

Testing for Renamability to Classes of Clause Sets*

(Extended abstract)

Albert Brandl Christian G. Fermüller Gernot Salzer[†]
Technische Universität Wien, Austria

Abstract

This paper investigates the problem of testing clause sets for membership in classes known from literature. In particular, we are interested in classes defined via renaming: Is it possible to rename the predicates in a way such that positive and negative literals satisfy certain conditions? We show that for classes like *Horn* or *OCC1N* [5] the existence of such renamings can be decided in polynomial time, whereas the same problem is NP-complete for class *PVD* [5]. The decision procedures are based on hyper-resolution; if a renaming exists, it can be extracted from the final saturated clause set.

1 Introduction

Within the last decade the classical decision problem for predicate logic was revisited in the frame of clause logic. Instead of deciding the validity of first order formulas in prenex normal form—usually without function symbols—the aim was now to decide the satisfiability of clause sets with a rich functional structure. Many classes of clause sets—with and without equality—were shown to be efficiently decidable using refinements of resolution, paramodulation and superposition [1, 5, 6, 8], subsuming most of the classical results. For some classes it was possible to go even further: the output of the decision procedure can be used to derive finite representations of models [3, 4].

When implementing decision procedures one encounters an additional problem: before actually submitting a clause set to the decision procedure for a class CL, one has to check that the clause set satisfies all criteria characterizing CL. Applying a decision procedure to clause sets outside of its scope might yield uninterpretable results or lead to non-termination. For many classes this check is quite simple as the criteria can be tested locally one literal or one clause at a time. However, some classes involve global conditions like renaming;¹ a naïve algorithm would consider all possible renamings, whose number is exponential in the number of different predicate symbols.

*Supported by FWF grant P11624-MAT.

[†]Communicating author. Address: Technische Universität, Karlsplatz 13/E185, A-1040 Wien/Austria. E-mail: salzer@logic.at.

¹*Renaming* a predicate symbol P here means mirroring the polarity of P throughout the whole clause set by replacing all occurrences of $\neg P(\dots)$ by $P(\dots)$ and all occurrences of $P(\dots)$ by $\neg P(\dots)$. The renamed clause set is similar to the original one in many respects; e.g., they are equivalent with respect to satisfiability.

Example 1 To find out whether a given clause set is Horn one has simply to check that the number of positive literals per clause is zero or one. Obviously this test is linear in the size of the clauses. However, it is more complex to check whether a clause set can be *made* Horn by renaming some of its predicate symbols. E.g., the clause set $\{\{P(x), Q(x)\}, \{\neg P(y), \neg Q(y)\}\}$ is not Horn, but can be made Horn by renaming P (or alternatively Q), yielding the set $\{\{\neg P(x), Q(x)\}, \{P(y), \neg Q(y)\}\}$.

In this paper we investigate three instances of the renamability problem. After fixing some notions and notations in section 2, we describe in section 3 a technique—introduced by Harry R. Lewis in [10]—for testing renamability to Horn.² In 4 we show that the renamability problem for *OCCIN* can be decided by a related method in polynomial time. Section 5 proves that the renamability problem for *PVD* is NP-complete. The final section discusses how suitable renamings can be extracted from the information generated by the decision procedures.

2 Basic Notions

For basic notions like clause, literal, etc. we refer the reader to textbooks like [2].

The *dual* of a literal L is denoted by L^d and is defined by $P(\dots)^d = \neg P(\dots)$ and $(\neg P(\dots))^d = P(\dots)$. The *propositional skeleton* of L is denoted by $\text{skel}(L)$ and is defined as $\text{skel}(\neg P(\dots)) = \neg P$, where the second occurrence of P is interpreted as propositional variable. skel is extended to clauses and sets of clauses in the obvious way. The set of variables occurring in a term, atom, literal or clause E is denoted by $\text{var}(E)$. By $\text{occ}(v, E)$ we denote the number of occurrences of variable v in E . For a clause C , the subset of its positive literals is denoted by C^+ , the subset of its negative literals by C^- .

A *renaming* is a set of predicate symbols. The application of a renaming σ to a literal L , denoted by $\sigma(L)$, is L^d if the predicate symbol of L occurs in σ , and L otherwise. The result of renaming a clause $C = \{L_1, \dots, L_n\}$ by σ is $\sigma(C) = \{\sigma(L_1), \dots, \sigma(L_n)\}$. Similarly, a clause set $\mathcal{C} = \{C_1, \dots, C_n\}$ is renamed to $\sigma(\mathcal{C}) = \{\sigma(C_1), \dots, \sigma(C_n)\}$. For every model \mathcal{M} of a propositional clause set, $\sigma_{\mathcal{M}}$ denotes the set of propositional variables that are true in \mathcal{M} .

The *depth* $\tau(t)$ of a term t is defined as $\tau(t) = 0$ if t is a constant or variable, and as $\tau(f(t_1, \dots, t_n)) = 1 + \max\{\tau(t_i) \mid 1 \leq i \leq n\}$ for a functional term. For an atom or literal we define $\tau(\neg P(t_1, \dots, t_n)) = \max\{\tau(t_i) \mid 1 \leq i \leq n\}$. If C is a clause then $\tau(C)$ is an abbreviation for $\max\{\tau(L) \mid L \in C\}$. The *maximal depth of occurrence* $\tau_{\max}(v, t)$ of a variable v in a term t is defined by $\tau_{\max}(v, v) = 0$ and $\tau_{\max}(v, f(t_1, \dots, t_n)) = 1 + \max\{\tau_{\max}(v, t_i) \mid v \in \text{var}(t_i), 1 \leq i \leq n\}$. Analogously, we define the *minimal depth of occurrence* $\tau_{\min}(v, t)$. These definitions are extended to atoms and literals in the obvious way.

3 Testing for Renamability to Horn

Definition 1 A clause set, \mathcal{C} , is Horn iff each of its clauses contains at most one positive literal. \mathcal{C} is renamable to Horn if there is a renaming, σ , such that $\sigma(\mathcal{C})$ is Horn.

²We thank one of the referees for pointing out to us that there exists an impressive body of literature on renamability to Horn. In particular, we learned that we re-discovered Lewis' proof. (We apologize for not having done our homework.)

The importance of the Horn fragment of clause logic is well known. It is also well known that clause sets that are not Horn can often be converted to Horn by systematically renaming the predicate symbols (see example 1). The fact that renamability to Horn is decidable in polynomial, even linear time is well documented in the literature (see e.g. [7, 9–11]). However, since we feel that our proof technique is best explained for the case of clause sets renamable to Horn we briefly re-describe Lewis’ idea [10] of converting the renamability problem into a satisfiability problem for sets of propositional Krom clauses. The significance of the method lies in the fact that it can be adapted to many similar problems; see e.g. section 4. We illustrate the method by an example.

Example 2 Let \mathcal{C} be the set containing the clauses $C_1 = \{P(x), Q(x), R(x)\}$, $C_2 = \{\neg P(y), Q(y)\}$, $C_3 = \{\neg R(x)\}$, and $C_4 = \{\neg P(x), \neg Q(x)\}$. Obviously C_1 is not Horn, therefore \mathcal{C} is not Horn. Of course C_1 can be renamed to Horn, e.g. by applying the renaming $\sigma = \{P, Q\}$. $\sigma(C_2)$ and $\sigma(C_3)$ are Horn, too, but $\sigma(C_4)$ is not. Therefore we have to ‘backtrack’ and to try another candidate for a renaming to Horn. The question is whether we can compute an appropriate renaming σ without backtracking. Observe that C_1 imposes the following restriction on σ if $\sigma(C_1)$ is to be Horn:

either $P \in \sigma$ or $Q \in \sigma$; and either $P \in \sigma$ or $R \in \sigma$; and either $Q \in \sigma$ or $R \in \sigma$.

This just expresses the fact that for every pair of literals in a Horn clause at least one of the two literals has to be negative. Similarly, we obtain the following condition for the only pair of literals in C_2 :

either $P \notin \sigma$ or $Q \in \sigma$.

Since C_3 is singleton there is no corresponding condition: all singleton clauses are Horn by definition. For C_4 we obtain:

either $P \notin \sigma$ or $Q \notin \sigma$.

The conditions are simultaneously satisfiable. This can easily be seen by representing them as a set of propositional Krom³ clauses. If we abbreviate the proposition ‘ $P \in \sigma$ ’ by P and similarly ‘ $P \notin \sigma$ ’ by $\neg P$ (interpreting the predicate symbol as a propositional variable) we obtain the clause set

$$\kappa(\mathcal{C}) = \{\{P, Q\}, \{P, R\}, \{Q, R\}, \{\neg P, Q\}, \{\neg P, \neg Q\}\} .$$

A model for $\kappa(\mathcal{C})$ is given by setting Q and R to true and P to false. It corresponds to the renaming $\sigma = \{Q, R\}$. In fact, the set of models for $\kappa(\mathcal{C})$ represent *all* renamings σ for which $\sigma(\mathcal{C})$ is Horn.

For a clause $C = \{L_1, \dots, L_n\}$, let $\kappa(C)$ be the set $\{\{\text{skel}(L_i), \text{skel}(L_j)\} \mid i \neq j\}$ of propositional Krom clauses. For a clause set \mathcal{C} , let $\kappa(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} \kappa(C)$.

Proposition 1 (Lewis [10]) *For every clause set \mathcal{C} , $\sigma_{\mathcal{M}}(\mathcal{C})$ is Horn iff \mathcal{M} is a model of $\kappa(\mathcal{C})$.*

The satisfiability of propositional Krom clause sets can be tested in polynomial time (e.g. by computing all resolvents). Moreover, $\kappa(\mathcal{C})$ is of at most quadratic size with respect to the size of \mathcal{C} and can easily be computed in polynomial time. Therefore renamability to Horn can be tested in polynomial time. More importantly, the renamings can be efficiently computed by hyper-resolution and splitting (see section 6).

³A Krom clause is a clause containing at most two literals.

4 Testing for Renamability to OCC1N

Definition 2 A clause set, \mathcal{C} , is in class OCC1N iff every clause $C \in \mathcal{C}$ satisfies the following conditions:

(OCC1) $\text{occ}(v, C^+) = 1$ for all $v \in \text{var}(C^+)$;

(OCC2) $\tau_{\max}(v, C^+) \leq \tau_{\min}(v, C^-)$ for all $v \in \text{var}(C^-) \cap \text{var}(C^+)$.

\mathcal{C} is renamable to OCC1N if there is a renaming, σ , such that $\sigma(\mathcal{C})$ is in OCC1N.

In [5] it is shown that the satisfiability of clause sets in OCC1N can be decided by hyper-resolution. Therefore the class of clause sets which are renamable to OCC1N is decidable, too: just apply the decision procedure to the renamed clause set.

Like in the case of renamability to Horn we encode the class membership conditions for each candidate clause by propositional Krom clauses. For a clause C we define three corresponding sets of propositional Krom clauses:

- $\kappa_1(C) = \{\{\text{skel}(L)\} \mid L \in C, \exists v: \text{occ}(v, L) > 1\}$,
- $\kappa_2(C) = \{\{\text{skel}(L), \text{skel}(M)\} \mid L, M \in C, L \neq M, \text{var}(L) \cap \text{var}(M) \neq \emptyset\}$,
- $\kappa_3(C) = \{\{\text{skel}(L), \text{skel}(M)^d\} \mid L, M \in C, \exists v: \tau_{\max}(v, L) > \tau_{\min}(v, M)\}$.

For a clause set \mathcal{C} , let $\kappa(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} (\kappa_1(C) \cup \kappa_2(C) \cup \kappa_3(C))$.

κ_1 encodes the fact that all non-linear literals have to be negative in a clause belonging to a clause set in OCC1N. More exactly, the sign of a literal L where $\text{occ}(v, L) > 1$ for some variable v has to be renamed if L is positive and must remain unchanged if L is negative. κ_2 takes care of the fact that no two different positive literals may share variables. κ_3 corresponds to condition OCC2.

One can show that $\kappa(\mathcal{C})$ is satisfiable iff \mathcal{C} is renamable to OCC1N. We even have:

Proposition 2 For every clause set \mathcal{C} , $\sigma_{\mathcal{M}}(\mathcal{C})$ is in OCC1N iff \mathcal{M} is a model of $\kappa(\mathcal{C})$.

Again, renamability to OCC1N can be decided in polynomial time due to the fact that $\kappa(\mathcal{C})$ is a polynomially bounded set of propositional Krom clauses.

5 Testing for Renamability to PVD

Definition 3 A clause set, \mathcal{C} , is in class PVD iff every clause $C \in \mathcal{C}$ and every variable $v \in \text{var}(C^+)$ satisfies the following condition:

(PVD1) $v \in \text{var}(C^-)$, and $\tau_{\max}(v, C^+) \leq \tau_{\max}(v, C^-)$.

\mathcal{C} is renamable to PVD if there is a renaming, σ , such that $\sigma(\mathcal{C})$ is in PVD.

In [5] it is shown that the satisfiability of clause sets in PVD can be decided by hyper-resolution. Therefore the class of clause sets which are renamable to PVD is decidable, too: just apply the decision procedure to the renamed clause set.⁴

⁴In fact, [5] defines OCC1N and PVD to include also the clause sets renamable to these classes, and uses semantic clash resolution with settings as decision procedure. Finding a suitable setting corresponds to choosing an appropriate renaming. Our view is justified by practice as it is easier to use a single fixed theorem prover for hyper-resolution and to do the renaming as a pre-processing.

We show that for a given clause set, condition PVD1 can be encoded by propositional clauses, which are satisfiable iff the clause set is renamable to PVD. We start by illustrating the main idea by an example.

Example 3 Consider the clause $C = \{P(f(x), y), Q(f(x), f(y)), \neg R(x, f(y))\}$. C does not satisfy condition PVD1 because of variable x which occurs at depth 1 in the positive part and only at depth 0 in the negative part. To satisfy PVD1, at least one of the two literals containing x at maximal depth, $P(f(x), y)$ and $Q(f(x), f(y))$, has to be ‘moved’ to the negative part. This can be achieved by renaming either P or Q (or both). If we abbreviate the proposition ‘predicate symbol P gets renamed’ by just P (interpreting the predicate symbol as a propositional variable), the renaming requirement for x reads $P \vee Q$, or $\{P, Q\}$ in set notation. A similar requirement can be stated for y : either Q has to be renamed (making $Q(f(x), f(y))$ negative), or R is *not* renamed (retaining the negative literal $\neg R(x, f(y))$), which corresponds to $\{Q, \neg R\}$.

If we denote an interpretation by the set of all atoms true in it, the models of the clause set $\{\{P, Q\}, \{Q, \neg R\}\}$ are given by $\{P\}$, $\{Q\}$, $\{P, Q\}$, $\{Q, R\}$, and $\{P, Q, R\}$. Each model describes a renaming which yields a clause in PVD when applied to C . E.g., the model $\{Q, R\}$ corresponds to renaming Q and R and leaving P unchanged.

In general the clause set consists of several clauses. To encode all restrictions we have to construct one propositional clause per variable and per clause.

For a variable v occurring in a clause C , let C_v be the set of all literals in C containing an occurrence of v of maximal depth, i.e., $C_v = \{L \in C \mid v \in \text{var}(L), \tau_{\max}(v, L) = \tau_{\max}(v, C)\}$. For a clause set \mathcal{C} , let $\kappa(\mathcal{C}) = \{\text{skel}(C_v) \mid C \in \mathcal{C}, v \in \text{var}(C)\}$.

Proposition 3 For every clause set \mathcal{C} , $\sigma_{\mathcal{M}}(\mathcal{C})$ is in PVD iff \mathcal{M} is a model of $\kappa(\mathcal{C})$.

To determine the complexity of the renamability problem for PVD, observe that the length of $\kappa(\mathcal{C})$ is polynomially related to the length of \mathcal{C} . By proposition 3, the renamability problem reduces to the satisfiability problem for propositional CNFs. On the other hand, the satisfiability problem can also be reduced to the renamability problem for PVD: given an arbitrary propositional CNF, interpret each propositional variable as a unary predicate symbol and add some dummy variable x as argument. The resulting clause set is renamable to PVD iff the CNF is satisfiable. Therefore both problems are equivalent, and we conclude that the renamability problem for PVD is NP-complete.

6 Extracting Suitable Renamings

In the last three sections we showed that the requirements for a clause set to be renamable to Horn, OCC1N, or PVD, can be encoded as propositional clauses. The clause set is renamable iff the propositional clauses are satisfiable. Moreover, the propositional models are exactly the admissible renamings.

One way to test the satisfiability of a propositional clause set \mathcal{C} is to saturate \mathcal{C} under hyper-resolution; call the saturated set $\text{hyper}(\mathcal{C})$. \mathcal{C} is satisfiable iff $\text{hyper}(\mathcal{C})$ does not contain the empty clause; in this case $\text{hyper}(\mathcal{C})$ can be used to find all models of \mathcal{C} .

Suppose for the moment that \mathcal{C} itself is Horn. Then $\text{hyper}(\mathcal{C})$ is just the set of (singleton sets of) atoms that are true in any model of \mathcal{C} (plus, of course, the initial negative clauses). Thus, $\text{hyper}(\mathcal{C})$ is a representation of a model of \mathcal{C} .

If \mathcal{C} is not Horn we can still compute such a representation by replacing some non-singleton hyper-resolvent in $\text{hyper}(\mathcal{C})$ by one of its literals and saturating the resulting

clause set again. By iterating this process of splitting and saturating we obtain an ‘atomic representation’ of a model of \mathcal{C} . This procedure for building models is described for general first order clauses in detail in [3, 4]. If \mathcal{C} is Krom and propositional as in sections 3 and 4 there are only polynomially many different (hyper-)resolvents of \mathcal{C} , and the model building procedure terminates in polynomial time.

Acknowledgments

We would like to thank the referees for drawing our attention to relevant literature as well as for correcting some important typos.

References

- [1] L. Bachmair, H. Ganzinger, and U. Waldmann. Superposition with simplification as a decision procedure for the monadic class with equality. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Computational Logic and Proof Theory (Third Kurt Gödel Colloquium 1993)*, LNCS 713, pages 83–96. Springer-Verlag, 1993.
- [2] C. L. Chang and R. C. T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.
- [3] C. G. Fermüller and A. Leitsch. Model building by resolution. In *6th Workshop on Computer Science Logic (CSL’92)*, LNCS 702, pages 134–148. Springer-Verlag, 1993.
- [4] C. G. Fermüller and A. Leitsch. Decision procedures and model building in equational clause logic. *Logic Journal of the Interest Group in Pure and Applied Logics (IGPL)*, 1998. To appear.
- [5] C. G. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Methods for the Decision Problem*. LNCS 679 (LNAI). Springer-Verlag, 1993.
- [6] C. G. Fermüller and G. Salzer. Ordered paramodulation and resolution as decision procedure. In A. Voronkov, editor, *Logic Programming and Automated Reasoning (LPAR’93)*, LNCS 698 (LNAI), pages 122–133. Springer, 1993.
- [7] Jean-Jacque Hébrard. A linear algorithm for renaming a set of clauses as a Horn set. *Theoretical Computer Science*, 124:343–350, 1994.
- [8] W. H. Joyner. Resolution strategies as decision procedures. *JACM*, 23(1):398–417, 1976.
- [9] Hans Kleine Büning. Existence of simple propositional formulas. *Information Processing Letters*, 36:177–182, 1990.
- [10] Harry R Lewis. Renaming a set of clauses as a Horn set. *Journal of the ACM*, 25(1):134–135, 1978.
- [11] Heikki Mannila and Kurt Mehlhorn. A fast algorithm for renaming a set of clauses as a Horn set. *Information Processing Letters*, 21:269–272, 1985.