{D\}) \cup \{P\}$ need not be stable, even if S is stable. Therefore we have to compute a reduction sequence on $\alpha(S)$ in order to obtain the next stable set.

Let us assume that the reduction sequence $(R^i_{Hr}(S))_{i\in\mathbb{N}}$ converges to its limit $R^*_{Hr}(S)$. The iterated reduction process can conveniently be defined by an operator on stable sets of clauses.

4.19. Definition *(the operator T).* T is defined on stable sets of clauses in $\mathcal{PDC}$ by $T(S) = R^*_{Hr}(\alpha(S))$.
The iteration of T is defined by:

$T^0(S) = S$ and $T^{i+1}(S) = T(T^i(S))$
if $T^i(S)$ is a stable set in $\mathcal{PDC}$ and $i \in \mathbb{N}$.

It is easy to verify that for all stable sets in $\mathcal{PDC}$ all $T^i(S)$ are again stable sets in $\mathcal{PDC}$. Therefore $T^i(S)$ is well-defined for all stable sets $S \in \mathcal{PDC}$ and $i \in \mathbb{N}$.

Let S be a stable set in $\mathcal{PDC}$. Then, by Lemma 4.18 we know that $\alpha(S) \to S$ is valid and that $\alpha(S)$ is satisfiable, provided S is satisfiable. Thus also $T(S)$ is satisfiable (by the correctness of R^*_{Hr}) and $T(S) \to S$ is valid. Therefore we already know that $T^i(S) \to S$ is valid and that $T^i(S)$ is satisfiable for all $i \in \mathbb{N}$. But, unfortunately, we have not yet reached our goal. We still have to show that the sequence $(T^i(S))_{i\in\mathbb{N}}$ converges, i.e. that there exists a number k such that $T^k(S) = T^{k+1}(S)$; we obtain such a k when all positive clauses in $T^k(S)$ are unit clauses and $\alpha(T^k(S)) = T^k(S)$. In this case $P(T^k(S))$ is an AR of a Herbrand model of S.

In order to prove the convergence of the sequence $(T^i(S))_{i\in\mathbb{N}}$ to a stable set of clauses $\mathcal{D}$ such that $P(\mathcal{D})$ consists of unit clauses only, we have to introduce a Noetherian ordering $\prec$ on sets of clauses and to show that the $T^i(S)$ are decreasing with respect to $\prec$.

4.20. Definition. Let S and $\mathcal{D}$ be two finite sets of condensed clauses. We define $S \prec \mathcal{D}$ if

(1) $S \leq_{ss} \mathcal{D}$,
(2) for all $C \in S$ there exists a $D \in \mathcal{D}$ s.t. $C \leq_{ss} D$ and $|C| \leq |D|$ and
(3) $\mathcal{D} \not\leq_{ss} S$.

It is not hard to show that $\prec$ is irreflexive, transitive and Noetherian; for a proof we refer to [Fermüller and Leitsch 1996]. From $\prec$ we would expect $T(S) \prec S$ for stable sets in $\mathcal{PDC}$; unfortunately this does not hold (a counterexample is defined in [Fermüller and Leitsch 1996]). Instead we have the following weaker property of $\prec$:

4.21. LEMMA. *Let S be a R_{Hr}-stable set in $\mathcal{PDC}$ containing a positive nonunit clause. Then $R^*_H(T(S)) \prec R^*_H(S)$.*

PROOF. (sketch) A detailed proof can be found in [Fermüller and Leitsch 1996].

First of all we reduce the problem to $R^*_H(\alpha(S)) \prec R^*_H(S)$; note that $T(S) = R^*_{Hr}(\alpha(S))$.

Condition (1) of Definition 4.20 is easy to prove: By definition of α we have $\alpha(S) \leq_{ss} S$. Because $\leq_{ss}$ is preserved under R_H we obtain $R^*_H(\alpha(S)) \leq_{ss} R^*_H(S)$.

In order to show condition (3) of Definition 4.20 we have to prove that there exists a $C \in R^*_H(\alpha(S))$ s.t. $R^*_H(S) \not\leq_{ss} \{C\}$. Let E be the nonunit positive clause selected by α. Then there exists an atom P in E with $\alpha(S) = (S - \{E\}) \cup \{P\}$. Then $P \in R^*_H(\alpha(S))$ and $R^*_H(S) \not\leq_{ss} \{P\}$.

Condition (2) of Definition 4.20 follows from the following more general property (which can be proved by induction on n):

(*) For all $n \geq 0$ and for all $C \in R^n_H(\alpha(S))$ there exists a $D \in R^*_H(S)$ s.t. $|C| \leq |D|$ and a renaming substitution η with $C\eta \subseteq D$.

For the proof of (*) it is crucial to have the monotone operator R_H instead of R_{Hr}.

This eventually yields $R^*_H(\alpha(S)) \prec R^*_H(S)$. It remains to show $R^*_H(T(S)) \prec R^*_H(S)$. By definition of T we have

$$R^*_H(T(S)) = R^*_H(R^*_{Hr}(\alpha(S))) \subseteq R^*_H(R^*_H(\alpha(S))) = R^*_H(\alpha(S)).$$

Moreover, by definition of R^*_H and R^*_{Hr} and the subsumption relation for sets of clauses we also obtain

$$T(S) = R^*_{Hr}(\alpha(S)) =_{ss} R^*_H(\alpha(S))$$

and $R^*_H(T(S)) =_{ss} R^*_H(\alpha(S))$. By this last property and by $R^*_H(T(S)) \subseteq R^*_H(\alpha(S))$, $R^*_H(\alpha(S)) \prec R^*_H(S)$ we eventually obtain

$$R^*_H(T(S)) \prec R^*_H(S).$$

□

4.22. EXAMPLE.
Consider the following set of condensed clauses

$$S = \{\{E(a), S(a)\}, \{Q(a), R(a)\}, \{P(x), Q(x), \neg R(x), \neg S(x)\}, \{\neg P(a), \neg Q(a)\}\}.$$

S is in $\mathcal{PVD}$ (and thus in $\mathcal{PDC}$) but S is not stable. We compute the reduction sequence $(R^i_{Hr}(S))_{i\in\mathbb{N}}$ which converges and gives

$$R^*_{Hr}(S) = R^1_{Hr}(S) = S \cup \{\{E(a), P(a), Q(a)\}\}.$$

By writing S_1 for $R^1_{Hr}(S)$ and applying α we obtain

$$\alpha(S_1) = (S_1 - \{\{E(a), S(a)\}\}) \cup \{\{S(a)\}\}.$$

By Lemma 4.18 we know that $\alpha(S_1)$ is satisfiable and that every of its models is a model of S too.
Again $\alpha(S_1)$ is not stable and we compute its corresponding R_{Hr}-reduction sequence.
Then let $S_2 = T(S_1) = R^*_{Hr}(\alpha(S_1))$. So we get

$$S_2 = \{\{S(a)\}, \{Q(a), R(a)\}, \{P(a), Q(a)\}, \{P(x), Q(x), \neg R(x), \neg S(x)\},$$
$$\{\neg P(a), \neg Q(a)\}\}.$$

Note that in the computation of S_2 we obtain the new clash resolvent $\{P(a), Q(a)\}$ which subsumes $\{E(a), P(a), Q(a)\}$. On S_2 we define

$$\alpha(S_2) = (S_2 - \{\{P(a), Q(a)\}\}) \cup \{\{Q(a)\}\}.$$

Then $S_3 = R^*_{Hr}(\alpha(S_2)) = T(S_2) =$

$$\{\{S(a)\}, \{Q(a)\}, \{P(x), Q(x), \neg R(x), \neg S(x)\}, \{\neg P(a), \neg Q(a)\}\}.$$

Clearly $\alpha(S_3) = S_3$ and our procedure stops with $T(S_3) = S_3$ and $S_3 = T^2(S_1)$ (in fact S_3 is a fixed point of T). By the validity of $T^2(S_1) \to S$ and by the satisfiability of $T^2(S_1)$—via the model $\mathcal{M} = \{S(a), Q(a)\}$—we obtain $\mathcal{M}$ as model of S itself.

We are now in the position to formulate our main result, the convergence of the sequence $(T^i(S))_{i\in\mathbb{N}}$ on stable set of clauses S for $S \in \mathcal{PDC}$. From this result we will extract an algorithm which, on satisfiable sets of clauses $S \in \mathcal{PVD}$, always terminates with an atomic representation of a Herbrand model of S (if S is unsatisfiable then $R^*_{Hr}(S) = \{\Box\}$ and the model building procedure does not start at all).

4.23. Theorem. *Let S be a stable and satisfiable set of clauses in $\mathcal{PDC}$. Then the sequence $(T^i(S))_{i\in\mathbb{N}}$ converges to a set of clauses $\mathcal{D}$ such that $P(\mathcal{D})$ is an atomic representation of a Herbrand model of S.*

Proof. We have to prove the existence of an iteration number i with $T^i(S) = T^{i+1}(S)$. From Lemma 4.21 we know that, for every satisfiable, R_{Hr}-stable set $\mathcal{D}$ containing positive nonunit clauses, $R^*_H(T(\mathcal{D})) \prec R^*_H(\mathcal{D})$. Then Lemma 4.18 implies that $\alpha(\mathcal{D})$ is satisfiable and implies $\mathcal{D}$; clearly the same holds for $T(\mathcal{D})$. By iterating this argument we obtain a descending chain of satisfiable sets of clauses

$$\cdots \prec R^*_H(T^{k+1}(S)) \prec R^*_H(T^k(S)) \prec \cdots R^*_H(S).$$

Because $\prec$ is Noetherian this chain is finite and there exists a minimal element $R^*_H(T^i(S))$. We have to show that $T^i(S) = T^{i+1}(S)$:

Let us assume that $T^i(S) \neq T^{i+1}(S)$; then, by definition of T, $\alpha(T^i(S)) \neq T^i(S)$. This, however, implies the existence of a nonunit positive clause in $T^i(S)$; by Lemma 4.21 we obtain $R^*_H(T^{i+1}(S)) \prec R^*_H(T^i(S))$ contradicting the assumption of minimality. This implies $T^i(S) = T^{i+1}(S)$.

We have also seen that $\alpha(T^i(S)) = T^i(S)$, i.e. all positive clauses in $T^i(S)$ are unit; by Lemma 4.16 $P(T^i(S))$ is an AR of a Herbrand model of $T^i(S)$. By the validity of $T^i(S) \supset S$, $P(T^i(S))$ is also an AR of a Herbrand model of S. □

If the sequence $(T^i(S))_{i\in\mathbb{N}}$ converges then we denote the limit by $T^*(S)$. By Theorem 4.23 we always may apply the following algorithm to sets of clauses in $\mathcal{PDC}$:

MB:

a) Compute $R^*_{Hr}(S)$.

b) If $\square \in R^*_{Hr}(S)$ then stop else compute $T^*(R^*_{Hr}(S))$.

MB is correct and complete; that means MB yields $\square$ for unsatisfiable sets of clauses S in $\mathcal{PVD}$ and Herbrand models for satisfiable ones. R^*_{Hr} is complete and always terminates on $\mathcal{PDC}$. Thus if S is satisfiable we obtain a stable set of clauses S' which is in $\mathcal{PDC}$ too. But then, by Theorem 4.23, $T^*(S')$ is defined and $P(T^*(S'))$ is an AR of a Herbrand model of S'. $S' \to S$ is valid and thus $P(T^*(S'))$ is also an AR of a Herbrand model of the set of input clauses S. Note that MB is free of backtracking and search; indeed the computation of T^* is purely "iterative". By $\mathcal{PVD} \cup \mathcal{OCCI} \subseteq \mathcal{PDC}$ the procedure MB works on the decision classes for hyperresolution defined in the beginning of this section. The output of MB is a representation of a (mostly) infinite model $\mathcal{M}$; however one can show that the models $\mathcal{M}$ constructed by MB on $\mathcal{PVD} \cup \mathcal{OCCI}$ can always be transformed into finite models [Fermüller and Leitsch 1996].

The whole model building method can be extended to the classes $\mathcal{PVD}_r$ and $\mathcal{OCCI}_r$ (see Definition 4.10): Given a set of clauses S, search for a sign renaming γ such that $\gamma(S) \in \mathcal{PVD}$ and then apply MB to $\gamma(S)$. MB then yields an AR of a Herbrand model of $\gamma(S)$; by changing the signs backwards one obtains an atom representation of a Herbrand model of S.

4.24. EXAMPLE. We define the following satisfiable set of clauses:

$$S = \{\{P(b)\}, \{P(f(x)), \neg P(x)\}, \{\neg P(a), \neg P(f(a))\}\}.$$

S is not in $\mathcal{PVD}$ and $(R^i_{Hr}(S))_{i\in\mathbb{N}}$ is divergent. Note that for all $i \geq 1$:

$$\{P(f^i(b))\} \in R^i_{Hr}(S) - R^{i-1}_{Hr}(S)$$

. But $S \in \mathcal{PVD}_r$ as can be seen by computing $\gamma(S)$ for the renaming γ exchanging P and $\neg P$. Indeed

$$\{\{\neg P(b)\}, \{P(x), \neg P(f(x))\}, \{P(a), P(f(a))\}\}$$

is in $\mathcal{PVD}$.
By setting $S_1 = \gamma(S)$ we obtain

$$R^0_{Hr}(S_1) = S_1 \text{ and } R^1_{Hr}(S_1) = \{\{\neg P(b)\}, \{P(x), \neg P(f(x))\}, \{P(a)\}\}.$$

Clearly $\rho_H(R^1_{Hr}(S_1)) = \emptyset$ and $R^2_{Hr}(S_1) = R^1_{Hr}(S_1)$.
We see that $(R^i_{Hr}(S))_{i\in\mathbb{N}}$ converges and

$$R^*_{Hr}(S_1) = \{\{\neg P(b)\}, \{P(x), \neg P(f(x))\}, \{P(a)\}\}.$$

By Lemma 4.16 (and clearly visible in this case) $\mathcal{A} : \{P(a)\}$ is an atomic representation of a Herbrand model of S_1 ($\mathcal{A}$ is also a ground representation of this model).

Therefore $\mathcal{M} : \{P(b)\} \cup \{P(f(t)) | t \in H(S)\}$ is a ground representation of a Herbrand model of S. By the principle of forward chaining (S is in Horn form) the set $P(R^*_H(S))$ must be an atomic representation of a Herbrand model of S. Indeed $P(R^*_H(S)) = \{P(f^n(b)) \mid n \in \mathbb{N}\} \subseteq \mathcal{M}$, but as the set is infinite the computation of R^*_H on S does not yield a syntactic model representation. However we can compute a finite AR of the model $\mathcal{M}$ directly out of $\mathcal{A}$ itself; such a representation is $\mathcal{B} : \{P(b), P(f(x))\}$ (over $H(S)$). Note that $\mathcal{M}$ does not represent a minimal Herbrand model, in contrast to $P(R^*_H(S))$.

It is not hard to show that the complement set of a finite ground representation always possesses a finite AR too; this property even holds for linear representations, i.e. for finite AR's $\mathcal{A}$ such that for all $A \in \mathcal{A}$, A contains every variable at most once [Fermüller and Leitsch 1993]. Moreover these representations can be obtained algorithmically.

Models do not only represent counterexamples but are also useful in refining resolution itself. In these model-based refinements only clauses are derived which are false in a predefined model $\mathcal{M}$. In order to automatize such a method we need an algorithm to evaluate clauses over $\mathcal{M}$; for finite models there are straightforward algorithms for this purpose. But the situation becomes more complex for representations of (infinite) Herbrand models. However there are algorithms (again based on hyperresolution) evaluating clauses over AR's constructed by the procedure MB [Fermüller and Leitsch 1996]. So we see that hyperresolution, being an efficient standard refinement in automated deduction, can be used 1. as *decision procedure*, 2. as *model building method* and 3. as *evaluation algorithm* over Herbrand models.

Considering the practical and conceptual importance of constructing counterexamples, the body of knowledge about model generation is relatively small. But in more recent times many different methods and techniques have been invented, analyzed and applied. Automated model building (sometimes also called model generation) is becoming a discipline on its own and one of the most fascinating applications of automated deduction. In this section we presented automated model building as an application of resolution decision theory, but there are several other approaches too. We just mention those of T. Tammet [1991], R. Manthey and F. Bry [1988], S. Klingenbeck [1996], P. Baumgartner and U. Furbach [1996],

P. Baumgartner, U. Furbach, and I. Niemelä [1996], R. Caferra and N. Zabel [1992], N. Peltier [1997*a*], J. Slaney [1992], M. Fujita, J. Slaney, and F. Bennet [1993], and C. Fermüller and A. Leitsch [1996, 1998].

Tammet's approach, like the one presented above, is based on resolution decision procedures. But while Tammet's method directly yields finite models, the method presented in this section produces *symbolic representations* of Herbrand models; finite models then are extracted from Herbrand models in a postprocessing step. Tammet used narrowing and worked with equations (in the object language); his method yields models of formulas from the union of the Ackermann and the monadic class (via the clausal form).

Caferra and Zabel defined an extension (called RAMC) of the resolution calculus by an equational constraint logic, which is specifically designed for the purpose of model building. Their method is symbolic and yields (like that in [Fermüller and Leitsch 1996]) complete representations of Herbrand models. In his Ph.D. thesis Nicolas Peltier gave a thorough discussion and comparison of the different model building methods [Peltier 1997*a*] and developed several new techniques; in particular it is shown that RAMC can simulate hyperresolution extending the range of model building beyond $\mathcal{PDC}$.

Manthey and Bry describe a hyperresolution prover called SATCHMO which is based on a model generation paradigm [Manthey and Bry 1988]. Their method of model building, although similar concerning the use of hyperresolution, differs from that presented in this section in several aspects. They essentially use splitting of positive clauses and backtracking, features that are both avoided in the method presented here. Moreover we make use of subsumption and replacement in an essential way and guarantee termination on specific syntax classes.

The approaches of S. Klingenbeck and of Baumgartner and Furbach are based on tableaux calculi instead on resolution. There countermodels are typically represented by open branches. The hypertableau method of Baumgartner and Furbach can be considered as an improvement of SATCHMO. A very strong tableau method, called RAMCET, has been developed by Nicolas Peltier ([Peltier 1997*b*]); in RAMCET equational constraints are used to prune infinite branches leading to a much better termination behavior than obtained by ordinary tableaux.

In [Fermüller and Leitsch 1998] model building is extended to clause logic with equality. In particular it is shown that $\mathcal{PVD}$ remains decidable if extended by ground equality (it becomes undecidable for full equality). The iteration method, similar to the one defined above, is based on positive resolution and ordered paramodulation. Although the admissible equations are ground the expressivity (for models) is increased considerably.

Slaney [1992] devised the program FINDER that identifies finite models (of reasonable small cardinality) of clause sets whenever they exist. The algorithm is based on a clever variant of exhaustive search through all finite interpretations and does not refer to resolution or another first-order inference system. Fujita, Slaney and Bennett [1993] defined SCOTT, a combination of FINDER with the resolution theorem prover OTTER [McCune 1995]. His method was applied successfully to solve open problems in algebra. In comparison to the other methods mentioned above,

Slaney's method behaves like a "numeric" versus a symbolic one.

Despite the differences all methods aim at more intelligent inference systems which, besides producing proofs efficiently, are also apt to construct counterexamples.

5. Resolution decision procedures for description logics

Two research areas where decidability issues play a particularly prominent role are: extended modal logics and description logics. Although it is not difficult to see that most of the logics under consideration can be translated to first-order logic, it is not obvious what the characteristics of the corresponding classes of first-order formulas are which make these logics decidable. Furthermore, the fact that the class of first-order formulas resulting from the translation of modal formulas or expressions in a description logic is decidable, does not necessarily indicate whether and how a resolution-based decision procedure for this class can be obtained. Following Hustadt and Schmidt [1999*a*, 1999*c*] and Hustadt [1999] we present various classes of clause sets into which nonclassical logics can be embedded by the use of suitable translation morphisms.

We focus on a language called $\mathcal{ALB}$ which is defined as follows. A *(terminological) signature* is given by a tuple $\Sigma = (\mathsf{O}, \mathsf{C}, \mathsf{R})$ of three disjoint alphabets, the set C of *concept symbols*, the set R of role symbols, and the set O of *object symbols*. Concept symbols and role symbols are also called *atomic concepts* and *atomic roles*.

The set of concept terms (or just concepts) and role terms (or just roles) is inductively defined as follows. Every concept symbol is a concept term and every role symbol is a role term. Now assume that C and D are concepts, and R and S are roles. Then

- $\top$ (top concept), $\bot$ (bottom concept), $C \sqcap D$ (concept intersection), $C \sqcup D$ (concept union), $\neg C$ (concept complement), $\forall R.C$ (universal restriction), and $\exists R.C$ (existential restriction) are concept terms,
- $\bigtriangledown$ (top role), $\bigtriangleup$ (bottom role), $R \sqcap S$ (role intersection), $R \sqcup S$ (role union), $\neg R$ (role complement), R^{-1} (role converse), $R\restriction C$ (domain restriction), and $R\downharpoonright C$ (range restriction) are role terms.

Concept term and role terms are (terminological) expressions. We obtain the well-known description logic $\mathcal{ALC}$ [Schmidt-Schauß and Smolka 1991] by restricting $\mathcal{ALB}$ role terms to role symbols.

5.1. Example. Let Cheese and Beef be concept symbols and eats a role symbol. Then $\forall \neg$eats.$\neg$Cheese and $\forall \bigtriangledown$.Cheese are $\mathcal{ALB}$ concepts, but not $\mathcal{ALC}$ concepts. Person $\sqcap \exists$ eats.Beef and Cheese $\sqcup$ Beef are both $\mathcal{ALB}$ concepts and $\mathcal{ALC}$ concepts.

A *knowledge base* has two parts: A *TBox* comprising of terminological sentences and an *ABox* comprising of assertional sentences. *Terminological sentences* are of the form $C \dot{\sqsubseteq} D$, $C \doteq D$, $R \dot{\sqsubseteq} S$, and $R \doteq S$, and *assertional sentences* are of the form $a \in C$ and $(a, b) \in R$, where C and D are concepts, R and S are roles, and a and b are object symbols.

5.2. EXAMPLE. The following is a small $\mathcal{ALB}$ knowledge base.

TBox :	$\text{Person} \dot{\sqsubseteq} \text{Male} \sqcup \text{Female}$	
	$\text{Parent} \doteq \text{Person} \sqcap \exists\, \text{hasChild}.\text{Person}$	
	$\text{Person} \sqcap \text{Professor} \dot{\sqsubseteq} \exists\, \text{hasDegree}.\text{PhD}$	
	$\text{CheeseLover} \doteq \text{Person} \sqcap \forall\, \neg\text{eats}.\neg\text{Cheese}$	
ABox :	$\text{bob} \in \text{Person}$	$\text{jim} \in \text{Person}$
	$\text{bob} \in \text{Professor}$	$\text{cheddar} \in \text{Cheese}$
	$(\text{bob}, \text{jim}) \in \text{hasChild}$	$(\text{bob}, \text{cheddar}) \in \neg\text{eats}$

The knowledge base consists of four terminological sentences and six assertional sentences. Intuitively, the first sentence of the TBox defines that the concept Person is a subset of the union of the concepts Male and Female, that is, every element of the set Person is either an element of the set Male or an element of the set Female. The first sentence of the ABox defines that the object bob is an element of the concept Person. The formal semantics of knowledge bases will be presented in Definition 5.6.

5.3. DEFINITION. A symbol S_0 *uses* a symbol S_1 *directly* in a TBox T if and only if T contains a sentence of the form $S_0 \doteq E$ or $S_0 \dot{\sqsubseteq} E$ such that S_1 occurs in E. A symbol S_0 *uses* S_n if and only if there is a chain of symbols $S_0, \ldots, S_n$ such that S_i uses S_{i+1} directly, for every i, $1 \leq i \leq n-1$. A knowledge base Γ is said to contain a *terminological cycle* if and only if some symbol uses itself in the TBox of Γ.

5.4. EXAMPLE. In the knowledge base of Example 5.2 the concept symbol Parent uses the concept symbol Person directly and Person uses Male and Female directly. So, Parent uses Male and Female. It is straightforward to check that the knowledge base contains no terminological cycles.

The standard definition of knowledge bases imposes the following restrictions on the set of admissible terminological sentences: (i) The left-hand sides of terminological sentences have to be concept symbols or role symbols, (ii) any concept or role symbol occurs at most once on the left-hand side of any terminological sentence, and (iii) there are no terminological cycles. Knowledge bases obeying these restrictions are known as *descriptive knowledge bases*. In this context terminological sentences are called *definitions*. When no restrictions are imposed, we speak of *general knowledge bases*.

5.5. EXAMPLE. The first terminological sentence of the knowledge base above obeys restrictions (ii) and (iii), but it violates restriction (i) of the definition of descriptive knowledge bases. Thus, it is a general knowledge base.

5.6. DEFINITION. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \_^{\mathcal{I}})$ over a signature $\Sigma = (\mathsf{O}, \mathsf{C}, \mathsf{R})$ consists of a nonempty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\_^{\mathcal{I}}$ mapping every

object symbol in O to an element of $\Delta^{\mathcal{I}}$, such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$, every concept symbol A in C to a subset $C^{\mathcal{I}}$ of Δ and every role symbol P in R to a subset of $\Delta \times \Delta$.

The interpretation function is extended to arbitrary concepts and roles as follows:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} & \qquad \nabla^{\mathcal{I}} &= \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset & \qquad \triangle^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} - C^{\mathcal{I}} & \qquad (\neg R)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} - R^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} & \qquad (P \sqcap Q)^{\mathcal{I}} &= P^{\mathcal{I}} \cap Q^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} & \qquad (P \sqcup Q)^{\mathcal{I}} &= P^{\mathcal{I}} \cup Q^{\mathcal{I}} \\
(\forall\, R.C)^{\mathcal{I}} &= \{d \mid (\forall e)\,(d,e) \in R^{\mathcal{I}} \rightarrow e \in C^{\mathcal{I}}\} & \qquad R\lfloor C^{\mathcal{I}} &= R^{\mathcal{I}} \cap (\Delta^{I} \times C) \\
(\exists\, R.C)^{\mathcal{I}} &= \{d \mid (\exists e)\,(d,e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} & \qquad R\rceil C^{\mathcal{I}} &= R^{\mathcal{I}} \cap (C \times \Delta^{I}) \\
& & \qquad R^{-1^{\mathcal{I}}} &= \{(d,e) \mid (e,d) \in R^{\mathcal{I}}\}
\end{aligned}$$

An interpretation $\mathcal{I}$ satisfies the assertional and terminological sentences of $\mathcal{ALB}$

$$\begin{aligned}
a \in C \quad &\text{if} \quad a^{\mathcal{I}} \in C^{\mathcal{I}} & \qquad (a,b) \in R \quad &\text{if} \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}} \\
C \dot{\sqsubseteq} D \quad &\text{if} \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}} & \qquad R \dot{\sqsubseteq} S \quad &\text{if} \quad R^{\mathcal{I}} \subseteq S^{\mathcal{I}} \\
C \doteq D \quad &\text{if} \quad C^{\mathcal{I}} = D^{\mathcal{I}} & \qquad R \doteq S \quad &\text{if} \quad R^{\mathcal{I}} = S^{\mathcal{I}}.
\end{aligned}$$

An interpretation $\mathcal{I}$ is a *model* of a knowledge base Γ if and only if $\mathcal{I}$ satisfies all sentences in Γ.

A terminological expression E_1 is in *negation normal form* if for every subexpression $\neg E_2$ of E_1, E_2 is a concept or role symbol. For every terminological expression E_1 there exists a terminological expression E_2 in negation normal form such that for every interpretation $\mathcal{I}$, $E_1^{\mathcal{I}} = E_2^{\mathcal{I}}$.

It is well known that description logics can be embedded into sublanguages of first-order logic (with equality), Figure 1 specifies one possible embedding of $\mathcal{ALB}$ expressions into first-order logic, the so so-called *standard* or *relational translation*. In Section 5.2 we discuss an alternative embedding, the optimized functional translation.

5.7. Example. The translation of the knowledge base in Example 5.2 is the following first-order formula:

$$\begin{aligned}
&(\forall x)\,(p_{\mathrm{Person}}(x) \rightarrow (p_{\mathrm{Male}}(x) \vee p_{\mathrm{Female}}(x))) \\
\wedge\ &(\forall x)\,(p_{\mathrm{Parent}}(x) \leftrightarrow (p_{\mathrm{Person}}(x) \wedge (\exists y)\,(p_{\mathrm{hasChild}}(x,y) \wedge p_{\mathrm{Person}}(y)))) \\
\wedge\ &(\forall x)\,((p_{\mathrm{Person}}(x) \wedge p_{\mathrm{Professor}}(x)) \rightarrow (\exists y)\,(p_{\mathrm{hasDegree}}(x,y) \wedge p_{\mathrm{PhD}}(y))) \\
\wedge\ &(\forall x)\,(p_{\mathrm{CheeseLover}}(x) \leftrightarrow (p_{\mathrm{Person}}(x) \wedge (\forall y)\,(\neg p_{\mathrm{eats}}(x,y) \rightarrow \neg p_{\mathrm{Cheese}}(y)))) \\
\wedge\ &p_{\mathrm{Person}}(\mathrm{bob}) \wedge p_{\mathrm{Person}}(\mathrm{jim}) \wedge p_{\mathrm{Professor}}(\mathrm{bob}) \wedge p_{\mathrm{Cheese}}(\mathrm{cheddar}) \\
\wedge\ &p_{\mathrm{hasChild}}(\mathrm{bob},\mathrm{jim}) \wedge \neg p_{\mathrm{eats}}(\mathrm{bob},\mathrm{cheddar}).
\end{aligned}$$

Embedding of sentences:

$$\Pi(C \dot{\sqsubseteq} D) = (\forall x)\,(\pi(C,x) \rightarrow \pi(D,x)) \qquad \Pi(R \dot{\sqsubseteq} S) = (\forall x)(\forall y)\,(\pi(R,x,y) \rightarrow \pi(S,x,y))$$
$$\Pi(C \doteq D) = (\forall x)\,(\pi(C,x) \leftrightarrow \pi(D,x)) \qquad \Pi(R \doteq S) = (\forall x)(\forall y)\,(\pi(R,x,y) \leftrightarrow \pi(S,x,y))$$
$$\Pi(a \in C) = \pi(C,\underline{a}) \qquad \Pi((a,b) \in R) = \pi(R,\underline{a},\underline{b})$$

where $\underline{a}$ and $\underline{b}$ are constants uniquely associated with a and b.
Embedding of terms:

$$\pi(A,X) = p_A(X) \qquad \pi(P,X,Y) = p_P(X,Y)$$
$$\pi(\neg C,X) = \neg\pi(C,X) \qquad \pi(\neg R,X,Y) = \neg\pi(R,X,Y)$$
$$\pi(\top,X) = \top \qquad \pi(\nabla,X,Y) = \top$$
$$\pi(\bot,X) = \bot \qquad \pi(\triangle,X,Y) = \bot$$
$$\pi(C \sqcap D,X) = \pi(C,X) \wedge \pi(D,X) \qquad \pi(R \sqcap S,X,Y) = \pi(R,X,Y) \wedge \pi(S,X,Y)$$
$$\pi(C \sqcup D,X) = \pi(C,X) \vee \pi(D,X) \qquad \pi(R \sqcup S,X,Y) = \pi(R,X,Y) \vee \pi(S,X,Y)$$
$$\pi(\forall R.C,X) = (\forall y)\,(\pi(R,X,y) \rightarrow \pi(C,y)) \qquad \pi(R\lfloor C,X,Y) = \pi(R,X,Y) \wedge \pi(C,Y)$$
$$\pi(\exists R.C,X) = (\exists y)\,(\pi(R,X,y) \wedge \pi(C,y)) \qquad \pi(R\lceil C,X,Y) = \pi(R,X,Y) \wedge \pi(C,X)$$
$$\pi(R^{-1},X,Y) = \pi(R,Y,X)$$

where X and Y are meta-variables for variables and constants, and p_A (respectively p_P) denotes a unary (binary) predicate symbol uniquely associated with the concept symbol A (role symbol P).

Figure 1: The standard embedding of $\mathcal{ALCB}$ into first-order logic

In the case of $\mathcal{ALCB}$ all common inferential services for knowledge bases, like subsumption tests for concepts, TBox classification, realization, retrieval, can be reduced to tests of the satisfiability of a knowledge base.

5.1. Decidability by ordered resolution

The conversion to clausal form of first-order formulas resulting from the translation of $\mathcal{ALCB}$ knowledge bases, makes use of the following structural transformation. For ease of presentation we assume any first-order formula ϕ is in negation normal form.

Recall the notion of a definitional form of a first-order formula from Section 2. Note that the embedding Π preserves the structure of $\mathcal{ALCB}$ expression, that is, with every occurrence of an $\mathcal{ALCB}$ expression in a knowledge base Γ we can uniquely associate an occurrence in the first-order formulae $\Pi(\Gamma)$. Let Λ^r be the set of positions of nonatomic concepts and nonatomic roles in the knowledge base Γ. By Ξ_r we denote the transformation taking $\Pi(\Gamma)$ to its *definitional form* $\mathrm{Def}_{\Lambda^r}(\Pi(\Gamma))$.We assume that the variable ordering in a literal $Q_\lambda(x,y)$ introduced by Ξ_r follows the convention we have used in the definition of π, that is, for a subformulae like $R(x,y) \star S(x,y)$ associated with $R \star S$ and subformulae like $R(y,x)$ associated with

R^{-1} we introduce $Q_\lambda(x, y)$ (not $Q_\lambda(y, x)$).

5.8. Theorem. *Let Γ be any $\mathcal{ALB}$ knowledge base. $\Xi_r\Pi(\Gamma)$ can be computed in polynomial time, and Γ is satisfiable iff $\Xi_r\Pi(\Gamma)$ is satisfiable.*

Proof. It is straightforward to see that the translation of Γ and the computation of the definitional form can be performed in polynomial time. That Π is satisfiability equivalence preserving can be checked with respect to the semantics given in Definition 5.6. By Proposition 1 Ξ_r is also satisfiability equivalence preserving. □

Next we characterise a class of clauses which we call *DL-clauses*. Let C be a clause and L a literal in C. We refer to a literal L as *embracing* in C if for every L' in C, $V(L') \cap V(L) \neq \emptyset$ implies $V(L') \subseteq V(L)$ (that is, it contains all variables occurring in the split component in which it occurs). A term t in C is called *embracing* if for every L' in C, $V(L') \cap V(t) \neq \emptyset$ implies $V(L) \subseteq V(t)$. A literal L is *singular* if it contains no functional term and $V(L)$ is a singleton. A literal is *flat* if it is nonground and contains no functional term.

5.9. Definition *(DL-clause).* A literal L is a *DL-literal* if the following is true.

1. L is simple (see Definition 3.36),
2. L is either monadic or dyadic, and contains at most 2 variables,
3. L is ground whenever it contains a constant symbol, and
4. the maximal arity of any function symbol in L is 1.

A clause C is a *DL-clause*, if

1. when C contains a functional term t, then t is embracing,
2. C is ground whenever C contains a constant symbol,
3. all literals in C are DL-literals, and
4. the argument multisets of all flat, dyadic literals coincide.

Property (4) is important to enable us to use a liftable ordering for our resolution decision procedure. It excludes clauses like $\{p(x, x), q(x, y)\}$. In order to avoid possibly unbounded chains of variables across literals we would need to restrict resolution inferences to the literal $q(x, y)$. But, when such clauses are present it is in general not possible to devise a liftable ordering such that only the second literal is maximal. By contrast, clauses like $\{p(x, x), q(x, x)\}$ and $\{p(x, y), q(x, y)\}$ are DL-clauses, and may occur in a derivation from $\Xi_r\Pi(\Gamma)$.

5.10. Lemma. *Let Γ be an $\mathcal{ALB}$ knowledge base. Every clause in the standard clausal form of $\Xi_r\Pi(\Gamma)$ belongs to the class of DL-clauses.*

We define an A-ordering $<_{\mathcal{AOV}}$ on atoms as follows.

5.11. Definition. Let A and B be two atoms, then $A <_{\mathcal{AOV}} B$ if either the multiset of arguments of A is a strict sub-multiset of the multiset of arguments of B or for every argument s of A there exists an argument t of B such that s is a strict subterm of t.

Hustadt and Schmidt [1999*a*] show that ordered resolution and ordered factoring on DL-clauses with respect to $<_{\mathit{ocv}}$ will result in DL-clauses.

5.12. LEMMA. *Let $C = \{L_1, L_2\} \cup D$ be an indecomposable, DL-clause with σ a most general unifier of L_1 and L_2. The split components of $(\{L_1\} \cup D)\sigma$ are DL-clauses.*

PROOF. Since C is indecomposable and contains at least two literals, it is not a ground clause. By property (2) it contains no constants. So, the most general unifier σ will not introduce any constant symbols. Also, there are no functional terms in the range of σ, since all functional terms in C are embracing. Therefore, σ is a variable renaming. It is straightforward to see that variable renamings preserve the properties (1)–(4). □

5.13. LEMMA. *DL-clauses are preserved under $<_{\mathit{ocv}}$-ordered resolution; more precisely: Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be two variable-disjoint, indecomposable, DL-clauses such that A_1 and A_2 are unifiable with most general unifier σ, and there is no literal $A_3\sigma$ in $(D_1 \cup D_2)\sigma$ such that $A_1\sigma <_{\mathit{ocv}} A_3\sigma$. Then the split components of $(D_1 \cup D_2)\sigma$ are DL-clauses.*

PROOF. It is not difficult to show that the following holds:

1. A_1 is embracing in C_1.
2. If A_1 is a flat, singular literal, then C_1 contains only flat, singular literals.
3. If A_1 is flat, then C_1 contains no functional term.
4. If $A_1\sigma$ contains a functional term t, then for no variable x in C_2 is $x\sigma$ a functional term, and t is embracing in $(D_1 \cup D_2)\sigma$.
5. If A_1 contains a functional term t, then for no variable x in C_1, is $x\sigma$ a functional term and t is embracing in $(D_1 \cup D_2)\sigma$.

Analogously, for $\neg A_2$ and C_2. Let $(D_1 \cup D_2)\sigma$ be nonempty and E be a split component of it.

1. It follows that no functional term in E has a functional term argument and that functional terms in E are embracing.
2. Suppose $A_1\sigma$ contains a constant. Then either A_1 or A_2 is ground. Since A_1 and $\neg A_2$ are embracing, E is a unit ground clause.
3. It follows from (1) and (2) that all literals in E are DL-literals.
4. Only if $A_1\sigma = A_2\sigma$ is a flat, dyadic literal does E possibly contain a flat, dyadic literal $L\sigma$. The argument multiset of L coincides with that of either A_1 or A_2. Therefore, the argument multiset of $L\sigma$ coincides with $A_1\sigma$. So does the argument multiset of any flat, dyadic literal in E.

□

5.14. THEOREM. *Let Γ be an $\mathcal{ALB}$ knowledge base and let S be the standard clausal form of $\Xi_r\Pi(\Gamma)$. Then any derivation from S by a combination of $R_{<_{\mathit{ocv}}}$ and the splitting or condensation rule terminates.*

Proof. By Theorem 5.8 and Lemmas 5.10, 5.12 and 5.13; because any set of nonvariant, indecomposable DL-clauses built from finitely many predicate and function symbols is finitely bounded; and the fact that any application $R_{<\infty\nu}$ is followed immediately by applications of the splitting rule. □

5.15. Corollary. *The satisfiability problem for $\mathcal{ALB}$ knowledge bases is decidable; the transformation $\Xi_r\Pi$ and $R_{<\infty\nu}$ combined with the splitting or condensation rule provide a decision procedure.*

Given an $\mathcal{ALB}$ knowledge base Γ, the characteristics of clauses in the standard clausal form of $\Xi_r\Pi(\Gamma)$ are rather simple. It is therefore not surprising that there exists a variety of alternative solvable classes into which we can embed satisfiability problem for $\mathcal{ALB}$ knowledge bases.

Tammet [1992, 1995] has introduced the following fragment of first-order logic:

5.16. Definition. Any formula φ of the predicate logic without equality and without nonconstant function symbols belongs to the class One-free if any subformula of φ starting with a quantifier contains no more than one free variable.

It is straightforward to see that for any knowledge base Γ, $\Pi(\Gamma)$ is in the class One-free. Furthermore, Ξ_r transforms any formula φ in One-free into a conjunction of formulas of the class $\mathcal{K}$. The standard clausal form of $\Xi_r(\varphi)$ is an element of $\mathcal{KC}$. Therefore, the refinement $R_{<_z}$ defined in Section 3.6 combined with the splitting rule provides a decision procedure for the class One-free as well as for the satisfiability problem of $\mathcal{ALB}$ knowledge bases.

5.17. Theorem. *The class One-free is decidable; $R_{<_z}$ combined with the splitting rule provides a decision procedure.*

5.18. Corollary. *The transformation $\Xi_r\Pi$ and $R_{<_z}$ combined with the splitting rule provides a decision procedure for the satisfiability problem of $\mathcal{ALB}$ knowledge bases.*

5.2. Variations of $\mathcal{ALB}$

A closer inspection of the formulas we obtain from translating $\mathcal{ALB}$ knowledge bases into first-order logic reveals, that only the presence of the top role ∇ and role complement prevents $\Pi(\Gamma)$ from being guarded for arbitrary knowledge bases Γ.

5.19. Example. Consider the $\mathcal{ALB}$ expressions $\forall\,\nabla.A$ and $\forall\,\neg P.A$. The translation of these expression is $(\forall y)\,(\top \to A(y))$ and $(\forall y)\,(\neg P(x,y) \to A(y))$, respectively. In both cases the first-order formulas lack an appropriate guard to satisfy the definition of guarded formulas.

Consequently, the reduct of $\mathcal{ALB}$ without the top role and role complement allows for the use of the resolution decision procedures for the guarded fragment.

5.20. Corollary. *$R_{<_{\mathcal{GF}}}$ provides a decision procedure for the satisfiability problem of knowledge bases over the reduct of $\mathcal{ALB}$ without the top role and role complement.*

If we restrict ourselves even further, namely to descriptive knowledge bases over $\mathcal{ALC}$, then all the inferential services for knowledge bases can be reduced to tests of the satisfiability of concept terms [Donini et al. 1994]. See also [Calvanese et al. 2001] (Chapter 23 of this Handbook) for a discussion of inferential services in description logics and their interrelationships. Since $\mathcal{ALC}$ is a notational variant of basic multi-modal logic, the techniques described in [Ohlbach et al. 2001] (Chapter 21 of this Handbook) are applicable. In particular, the optimized functional translation [Schmidt 1997, Schmidt 1998] can be used instead of the standard translation described in Section 5 to map concept terms into first-order logic.

The optimized functional translation maps concept terms of $\mathcal{ALC}$ into a logic, called *basic path logic*, which is a monadic fragment of sorted first-order logic with sorts AF_R uniquely associated with the role symbols and an additional sort W, binary function symbols $[\_\_]_{AF_R}$ of sort $W \times AF_R \to W$, and one constant ϵ of sort W. For better readability we write $[\_\_]$ instead of $[\_\_]_{AF_R}$, since the sort of $[\_\_]_{AF_R}$ is uniquely determined by the sort of its second argument.

5.21. Example. Let AF_{R_1} and AF_{R_2} be sorts associated with the role symbols R_1 and R_2, respectively. Let the constant symbol c be of sort AF_{R_1} and the variable x of sort AF_{R_2}. Then $[[\epsilon\, c]\, x]$ is a term of sort W.

The optimized functional translation consists of a sequence of transformations. The first transformation Π_f maps a concept term C to its so-called functional translation defined by $(\forall x)\, \pi_f(\varphi, x)$, where π_f is given by

$$\begin{aligned}
\pi_f(A, s) &= A(s)\\
\pi_f(\neg C, s) &= \neg \pi_f(C, s)\\
\pi_f(C \sqcap D, s) &= \pi_f(C, s) \wedge \pi_f(D, s)\\
\pi_f(C \sqcup D, s) &= \pi_f(C, s) \vee \pi_f(D, s)\\
\pi_f(\forall\, R.C, s) &= def_R(s) \to (\forall x{:}AF_R)\, \pi_f(C, [s\, x])\\
\pi_f(\exists\, R.C, s) &= def_R(s) \wedge (\exists x{:}AF_R)\, \pi_f(C, [s\, x]).
\end{aligned}$$

The second transformation applies the so-called quantifier exchange operator Υ, which moves existential quantifiers inwards over universal quantifiers using the rule '$(\exists x)(\forall y)\, \psi$ becomes $(\forall y)(\exists x)\, \psi$'. The two transformation are combined into a the function $\overline{\Pi}_f$ mapping concept terms C to $\neg\Upsilon\Pi_f(\neg C)$.

The mapping $\overline{\Pi}_f$ is satisfiability equivalence preserving for $\mathcal{ALC}$, that is, a concept term C is satisfiable if and only if $\neg\Upsilon\Pi_f(\neg C)$ is satisfiable [Ohlbach and Schmidt 1997]. Note that in the closed first-order formula $\overline{\Pi}_f(C)$ no existential quantifier occurs in the scope of a universal quantifier. Thus, all skolem functions in the standard clausal form of $\overline{\Pi}_f(C)$ are constants. The following theorem is a consequence of the results of Ohlbach and Schmidt [1997] for basic path logic.

5.22. Theorem. *Let C be an $\mathcal{ALC}$ concept term. Then any derivation by unrefined resolution combined with the splitting or condensation rule from $\overline{\Pi}_f(C)$ terminates.*

5.23. Corollary. *The mapping $\overline{\Pi}_f$ and unrefined resolution combined with the splitting or condensation rule provide a decision procedure for the satisfiability problem of $\mathcal{ALC}$ concept terms.*

By additional transformations we can embed the basic path logic into a subclass of the Bernays-Schönfinkel class. First, we replace all occurrences of literals $P(s)$ where s is a path of the form $[[[\epsilon\,\alpha^1_{R_1}]\,\alpha^2_{R_2}]\ldots\alpha^n_{R_n}]$ where $\alpha^j_{R_j}$ is a variable or constant of sort AF_{R_j}, for $1 \leq j \leq n$, by $P_{n+1}(\epsilon, \alpha^1_{R_1}, \ldots, \alpha^n_{R_n})$ where P_{n+1} is an $(n{+}1)$-ary predicate symbol uniquely associated with P and n. Second, the sort information associated with the variables and constants occurring in the literals in the clause set can be encoded in the predicate symbols of the literals. So, we can replace all occurrences of literals $P_{n+1}(\epsilon, \alpha^1_{R_1}, \ldots, \alpha^n_{R_n})$ by $P_{R_1\ldots R_n}(\epsilon, \alpha^1, \ldots, \alpha^n)$ where $P_{R_1\ldots R_n}$ is a predicate symbol uniquely associated with the predicate symbol P_{n+1} and the sorts AF_{R_1}, …, AF_{R_n}. The variables and constants α^1, …, α^n no longer carry any sort information. Finally, we observe that all literals in the transformed clause set share the first argument ϵ, which we can eliminate safely.

This sequence of three transformations can be combined in one:

$$P([[[\epsilon\,\alpha^1_{R_1}]\,\alpha^2_{R_2}]\ldots\alpha^n_{R_n}]) \quad \text{becomes} \quad P_{R_1\ldots R_n}(\alpha^1, \ldots, \alpha^n).$$

We denote this transformation by $\Xi_{\mathcal{BS}}$.

5.24. Theorem. *For every $\mathcal{ALC}$ concept term C, the first-order formula $\Xi_{\mathcal{BS}}\overline{\Pi}_f(C)$ is satisfiable if and only if C is satisfiable and $\Xi_{\mathcal{BS}}\overline{\Pi}_f(C)$ is an element of the Bernays-Schönfinkel class.*

Note that not every formula of the Bernays-Schönfinkel class is in the range of $\Xi_{\mathcal{BS}}\overline{\Pi}_f$. For the particular subclass of $\mathcal{BS}$ we obtain from $\Xi_{\mathcal{BS}}\overline{\Pi}_f$ that Theorem 5.22 is still valid. Since the number of ground instances of formulas in the Bernays-Schönfinkel class is finite it is also possible to use propositional theorem proving methods to obtain decision procedures for the satisfiability problem of $\mathcal{ALC}$ concept terms.

5.3. Decidability by hyperresolution

Following [Hustadt and Schmidt 1999*a*] we define a decision procedure based on hyperresolution combined with the splitting rule for the satisfiability problem of $\mathcal{ALC}$ concept terms. The derivations are in essence exactly as for tableaux calculi for description logics. However, compared to tableaux calculi the procedure has the advantage that (i) it provides more flexibility concerning the theorem proving strategy, and (ii) it allows for the application of general redundancy criteria.

Let C be an $\mathcal{ALC}$ concept term. Let the clause set S be the standard clausal form of $\Xi_r\Pi(C)$. There is exactly one positive ground unit clause in S and all other clauses

contain negative literals. The clauses containing negative literals alone cannot form a clash sequence.

In all clauses except those of the form

$$\{\neg p_A(x), \neg p_R(x,y), p_B(y)\} \tag{5.1}$$

there is a unary negative literal containing all variables of the clause, and with the exception of

$$\{\neg p_A(x), p_R(x, f(x))\} \tag{5.2}$$

and

$$\{\neg p_A(x), p_B(f(x))\} \tag{5.3}$$

no variables occur as arguments of functional terms.

Given these observations it is straightforward to prove by induction that all hyperresolvents derivable from the standard clausal form of $\Xi_r\Pi(C)$ are ground clauses. With eager applications of the splitting rule to hyperresolvents we can ensure that all positive clauses in a derivation are ground unit clauses. However, in inference steps by hyperresolution involving clauses of the forms (5.2) and (5.3) as nonpositive premises the term depth of the hyperresolvent is greater than the term depth of the positive premise of the inference. So, to ensure termination of hyperresolution we have to show that there is an upper bound on the term depth of hyperresolvents.

Recall, that the standard translation Π into first-order logic preserves the structure of concepts and roles and that the application of the transformation Ξ_r to a formula φ establishes a correspondence between subformula of φ and the new predicate symbols introduced by Ξ_r. Consequently, the subexpression relation on the concept term C induces an acyclic relation on the predicate symbols in $\Xi_r\Pi(C)$. Formally, define a dependency relation $\succ_c^1$ on the predicate symbols in $\Xi_r\Pi(C)$ by $p_A \succ_c^1 p_B$, if there is a definition $(\forall \bar{x})\,(\phi \rightarrow \psi)$ in $\Xi_r\Pi(\overline{C})$ such that p_A occurs in ϕ and p_B occurs in ψ. Let $\succ_{\mathsf{S}}$ be an ordering on the predicate symbols in $\Xi_r\Pi(\overline{\Gamma})$ which is compatible with the transitive closure of $\succ_c^1$. Since the subexpression relation on C is acyclic, it is always possible to find such an ordering. We will use the ordering $\succ_{\mathsf{S}}$ to show the termination of hyperresolution on the standard clausal form of $\Xi_r\Pi(C)$.

Next, define a measure μ_S on ground unit clauses occurring in a derivation from the clausal form S of $\Xi_r\Pi(C)$ by

$$\mu_D(C) = \begin{cases} (p_A, p_A), \text{ if } C = \{p_A(t)\} \\ (p_A, p_R), \text{ if } C = \{p_R(s,t)\} \text{ has been derived from} \\ \qquad\qquad \{\neg p_A(x), p_R(x, f(x))\} \in S \end{cases}$$

That is, the measure associated with a ground unit clause is a pair of predicate symbols. Measures are compared by the lexicographic combination $\succ_{\mathsf{S}}^2 = (\succ_{\mathsf{S}}, \succ_{\mathsf{S}})$. Since $\succ_{\mathsf{S}}$ is well-founded, also $\succ_{\mathsf{S}}^2$ is well-founded.

Note that any clause of the form $\{p_R(s,t)\}$ occurring in a derivation from S is derived by resolving a unit clause $\{p_A(s)\}$ with a clause of the form (5.2). Thus,

μ_S is well-defined, i.e. it assigns a pair of predicate symbols to every ground unit clause occurring in a derivation from S.

It is straightforward to check that for any inference step by hyperresolution on S the conclusion of the inference step is smaller or equal to the measure of one of its positive premises. Termination then follows from the well-foundedness of $\succ^2_S$.

Note that we can use μ_S and $\succ^2_S$ to define an atom complexity measure as described on page 1819. Then the following theorem is a consequence of the results in [Leitsch 1993].

5.25. Theorem. *Let C be an $\mathcal{ALC}$ concept term and let S be the standard clausal form of $\Xi_r\Pi(C)$. Then any derivation from S by hyperresolution combined with the splitting rule terminates.*

5.26. Corollary. *The transformation $\Xi_r\Pi$ and hyperresolution combined with the splitting rule provide a decision procedure for the satisfiability problem of $\mathcal{ALC}$ concept terms.*

Hustadt and Schmidt [1999*c*] extend the approach presented in this section to general knowledge bases over the description logic $\mathcal{ALC}$.

5.4. Simulation of tableaux decision procedures for $\mathcal{ALC}$

Hustadt and Schmidt [1999*a*] also consider the relation between the decision procedure based on hyperresolution and a standard tableaux decision procedure for the satisfiability problem of $\mathcal{ALC}$ concept terms.

Given an $\mathcal{ALC}$ concept term C we start with an initial constraint system $\{a \in C\}$ for an arbitrary object symbol a and apply the following transformation rules:

1. $\Delta \Rightarrow_{\sqcap} \Delta \cup \{a \in C, a \in D\}$, if $a \in (C \sqcap D)$ is in Δ, $a \in C$ and $a \in D$ are not both in Δ.
2. $\Delta \Rightarrow_{\sqcup} \Delta \cup \{a \in E\}$, if $a \in (C \sqcup D)$ is in Δ, neither $a \in C$ nor $a \in D$ is in Δ, and $E = C$ or $E = D$.
3. $\Delta \Rightarrow_{\exists} \Delta \cup \{(a, b) \in R, b \in C\}$, if $a \in \exists R.C$ is in Δ, there is no c such that both $(a, c) \in R$ and $c \in C$ are in Δ, and b is a new object symbol with respect to Δ.
4. $\Delta \Rightarrow_{\forall} \Delta \cup \{b \in C\}$, if $a \in \forall R.C$ and $(a, b) \in R$ are in Δ, and $b \in C$ is not in Δ.
5. $\Delta \Rightarrow_{\bot} \Delta \cup \{a \in \bot\}$, if $a \in A$ and $a \in \neg A$ are in Δ, where A is a concept symbol.

Let $\Rightarrow_{\mathcal{TAB}}$ be the transitive closure of the union of the transformation rules given above. A constraint system Δ contains a *clash* if $\{a \in \bot\} \subset \Delta$. A constraint system Δ is satisfiable iff there exists a constraint system Δ' such that (i) $\Delta \Rightarrow_{\mathcal{TAB}} \Delta'$, (ii) no further applications of $\Rightarrow_{\mathcal{TAB}}$ to Δ' are possible, and (iii) Δ' is clash-free. A concept term C is satisfiable if and only if the constraint system $\{a \in C\}$ is satisfiable.

The correspondence between the tableaux decision procedure and the decision procedure based on hyperresolution presented in Section 5.3 is not difficult to see.

Remember that for every concept C and every role R, which may possibly occur in an ABox during a satisfiability test, there exist corresponding predicate symbols p_C and p_R in the standard clausal form of $\Xi_r\Pi(\overline{\Gamma})$. Likewise for every object symbol a we will have a corresponding term t_a.

Transformation by the inference rules is simulated as follows:

1. An application of the $\Rightarrow_\sqcap$ rule corresponds to hyperresolution inference steps between a ground clause $\{p_{C\sqcap D}(t_a)\}$ and clauses $\{\neg p_{C\sqcap D}(x), p_C(x)\}$ and $\{\neg p_{C\sqcap D}(x), p_D(x)\}$, generating the hyperresolvents $\{p_C(t_a)\}$ and $\{p_D(t_a)\}$.
2. An application of the $\Rightarrow_\sqcup$ rule corresponds to an inference step between a ground unit clause $\{p_{C\sqcup D}(t_a)\}$ and $\{\neg p_{C\sqcup D}(x), p_C(x), p_D(x)\}$. We then apply the splitting rule to the conclusion $\{p_C(t_a), p_D(t_a)\}$ which will generate two branches, one on which our set of clauses contains $\{p_C(t_a)\}$ and one on which it contains $\{p_D(t_a)\}$.
3. An application of the $\Rightarrow_\exists$ rule corresponds to two hyperresolution inference steps between the clauses $\{p_{\exists R.C}(t_a)\}$, $\{\neg p_{\exists R.C}(x), p_R(x, f(x))\}$, and $\{\neg p_{\exists R.C}(x), p_C(f(x))\}$. This will add $\{p_R(t_a, f(t_a))\}$ and $\{p_C(f(t_a))\}$ to the clause set. The term $f(t_a)$ corresponds to the new object symbol b introduced by the $\Rightarrow_\exists$ rule, that is, $t_b = f(t_a)$.
4. An application of the $\Rightarrow_\forall$ rule corresponds to a hyperresolution inference step with nonpositive premise $\{\neg p_{\forall R.C}(x), \neg p_R(x, y), p_C(y)\}$ and positive premises $\{p_{\forall R.C}(t_a)\}$ and $\{p_R(t_a, t_b)\}$.
5. An application of the $\Rightarrow_\bot$ rule corresponds to a hyperresolution inference step with nonpositive premise $\{\neg p_{\neg A}(x), \neg p_A(x)\}$ and positive premises $\{p_A(t_a)\}$ and $\{p_{\neg A}(t_a)\}$. The hyperresolvent is the empty clause showing that the current clause set is unsatisfiable.

5.27. Theorem. *The hyperresolution decision procedure p-simulates tableaux decision procedures for $\mathcal{ALC}$.*

5.5. Model generation

As with tableaux-based procedures and the approach presented in Section 4.2 hyperresolution lends itself for the construction of a model when the empty clause was not derived. We briefly describe how this can be done.

First, define a translation mapping which maps the first-order syntax back to the original syntax. This exploits the one to one correspondence between ground terms and objects, and predicate symbols and concept and role subexpressions, respectively. For any ground term t_a, let $\widehat{t_a}$ denote the object symbol a uniquely associated with t_a. Let $\widehat{C}$ and $\widehat{R}$ denote the concept and role obtained by replacing any occurrences of $\overline{I}$ by $\neg A$, and P^u and P^d by P, respectively. Now, define the mapping Π^{-1} by $\Pi^{-1}(\{p_C(t_a)\}) = \widehat{t_a} \in \widehat{C}$, $\Pi^{-1}(\{p_R(t_a, t_b)\}) = (\widehat{t_a}, \widehat{t_b}) \in \widehat{R}$ and the straightforward extension to clauses and sets of clauses. Given a set S of unit ground clauses, $\Pi^{-1}(S)$ is a set of assertional sentences.

Second, we denote by $\widehat{\mathcal{I}}$ the function mapping a set of assertional sentences to an interpretation $(\Delta^{\mathcal{I}}, \_^{\mathcal{I}})$ in the sense of Definition 5.6 which is defined as follows. Given a set Γ of assertional sentences, the domain $\Delta^{\mathcal{I}}$ of $\widehat{\mathcal{I}}(\Gamma)$ is the set of all object symbols in Γ. For every concept symbol A, an element a of $\Delta^{\mathcal{I}}$ is in $A^{\mathcal{I}}$ iff the assertional sentence $a \in A$ is an element of Γ. Correspondingly, for every role symbol P, a pair (a, b) of elements of $\Delta^{\mathcal{I}}$ is in $P^{\mathcal{I}}$ iff the assertional sentence $(a, b) \in P$ is an element of Γ.

Recall from Section 4.2 the notion of an atomic representation of a Herbrand model of a set S of clauses.

5.28. Theorem. *Let Γ be a descriptive knowledge base, S the standard clausal form of $\Xi_r\Pi(\overline{\Gamma})$, and let $\mathcal{A}$ be $P(R^*_H(S))$. Then $\mathcal{A}$ is a finite, ground AR of S, and $\widehat{\mathcal{I}}(\Pi^{-1}(\mathcal{A}))$ is a model of Γ.*

Proof. As noted before, during the derivation only ground unit clauses are generated. To prove that $\mathcal{A}$ is a model of $R^*_H(S)$ we have to show that any ground instance of a clause C in $R^*_H(S)$ is true in $\mathcal{A}$. This obviously holds for any of the positive ground unit clauses in $R^*_H(S)$. Also, any negative ground unit clause $\{\neg A\}$ is true in $\mathcal{A}$. Let $C = \{\neg A_1\sigma, \ldots, \neg A_n\sigma\} \cup D\sigma$, where $D\sigma$ contains no negative literals, be the ground instance of clause in $R^*_H(S)$. If one of the $A_i\sigma$, $1 \leq i \leq n$ is not in $\mathcal{A}$, then C is true. Otherwise, $R^*_H(S)$ contains the unit clauses $\{A_i\sigma\}$, $1 \leq i \leq n$, and we have derived $D\sigma$ at one stage of the derivation. Consequently, one of the split components of $D\sigma$ is in $R^*_H(S)$. The split components of a ground clause are unit ground clauses, for which we have already shown that they are true in $\mathcal{A}$. It follows that C is true in $\mathcal{A}$. Hence $\mathcal{A}$ is a model of $R^*_H(S)$ and also S.

By Theorem 5.27 and the correspondence between predicate symbols and ground terms in the clause set and the symbols in the knowledge base, it follows that $\Pi^{-1}(\mathcal{A})$ is identical to a clash-free knowledge base Γ' derivable from Γ such that no further applications of $\Rightarrow_{\mathcal{TAB}}$ are possible. It follows from the results of [Schmidt-Schauß and Smolka 1991] that $\widehat{\mathcal{I}}(\Pi^{-1}(\mathcal{A}))$ is a model of Γ. □

The finite model property is an immediate consequence of Theorems 5.14 and 5.28.

5.29. Corollary. *Let Γ be a descriptive knowledge base. If Γ is satisfiable, then it has a model of finite size.*

6. Related work

There are several extensions and applications of resolution decision procedures not described in the sections above which deserve attention. We just mention two types:

- extensions to clause logic with equality and
- extensions to constraint logic.

Almost all decision classes discussed above become undecidable if the equality predicate is allowed to occur. In particular this holds for the classes $\mathcal{E}_1$, $\mathcal{E}^+$, $\mathcal{S}^+$ and $\mathcal{PVD}$ (but not for the Bernays-Schönfinkel class). Important decidable classes remaining decidable under addition of equality are the (initially extended) Ackermann class and the monadic class. A decision procedure for an extension of the Ackermann class with equality is defined in [Fermüller and Salzer 1993]; it is based on ordered resolution and ordered paramodulation. A modification of the superposition calculus was applied to decide the monadic class with equality [Bachmair, Ganzinger and Waldmann 1993]. Both methods use extension by new constant symbols. In [Fermüller and Salzer 1993] new constant symbols (in equations) are introduced in a preprocessing step in order to guarantee termination; in [Bachmair et al. 1993] the extension by constants is part of the inference rule. This indicates that "ordinary" equational inference systems are not strong enough to handle more complicated equational decision problems. Nevertheless both methods mentioned above are sufficiently efficient to be applied as ordinary theorem provers as well. Although the class $\mathcal{PVD}$ becomes undecidable under addition of equality, it remains decidable if equality appears in ground form only; a corresponding decision procedure can be obtained by positive resolution and ordered paramodulation [Fermüller and Leitsch 1998]. In equational clause logic ordering is even more important than in ordinary clause logic: Note that unrestricted paramodulation does not even terminate on ground problems like $\{f(a) = a, P(a)\}$.

Another important extension of clause logic is that by equational constraints. In [Pichler 1998] it is shown that many clause classes remain decidable under addition of (equational) constraints; the corresponding decision procedures can be obtained from those in pure clause logic in a natural way.

Bibliography

Andréka H., Németi I. and van Benthem J. [1998], 'Modal languages and bounded fragments of predicate logic', *Journal of Philosophical Logic* **27**(3), 217–274.

Andréka H., van Benthem J. and Németi I. [1995], 'Back and forth between modal logic and classical logic', *Bulletin of the IGPL* **3**(5), 685–720.

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., Fermüller C. and Leitsch A. [1994], A non-elementary speed-up in proof length by structural clause form transformation, *in* 'Proceedings of the 9th Annual IEEE Symposium on Logic in Computer Science (LICS'94)', IEEE Computer Society Press, pp. 213–219.

Bachmair L. and Ganzinger H. [1994], 'Rewrite-based equational theorem proving with selection and simplification', *Journal of Logic and Computation* **4**(3), 217–247.

Bachmair L., Ganzinger H. and Waldmann U. [1993], Superposition with simplification as a decision procedure for the monadic class with equality, *in* 'Computational Logic and Proof Theory, Third Kurt Gödel Colloquium, KGC'93, Brno, Czech Republic, August 1993, *Proceedings*', Vol. 713 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 83–96.

Baumgartner P. and Furbach U. [1996], 'Hyper Tableaux. Part I: Proof Procedure and Model Generation', Dagstuhl-Seminar Reports *Disjunctive logic programming and databases: Non-monotonic aspects.*

Baumgartner P., Furbach U. and Niemelä I. [1996], Hyper Tableaux, *in* 'Logics in AI, Proc. JELIA'96', Vol. 1126 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 1–17.

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

Caferra R. and Zabel N. [1992], 'A method for simultaneous search for refutations and models by equational constraint solving', *J. Symbolic Computation* **13**, 613–641.

Calvanese D., Giacomo G. D., Lenzerini M. and Nardi D. [2001], Reasoning in expressive description logics, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 23.

Ceri S., Gottlob G. and Tanca L. [1990], *Logic Programming and Databases*, Springer Verlag.

Church A. [1936], 'A note on the entscheidungsproblem', *J. Symbolic Logic* **1**, 40–44.

de Nivelle H. [1995], Ordering Refinements of Resolution, PhD thesis, Delft University of Technology.

de Nivelle H. [1997], A classification of non-liftable orders for resolution, *in* W. McCune, ed., 'Automated Deduction – CADE-14', Vol. 1249 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 336–350.

de Nivelle H. [1998*a*], Deciding the E^+-class by an a posteriori, liftable order, ILLC Report ML-1998-030, University of Amsterdam, The Netherlands.

de Nivelle H. [1998*b*], Resolution decided the guarded fragment, ILLC report CT-98-01, University of Amsterdam, The Netherlands.

de Nivelle H. [n.d.], A family of resolution decision procedures, *in* 'Workshop on Logic, Language and Computation, Palo Alto USA, 1998'. To appear.

Denenberg L. and Lewis H. R. [1984], Logical syntax and computational complexity, Vol. 1104 of *Lecture Notes in Mathematics*, Springer Verlag, pp. 101–115.

Donini F. M., Lenzerini M., Nardi D. and Schaerf A. [1994], 'Deduction in concept languages: from subsumption to instance checking', *Journal of Logic and Computation* **4**(4), 423–452.

Dreben B. and Goldfarb W. D. [1979], *The Decision Problem*, Addison-Wesley.

Fermüller C. and Leitsch A. [1993], Model building by resolution, *in* E. B. *et al.*, ed., 'Computer Science Logic, 6th Workshop, CSL'92 San Miniato, Italy, September/Oktober 1992, *Selected Papers*', Vol. 702 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 134–148.

Fermüller C. and Leitsch A. [1996], 'Hyperresolution and automated model building', *J. Symbolic Computation* **6**(2), 173–230.

Fermüller C. and Leitsch A. [1998], 'Decision procedures and model building in equational clause logic', *Logic Journal of the Interest Group in Pure and Applied Logics (IGPL)* **6**(1), 17–41.

Fermüller C., Leitsch A., Tammet T. and Zamov N. [1993], *Resolution Methods for the Decision Problem*, Vol. 679 of *Lecture Notes in Artificial Intelligence*, Springer Verlag.

Fermüller C. and Salzer G. [1993], Ordered paramudulation and resolution as decision procedure, *in* A. Voronkov, ed., 'Logic Programming and Automated Reasoning, 4th International Conference, LPAR'93, St. Petersburg, Russia, July 1993, *Proceedings*', Vol. 698 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 122–133.

Frege G. [1884], *Die Grundlagen der Arithmetik. Eine logisch-mathematische Untersuchung über den Begriff der Zahl*, Breslau.

Fujita M., Slaney J. and Bennett F. [1993], Automatic generation of some results in finite algebra, *in* 'Proc. 13th IJCAI', pp. 52–57.

Ganzinger H. and de Nivelle H. [1999], A superposition decision procedure for the guarded fragment with equality, *in* 'Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science (LICS'99)', IEEE Computer Society Press, pp. 295–304.

Gurevich Y. [1973], Formuly s odnim $\forall$ (formulas with one $\forall$), *in* 'Izbrannye Voprosy Algebry i Logiki (Selected Questions in Algebra and Logic—in Memory of A. Mal'cev)', Nauka, Novosibirsk, pp. 97–100.

Hilbert D. and Ackermann W. [1928], *Grundzüge der theoretischen Logik*, Berlin.

Hustadt U. [1999], Resolution-Based Decision Procedures for Subclasses of First-Order Logic, PhD thesis, Universität des Saarlandes, Saarbrücken, Germany.

Hustadt U. and Schmidt R. A. [1999*a*], Issues of decidability for description logics in the framework of resolution, *in* R. Caferra and G. Salzer, eds, 'Automated Deduction in Classical and Non-Classical Logics: Selected Papers', Vol. 1761 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 192–206.

Hustadt U. and Schmidt R. A. [1999*b*], Maslov's class K revisited, *in* H. Ganzinger, ed., 'Proceedings of the 16th International Conference on Automated Deduction (CADE-16)', Vol. 1632 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 172–186.

Hustadt U. and Schmidt R. A. [1999*c*], On the relation of resolution and tableaux proof systems for description logics, *in* T. Dean, ed., 'Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)', Morgan Kaufmann, pp. 110–115.

Joyner W. H. [1973], Automated Theorem Proving and the Decision Problem, PhD thesis, Harvard University.

Joyner W. H. [1976], 'Resolution strategies as decision procedures', *J. Association of Computing Machinery* **23**(1), 398–417.

Kallik B. [1969], A decision procedure based on the resolution method, *in* 'Information Processing 68 (IFIP 68)', Vol. 1, North-Holland Publishing Company, pp. 269–275.

Klingenbeck S. [1996], Counter Examples in Semantic Tableaux, PhD thesis, University of Karlsruhe.

Kowalski R. and Hayes P. J. [1969], Semantic trees in automated theorem proving, *in* B. Meltzer and D. Michie, eds, 'Machine Intelligence 4', Edinburgh University Press, pp. 87–101.

Leibniz G. W. [1923], Calculus ratiocinator, *in* P. A. der Wissenschaften, ed., 'Sämtliche Schriften und Briefe', Reichel, Darmstadt.

Leitsch A. [1993], 'Deciding clause classes by semantic clash resolution', *Fundamenta Informaticae* **18**, 163–182.

Löwenheim L. [1915], 'Über Möglichkeiten im Relativkalkül', *Mathematische Annalen* **68**, 169–207.

Manthey R. and Bry F. [1988], Satchmo: a theorem prover implemented in Prolog, *in* '9th Conference on Automated Deduction', Vol. 310 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 415–434.

Maslov S. Y. [1968], 'The inverse method for establishing deducibility for logical calculi', *Proc. Steklov Inst. Math.* **98**, 25–96.

Maslov S. Y. [1971], The inverse method for establishing deducibility for logical calculi, *in* V. P. Orevkov, ed., 'The Calculi of Symbolic Logic I: Proceedings of the Steklov Institute of Mathematics edited by I.G. Petrovskiĭ and S. M. Nikol'skiĭ, number 98 (1968)', American Mathematical Society, pp. 25–96.

McCune W. [1995], Otter 3.0 Users Guide, Technical report, Argonne National Laboratory, Argonne (Ill.).

Noll I. [1980], A note on resolution: How to get rid of factoring without loosing completeness, *in* '5th Conference on Automated Deduction', Vol. 87 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 250–263.

Nonnengart A. and Weidenbach C. [2001], Computing small clause normal forms, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 6.

Ohlbach H. J. and Schmidt R. A. [1997], 'Functional translation and second-order frame properties of modal logics', *J. of Logic and Computation* **7**(5), 581–603.

Ohlbach H., Nonnengart A., de Rijke M. and Gabbay D. [2001], Encoding two-valued nonclassical logics in classical logic, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 21.

Peltier N. [1997*a*], 'Increasing the capabilities of model building by constraint solving with terms with integer exponents', *Journal of Symbolic Computation* **24**, 59–101.

Peltier N. [1997*b*], Simplifying formulae in tableaux. Pruning the search space and building models, *in* 'Proceeding of Tableaux'97', Vol. 1227 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 313–327.

Pichler R. [1998], Extending decidable clause classes via constraints, Technical report, Institut f. Computersprachen, TU Wien.

Plaisted D. A. and Greenbaum S. [1986], 'A structure-preserving clause form translation', *J. of Symbolic Computation* **2**, 293–304.

Robinson J. A. [1965*a*], 'The generalized resolution principle', *Intern. Journal of Computer Mathematics* **1**, 227–334.

Robinson J. A. [1965*b*], 'A machine oriented logic based on the resolution principle', *J. Association of Computing Machinery* **12**(1), 23–41.

Schmidt R. A. [1997], Optimised Modal Translation and Resolution, PhD thesis, Universität des Saarlandes, Saarbrücken, Germany.

Schmidt R. A. [1998], Resolution is a decision procedure for many propositional modal logics, *in* M. Kracht, M. de Rijke, H. Wansing and M. Zakharyaschev, eds, 'Advances in Modal Logic, Volume 1', Vol. 87 of *Lecture Notes*, CSLI Publications, Stanford, pp. 189–208.

Schmidt-Schauss M. and Smolka G. [1991], 'Attributive concept description with complements', *Artificial Intelligence* **48**, 1–26.

Slagle J. R. [1967], 'Automatic theorem proving with renamable and semantic resolution', *J. Association of Computing Machinery* **14**(4), 687–697.

Slaney J. [1992], Finder (finite domain enumerator): notes and guide, Technical report, Australian National University Automated Reasoning Project, Canberra.

Stenlund S. [1971], *Combinators λ–Terms and Proof Theory*, Reidel Publ. Comp.

Tammet T. [1991], Using resolution for deciding solvable classes and building finite models, *in* 'Baltic Computer Science', Vol. 502 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 33–64.

Tammet T. [1992], Resolution Methods for Decision Problems and Finite Model Building, PhD thesis, Chalmers University of Technology and University of Göteborg.

Tammet T. [1995], Using resolution for extending KL-ONE-type languages, *in* N. Pissinou, A. Silberschatz, E. K. Park and K. Makki, eds, 'Proceedings of the Fourth International Conference on Information and Knowledge Management (CIKM'95)', ACM Press.

Tammet T. [1996], Separate orderings for ground and non-ground literals preserve completeness of resolution. Unpublished manuscript.

Tammet T. [1997], 'Gandalf', *Journal of Automated Reasoning* **18**(2), 199–204.

Turing A. [1936/37], 'On computable numbers with an application to the Entscheidungsproblem', *Proc. of the London Math. Soc.* **Ser. 2**(42), 230–265.

Weidenbach C., Gaede B. and Rock G. [1996], SPASS and FLOTTER, version 0.42 (system description, *in* M. McRobbie and J. Slaney, eds, 'Automated Deduction – CADE-15', Vol. 1104 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 141–145.

Zamov N. K. [1989], 'Maslov's inverse method and decidable classes', *Annals of Pure and Applied Logic* **42**, 165–194.

Index

W

X