Flat and One-Variable Clauses: Complexity of Verifying Cryptographic Protocols with Single Blind Copying

Helmut Seidl Kumar Neeraj Verma
Institut für Informatik, TU München, Germany
{seidl,verma}@in.tum.de

February 1, 2008

Abstract

Cryptographic protocols with single blind copying were defined and modeled by Comon and Cortier using the new class $\mathcal{C}$ of first order clauses. They showed its satisfiability problem to be in 3-DEXPTIME. We improve this result by showing that satisfiability for this class is NEXPTIME-complete, using new resolution techniques. We show satisfiability to be DEXPTIME-complete if clauses are Horn, which is what is required for modeling cryptographic protocols. While translation to Horn clauses only gives a DEXPTIME upper bound for the secrecy problem for these protocols, we further show that this secrecy problem is actually DEXPTIME-complete.

1 Introduction

Several researchers have pursued modeling of cryptographic protocols using first order clauses [3, 6, 17] and related formalisms like tree automata and set constraints[5, 12, 13]. While protocol insecurity is NP-complete in case of a bounded number of sessions [16], this is helpful only for detecting some attacks. For certifying protocols, the number of sessions cannot be bounded, although we may use other safe abstractions. The approach using first order clauses is particularly useful for this class of problems. A common safe abstraction is to allow a bounded number of nonces, i.e. random numbers, to be used in infinitely many sessions. Security however still remains undecidable [5]. Hence further restrictions are necessary to obtain decidability.

In this direction, Comon and Cortier [6, 8] proposed the notion of protocols with single blind copying. Intuitively this restriction means that agents are allowed to copy at most one piece of data blindly in any protocol step, a restriction satisfied by most protocols in the literature. Comon and Cortier modeled the secrecy problem for these protocols using the new class $\mathcal{C}$ of first order clauses, and showed satisfiability for $\mathcal{C}$ to be decidable [6] in 3-DEXPTIME [8]. The NEXPTIME lower bound is easy. We show in this paper that satisfiability of this class is in NEXPTIME, thus NEXPTIME-complete. If clauses are restricted to be Horn, which suffices for modeling of cryptographic protocols, we show that satisfiability is DEXPTIME-complete (again the lower bound is easy). While translation to clauses only gives a DEXPTIME upper bound for the secrecy problem for this class of protocols, we further show that the secrecy problem for these protocols is also DEXPTIME-complete.

For proving our upper bounds, we introduce several variants of standard ordered resolution with selection and splitting [2]. Notably we consider resolution as consisting of instantiation of clauses, and of generation of propositional implications. This is in the style of Ganzinger and Korovin [10], but we adopt a slightly different approach, and generate interesting implications to obtain optimal complexity. More precisely, while the approach of [10], emphasizes a single phase of instantiation followed by propositional satisfiability checking, we interleave generation of interesting instantiations and propositional implications in an appropriate manner to obtain optimal complexity. We further show how this technique can be employed also in presence of rules for replacement of literals in clauses, which obey some ordering constraints. To deal with the notion of single blind copying we show how terms containing a single variable can be decomposed into simple terms whose unifiers are of very simple forms. As byproducts, we obtain optimal complexity for several subclasses of $\mathcal{C}$, involving so called *flat* and *one-variable* clauses.

Outline: We start in Section 2 by recalling basic notions about first order logic and resolution refinements. In Section 3 we introduce cryptographic protocols with single blind copying, discuss their modeling using the class $\mathcal{C}$ of first order clauses, and show that their secrecy problem is DEXPTIME-hard. To decide the class $\mathcal{C}$ we gradually introduce our techniques by obtaining DEXPTIME-completeness and NEXPTIME-completeness for one-variables clauses and flat clauses in Sections 4 and 5 respectively. In Section 6, the techniques from the two cases are combined with further ideas to show that satisfiability for $\mathcal{C}$ is NEXPTIME-complete. In Section 7 we adapt this proof to show that satisfiability for the Horn fragment of $\mathcal{C}$ is DEXPTIME-complete.

2 Resolution

We recall standard notions from first order logic. Fix a signature Σ of function symbols each with a given arity, and containing at least one zero-ary symbol. Let r be the maximal arity of function symbols in Σ. Fix a set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots\}$ of variables. Note that $\mathbf{x}_1, \mathbf{x}_2, \ldots$ (in bold face) are the actual elements of $\mathbf{X}$, where as $x, y, z, x_1, y_1, \ldots$ are used to represent arbitrary elements of $\mathbf{X}$. The set $T_\Sigma(\mathbf{X})$ of terms built from Σ and $\mathbf{X}$ is defined as usual. T_Σ is the set of *ground terms*, i.e. those not containing any variables. *Atoms* A are of the form $P(t_1, \ldots, t_n)$ where P is an n-ary predicate and t_i's are terms. *Literals* L are either positive literals $+A$ (or simply A) or negative literals $-A$, where A is an atom. $-(-A)$ is another notation for A. $\pm$ denotes $+$ or $-$ and $\mp$ denotes the opposite sign (and similarly for notations $\pm', \mp', \ldots$). A *clause* is a finite set of literals. A *negative clause* is one which contains only negative literals. If M is any term, literal or clause then the set $\mathsf{fv}(M)$ of variables occurring in them is defined as usual. If C_1 and C_2 are clauses then $C_1 \vee C_2$ denotes $C_1 \cup C_2$.

$C \vee \{L\}$ is written as $C \vee L$ (In this notation, we allow the possibility of $L \in C$). If $C_1, \ldots, C_n$ are clauses such that $\mathsf{fv}(C_i) \cap \mathsf{fv}(C_j) = \emptyset$ for $i \neq j$, and if C_i is non-empty for $i \geq 2$, then the clause $C_1 \vee \ldots \vee C_n$ is also written as $C_1 \sqcup \ldots \sqcup C_n$ to emphasize this property. *Ground literals and clauses* are ones not containing variables. A term, literal or clause is *trivial* if it contains no function symbols. A substitution is a function $\sigma : \mathbf{X} \to T_\Sigma(\mathbf{X})$. *Ground substitutions* map every variable to a ground term. We write $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ to say that $x_i\sigma = t_i$ for $1 \leq i \leq n$ and $x\sigma = x$ for $x \notin \{x_1, \ldots, x_n\}$. If M is a term, literal, clause, substitution or set of such objects, then the effect $M\sigma$ of applying σ to M is defined as usual. *Renamings* are bijections $\sigma : \mathbf{X} \to \mathbf{X}$. If M is a term, literal, clause or substitution, then a renaming of M is of the form $M\sigma$ for some renaming σ, and an instance of M is of the form $M\sigma$ for some substitution σ. If M and N are terms or literals then a *unifier* of M and N is a substitution σ such that $M\sigma = N\sigma$. If such a unifier exists then there is also a *most general unifier (mgu)*, i.e. a unifier σ such that for every unifier σ' of M and N, there is some σ'' such that $\sigma' = \sigma\sigma''$. Most general unifiers are unique upto renaming: if σ_1 and σ_2 are two mgus of M and N then σ_1 is a renaming of σ_2. Hence we may use the notation $mgu(M, N)$ to denote one of them. We write $M[x_1, \ldots, x_n]$ to say that $\mathsf{fv}(M) \subseteq \{x_1, \ldots, x_n\}$. If $t_1, \ldots, t_n$ are terms then $M[t_1, \ldots, t_n]$ denotes $M\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. If N is a set of terms them $M[N] = \{M[t_1, \ldots, t_n] \mid t_1, \ldots, t_n \in N\}$. If M is a set of terms, atoms, literals or clauses them $M[N] = \bigcup_{m \in M} m[N]$. A *Herbrand interpretation* $\mathcal{H}$ is a set of ground atoms. A clause C is *satisfied* in $\mathcal{H}$ if for every ground substitution σ, either $A \in \mathcal{H}$ for some $A \in C\sigma$, or $A \notin \mathcal{H}$ for some $-A \in C\sigma$. A set S of clauses is satisfied in $\mathcal{H}$ if every clause of S is satisfied in $\mathcal{H}$. If such a $\mathcal{H}$ exists then S is *satisfiable*, and $\mathcal{H}$ is a *Herbrand model* of S. A *Horn clause* is one containing at most one positive literal. If a set of Horn clauses is satisfiable then it has a least Herbrand model wrt the subset ordering.

Resolution and its refinements are well known methods for testing satisfiability of clauses. Given a strict partial order $<$ on atoms, a literal $\pm A$ is *maximal* in a clause C if there is no literal $\pm' B \in C$ with $A < B$. *Binary ordered resolution* and *ordered factorization* wrt ordering $<$ are defined by the following two rules respectively:

$$\frac{C_1 \vee A \qquad -B \vee C_2}{C_1\sigma \vee C_2\sigma} \qquad\qquad \frac{C_1 \vee \pm A \vee \pm B}{C_1\sigma \vee A\sigma}$$

where $\sigma = mgu(A, B)$ in both rules, A and B are maximal in the left and right premises respectively of the first rule, and A and B are both maximal in the premise of the second rule. We rename the premises of the first rule before resolution so that they don't share variables. The ordering $<$ is *stable* if: whenever $A_1 < A_2$ then $A_1\sigma < A_2\sigma$ for all substitutions σ. We write $S \Rightarrow_< S \cup \{C\}$ to say that C is obtained by one application of the binary ordered resolution or binary factorization rule on clauses in S (the subscript denotes the ordering used).

Another resolution rule is *splitting*. This can be described using *tableaux*. A *tableau* is of the form $S_1 \mid \ldots \mid S_n$, where $n \geq 0$ and each S_i, called a *branch* of the tableau, is a set of clauses (the $\mid$ operator is associative and commutative). A tableau is *satisfiable* if at least one of its branches is satisfiable. The tableau is called *closed* if each S_i contains the empty clause, denoted $\Box$. The *splitting* step on tableaux is defined by the rule

$$\mathcal{T} \mid S \rightarrow_{spl} \mathcal{T} \mid (S \setminus \{C_1 \sqcup C_2\}) \cup \{C_1\} \mid (S \setminus \{C_1 \sqcup C_2\}) \cup \{C_2\}$$

whenever $C_1 \sqcup C_2 \in S$ and C_1 and C_2 are non-empty. C_1 and C_2 are called *components* of the clause $C_1 \sqcup C_2$ being split. It is well known that splitting preserves satisfiability of tableaux. We may choose to apply splitting eagerly, or lazily or in some other fashion. Hence we define a *splitting strategy* to be a function ϕ such that $\mathcal{T} \rightarrow_{spl} \phi(\mathcal{T})$ for all tableaux $\mathcal{T}$. The relation $\Rightarrow_<$ is extended to tableaux as expected. Ordered resolution with splitting strategy is then defined by the rule

$$\mathcal{T}_1 \Rightarrow_{<,\phi} \phi(\mathcal{T}_2) \text{ whenever } \mathcal{T}_1 \Rightarrow_< \mathcal{T}_2$$

This provides us with a well known sound and complete method for testing satisfiability. For any binary relation R, R^* denotes the reflexive transitive closure of R, and R^+ denotes the transitive closure of R.

Lemma 1 *For any set S of clauses, for any stable ordering $<$, and for any splitting strategy ϕ, S is unsatisfiable iff $S \Rightarrow^*_{<,\phi} \mathcal{T}$ for some closed $\mathcal{T}$.*

If all predicates are zero-ary then the resulting clauses are *propositional clauses*. In this case we write $S \models_{\mathrm{p}} T$ to say that every Herbrand model of S is a Herbrand model of T. This notation will also be used when S and T are sets of first order clauses, by treating every (ground or non-ground) atom as a zero-ary predicate. For example $\{P(a), -P(a)\} \models_{\mathrm{p}} \Box$ but $\{P(x), -P(a)\} \not\models_{\mathrm{p}} \Box$. $S \models_{\mathrm{p}} \{C\}$ is also written as $S \models_{\mathrm{p}} C$. If $S \models_{\mathrm{p}} C$ then clearly $S\sigma \models_{\mathrm{p}} C\sigma$ for all substitution σ.

3 Cryptographic Protocols

We assume that Σ contains the binary functions $\{\_\}_\_$ and $\langle \_, \_ \rangle$ denoting encryption and pairing. *Messages* are terms of $T_\Sigma(\mathbf{X})$. A *state* is of the form $S(M_1, \ldots, M_n)$ where S with arity n is from a finite set of *control points* and M_i are messages. It denotes an agent at control point S with messages M_i in its memory. An *initialization state* is a state not containing variables. We assume some strict partial order $<$ on the set of control points. A *protocol rule* is of the form

$$S_1(M_1, \ldots, M_m) : \mathsf{recv}(M) \rightarrow S_2(N_1, \ldots, N_n) : \mathsf{send}(N)$$

where $S_1 < S_2$, M_i, N_j are messages, and M and N are each either a message, or a dummy symbol $?$ indicating nothing is received (resp. sent). For secrecy analysis we can replace $?$ by some public message, i.e. one which is known to everyone including the adversary. The rule says that an agent in state $S_1(M_1, \ldots, M_m)$ can receive message M, send a message N, and then move to state $S_2(N_1, \ldots, N_n)$, thus also modifying the messages in its memory. A *protocol* is a finite set of initialization states and protocol rules. This model is in the style of [9] and [5]. The assumption of single blind copying then says that each protocol rule contains at most one variable (which may occur anywhere any number of times in that rule). For example, the public-key Needham-Schroeder protocol

$$\begin{array}{ll} A \to B: & \{A, N_A\}_{K_B} \\ B \to A: & \{N_A, N_B\}_{K_B} \\ A \to B: & \{N_B\}_{K_B} \end{array}$$

is written in our notation as follows. For every pair of agents A and B in our system (finitely many of them suffice for finding all attacks against secrecy [7, 6]) we have two nonces N^1_{AB} and N^2_{AB} to be used in sessions where A plays the initiator's role and B plays the responder's role. We have initialization states $\mathsf{Init}_0(A, N^1_{AB})$ and $\mathsf{Resp}_0(B, N^2_{AB})$ for all agents A and B. Corresponding to the three lines in the protocol we have rules for all agents A and B

$$\begin{array}{rcl} \mathsf{Init}_0(A, N^1_{AB}){:}\mathsf{recv}(?) & \to & \mathsf{Init}_1(A, N^1_{AB}){:}\mathsf{send}(\{\langle A, N^1_{AB}\rangle\}_{K_B}) \\ \mathsf{Resp}_0(B, N^2_{AB}){:}\mathsf{recv}(\{\langle A, x\rangle\}_{K_B}) & \to & \mathsf{Resp}_1(B, x, N^2_{AB}){:}\mathsf{send}(\{\langle x, N^2_{AB}\rangle\}_{K_A}) \\ \mathsf{Init}_1(A, N^1_{AB}){:}\mathsf{recv}(\{\langle N^1_{AB}, x\rangle\}_{K_A}) & \to & \mathsf{Init}_2(A, N^1_{AB}, x){:}\mathsf{send}(\{x\}_{K_B}) \\ \mathsf{Resp}_1(B, x, N^2_{AB}){:}\mathsf{recv}(\{N^2_{AB}\}_{K_B}) & \to & \mathsf{Resp}_2(B, x, N^2_{AB}){:}\mathsf{send}(?) \end{array}$$

Any initialization state can be created any number of times and any protocol rule can be executed any number of times. The adversary has full control over the network: all messages received by agents are actually sent by the adversary and all messages sent by agents are actually received by the adversary. The adversary can obtain new messages from messages he knows, e.g. by performing encryption and decryption. To model this using Horn clauses, we create a unary predicate reach to model reachable states, and a unary predicate known to model messages known to the adversary. The initialization state $S(M_1, \ldots, M_n)$ is then modeled by the clause $\mathsf{reach}(S(M_1, \ldots, M_n))$, where S is a new function symbol we create. The protocol rule

$$S_1(M_1, \ldots, M_m) : \mathsf{recv}(M) \to S_2(N_1, \ldots, N_n) : \mathsf{send}(N)$$

is modeled by the clauses

$$\begin{array}{c} \mathsf{known}(N) \vee -\mathsf{reach}(S_1(M_1, \ldots, M_m)) \vee -\mathsf{known}(M) \\ \mathsf{reach}(S_2(N_1, \ldots, N_n)) \vee -\mathsf{reach}(S_1(M_1, \ldots, M_m)) \vee -\mathsf{known}(M) \end{array}$$

Under the assumption of single blind copying it is clear that all these clauses are *one-variable clauses*, i.e. clauses containing at most one variable. We need further clauses to express adversary capabilities. The clauses

$$\begin{array}{c} \mathsf{known}(\{\mathbf{x}_1\}_{\mathbf{x}_2}) \vee -\mathsf{known}(\mathbf{x}_1) \vee -\mathsf{known}(\mathbf{x}_2) \\ \mathsf{known}(\mathbf{x}_1) \vee -\mathsf{known}(\{\mathbf{x}_1\}_{\mathbf{x}_2}) \vee -\mathsf{known}(\mathbf{x}_2) \end{array}$$

express the encryption and decryption abilities of the adversary. We have similar clauses for his pairing and unpairing abilities, as well as clauses

$$\mathsf{known}(f(\mathbf{x}_1, \ldots, \mathbf{x}_n)) \vee -\mathsf{known}(\mathbf{x}_1) \vee \ldots \vee -\mathsf{known}(\mathbf{x}_n)$$

for any function f that the adversary knows to apply. All these are clearly *flat clauses*, i.e. clauses of the form

$$C = \bigvee_{i=1}^{k} \pm_i P_i(f_i(x_1^i, \ldots, x_{n_i}^i)) \vee \bigvee_{j=1}^{l} \pm_j Q_j(x_j)$$

where $\{x_1^i, \ldots, x_{n_i}^i\} = \mathsf{fv}(C)$ for $1 \leq i \leq k$. Asymmetric keys, i.e. keys K such that message $\{M\}_K$ can only be decrypted with the inverse key K^{-1}, are also easily dealt with using flat and one-variable clauses. The adversary's knowledge of other data c like agent's names, public keys, etc are expressed by clauses $\mathsf{known}(c)$. Then the least Herbrand model of this set of clauses describes exactly the reachable states and the messages known to the adversary. Then to check whether some message M remains secret, we add the clause $-\mathsf{known}(M)$ and check whether the resulting set is satisfiable.

A set of clauses is in the class $\mathcal{V}_1$ if each of its members is a one-variable clause. A set of clauses is in the class $\mathcal{F}$ if each of its members is a flat clause. More generally we have the class $\mathcal{C}$ proposed by Comon and Cortier [6, 8]: a set of clauses S is in the class $\mathcal{C}$ if for each $C \in S$ one of the following conditions is satisfied.

1. C is a one-variable clause
2. $C = \bigvee_{i=1}^{k} \pm_i P_i(u_i[f_i(x_1^i, \ldots, x_{n_i}^i)]) \vee \bigvee_{j=1}^{l} \pm_j Q_j(x_j)$, where for $1 \leq i \leq k$ we have $\{x_1^i, \ldots, x_{n_i}^i\} = \mathsf{fv}(C)$ and u_i contains at most one variable.

If all clauses are Horn then we have the corresponding classes $\mathcal{V}_1 Horn$, $\mathcal{F}Horn$ and $\mathcal{C}Horn$. Clearly the classes $\mathcal{V}_1$ (resp. $\mathcal{V}_1 Horn$) and $\mathcal{F}$ (resp. $\mathcal{F}Horn$) are included in the class $\mathcal{C}$ (resp. $\mathcal{C}Horn$) since the u_i's above can be trivial. Conversely any clause set in $\mathcal{C}$ can be considered as containing just flat and one-variable clauses. This is because we can replace a clause $C \vee \pm P(u[f(x_1, \ldots, x_n)])$ by the clause $C \vee \pm Pu(f(x_1, \ldots, x_n))$ and add clauses $-Pu(x) \vee P(u[x])$ and $Pu(x) \vee -P(u[x])$ where Pu is a fresh predicate. This transformation takes polynomial time and preserves satisfiability of the clause set. Hence now we need to deal with just flat and one-variable clauses. In the rest of the paper we derive optimal complexity results for all these classes.

Still this only gives us an upper bound for the secrecy problem of protocols since the clauses could be more general than necessary. It turns out, however, that this is not the case. In order to show this we rely on a reduction of the reachability problem for *alternating pushdown systems (APDS)*. In form of Horn clauses, an *APDS* is a finite set of clauses of the form

(i) $P(a)$ where a is a zero-ary symbol

(ii) $P(s[x]) \vee -Q(t[x])$ where s and t involve only unary function symbols, and

(iii) $P(x) \vee -P_1(x) \vee -P_2(x)$

Given any set S of *definite* clauses (i.e. Horn clauses having some positive literal), a ground atom A is *reachable* if A is in the least Herbrand model of S, i.e. if $S \cup \{-A\}$ is unsatisfiable. Reachability in APDS is DEXPTIME-hard [4]. We encode this problem into secrecy of protocols, as in [9]. Let K be a (symmetric) key not known to the adversary. Encode atoms $P(t)$ as messages $\{\langle P, t\rangle\}_K$, by treating P as some data. Create initialization states S_1 and S_2 (no message is stored in the states). Clauses (i-iii) above are translated as

$$\begin{array}{lll} S_1 : & \mathsf{recv}(?) & \rightarrow S_2 : \mathsf{send}(\{\langle P, a\rangle\}_K) \\ S_1 : & \mathsf{recv}(\{\langle Q, t[x]\rangle\}_K) & \rightarrow S_2 : \mathsf{send}(\{\langle P, s[x]\rangle\}_K) \\ S_1 : & \mathsf{recv}(\langle\{\langle P_1, x\rangle\}_K, \{\langle P_2, x\rangle\}_K\rangle) & \rightarrow S_2 : \mathsf{send}(\{\langle P, x\rangle\}_K) \end{array}$$

The intuition is that the adversary cannot decrypt messages encrypted with K. He also cannot encrypt messages with K. He can only forward messages which are encrypted with K. However he has the ability to pair messages. This is utilized in the translation of clause (iii). Then a message $\{M\}_K$ is known to the adversary iff M is of the form $\langle P, t\rangle$ and $P(t)$ is reachable in the APDS.

Theorem 1 *Secrecy problem for cryptographic protocols with single blind copying, with bounded number of nonces but unbounded number of sessions is DEXPTIME-hard, even if no message is allowed to be stored at any control point.*

4 One Variable Clauses: Decomposition of Terms

We first show that satisfiability for the classes $\mathcal{V}_1$ and $\mathcal{V}_1 Horn$ is DEXPTIME-complete. We recall also that although we consider only unary predicates, this is no restriction in the case of one-variable clauses, since we can encode atoms $P(t_1, \ldots, t_n)$ as $P'(f_n(t_1 \ldots, t_n))$ for fresh P' and f_n for every P of arity n. As shown in [6, 8], ordered resolution on one-variable clauses, for a suitable ordering, leads to a linear bound on the height of terms produced. This does not suffice for obtaining a DEXPTIME upper bound and we need to examine the forms of unifiers produced during resolution. We consider terms containing at most one variable (call them *one-variable terms*) to be compositions of simpler terms. A non-ground one-variable term $t[x]$ is called *reduced* if it is not of the form $u[v[x]]$ for any non-ground non-trivial one-variable terms $u[x]$ and $v[x]$. The term $f(g(x), h(g(x)))$ for example is not reduced because it can be written as $f(x, h(x))[g(x)]$. The term $f'(x, g(x), a)$ is reduced. Unifying it with the reduced term $f'(h(y), g(h(a)), y)$ produces ground unifier $\{x \mapsto h(y)[a], y \mapsto a\}$ and both $h(y)$ and a are strict subterms of the given terms. Indeed we find:

Lemma 2 *Let $s[x]$ and $t[y]$ be reduced, non-ground and non-trivial terms where $x \neq y$ and $s[x] \neq t[x]$. If s and t have a unifier σ then $x\sigma, y\sigma \in U[V]$ where U is the set of non-ground (possibly trivial) strict subterms of s and t, and V is the set of ground strict subterms of s and t.*

Proof: See Appendix A.

In case both terms (even if not reduced) have the same variable we have the following easy result:

Lemma 3 *Let σ be a unifier of two non-trivial, non-ground and distinct one-variable terms $s[x]$ and $t[x]$. Then $x\sigma$ is a ground strict subterm of s or of t.*

Proof: See Appendix A.

In the following one-variable clauses are simplified to involve only reduced terms.

Lemma 4 *Any non-ground one-variable term $t[x]$ can be uniquely written as $t[x] = t_1[t_2[\ldots[t_n[x]]\ldots]]$ where $n \geq 0$ and each $t_i[x]$ is non-trivial, non-ground and reduced. This decomposition can be computed in time polynomial in the size of t.*

Proof: We represent $t[x]$ as a DAG by doing maximal sharing of subterms. If $t[x] = x$ then the result is trivial. Otherwise let N be the position in this graph, other than the root node, closest to the root such that N lies on every path from the root to the node corresponding to the subterm x. Let t' be the strict subterm of t at position N and let t_1 be the term obtained from t by replacing the sub-DAG at N by x. Then $t = t_1[t']$ and t_1 is reduced. We then recursively decompose t'.

Uniqueness of decomposition follows from Lemma 2. □

Above and elsewhere, if $n = 0$ then $t_1[t_2[\ldots[t_n[x]]\ldots]]$ denotes x. Now if a clause set contains a clause $C = C' \vee \pm P(t[x])$, with $t[x]$ being non-ground, if $t[x] = t_1[\ldots[t_n[x]]\ldots]$ where each t_i is non-trivial and reduced, then we create fresh predicates $Pt_1\ldots t_i$ for $1 \leq i \leq n-1$ and replace C by the clause $C' \vee \pm Pt_1\ldots t_{n-1}(t_n[x])$. Also we add clauses $Pt_1\ldots t_i(t_{i+1}[x]) \vee -Pt_1\ldots t_{i+1}(x)$ and $-Pt_1\ldots t_i(t_{i+1}[x]) \vee Pt_1\ldots t_{i+1}(x)$ for $0 \leq i \leq n-2$ to our clause set. Note that the predicates $Pt_1\ldots t_i$ are considered invariant under renaming of terms t_j. For $i = 0$, $Pt_1\ldots t_i$ is same as P. Our transformation preserves satisfiability of the clause set. By Lemma 4 this takes polynomial time and eventually all non-ground literals in clauses are of the form $\pm P(t)$ with reduced t. Next if the clause set is of the form $S \cup \{C_1 \cup C_2\}$, where C_1 is non-empty and has only ground literals, and C_2 is non-empty and has only non-ground literals, then we do splitting to produce $S \cup \{C_1\} \mid S \cup \{C_2\}$. This process produces at most exponentially many branches each of which has polynomial size. Now it suffices to decide satisfiability of each branch in DEXPTIME. Hence now we assume that each clause is either:

(**Ca**) a ground clause, or

(**Cb**) a clause containing exactly one variable, each of whose literals is of the form $\pm P(t[x])$ where t is non-ground and reduced.

Consider a set S of clauses of type Ca and Cb. We show how to decide satisfiability of the set S. Wlog we assume that all clauses in S of type Cb contain the variable $\mathbf{x}_1$. Let Ng be the set of non-ground terms $t[\mathbf{x}_1]$ occurring as arguments in literals in S. Let Ngs be the set of non-ground subterms $t[\mathbf{x}_1]$ of terms in Ng. We assume that Ng and Ngs always contain the trivial term $\mathbf{x}_1$, otherwise we add this term to both sets. Let G be the set of ground subterms of terms occurring as arguments in literals in S. The sizes of Ng, Ngs and G are polynomial. Let $S^\dagger$ be the set of clauses of type Ca and Cb which only contain literals of the form $\pm P(t)$ for some $t \in \mathsf{Ng} \cup \mathsf{Ng}[\mathsf{Ngs}[\mathsf{G}]]$ (observe that $\mathsf{G} \subseteq \mathsf{Ngs}[\mathsf{G}] \subseteq \mathsf{Ng}[\mathsf{Ngs}[\mathsf{G}]]$). The size of $S^\dagger$ is at most exponential.

For resolution we use ordering $\prec$: $P(s) \prec Q(t)$ iff s is a strict subterm of t. We call $\prec$ the subterm ordering without causing confusion. This is clearly stable. This is the ordering that we are going to use throughout this paper. In particular this means that if a clause contains literals $\pm P(x)$ and $\pm' Q(t)$ where t is non-trivial and contains

x, then we cannot choose the literal $\pm P(x)$ to resolve upon in this clause. Because of the simple form of unifiers of reduced terms we have:

Lemma 5 *Binary ordered resolution and ordered factorization, wrt the subterm ordering, on clauses in $S^\dagger$ produces clauses which are again in $S^\dagger$ (upto renaming).*

Proof: Factorization on a ground clause doesn't produce any new clause. Now suppose we factorize the non-ground clause $C[\mathbf{x}_1] \vee \pm P(s[\mathbf{x}_1]) \vee \pm P(t[\mathbf{x}_1])$ to produce the clause $C[\mathbf{x}_1]\sigma \vee \pm P(s[\mathbf{x}_1])\sigma$ where $\sigma = mgu(s[\mathbf{x}_1], t[\mathbf{x}_1])$. If the premise has only trivial literals then factorization is equivalent to doing nothing. Otherwise by ordering constraints, s and t are non-trivial. By Lemma 3 either $s[\mathbf{x}_1] = t[\mathbf{x}_1]$ in which case factorization does nothing, or $\mathbf{x}_1\sigma$ is a ground subterm of $s[\mathbf{x}_1]$ or of $t[\mathbf{x}_1]$. In the latter case all literals in $(C[\mathbf{x}_1] \vee P(s[\mathbf{x}_1])\sigma$ are of the form $\pm' Q(t'[\mathbf{x}_1]\sigma))$ where $t'[\mathbf{x}_1] \in \mathsf{Ng}$ and $\mathbf{x}_1\sigma \in \mathsf{G} \subseteq \mathsf{Ngs}[\mathsf{G}]$.

Now we consider binary resolution steps. We have the following cases:

- If both clauses are ground then the result is clear.
- Now consider both clauses $C_1[\mathbf{x}_1]$ and $C_2[\mathbf{x}_1]$ to be non-ground. Before resolution we rename the second clause to obtain $C_2[\mathbf{x}_2]$. Clearly all literals in $C_1[\mathbf{x}_1]$ and $C_2[\mathbf{x}_1]$ are of the form $\pm Q(u[\mathbf{x}_1])$ where $u[\mathbf{x}_1] \in \mathsf{Ng}$. Let $C_1[\mathbf{x}_1] = C_1'[\mathbf{x}_1] \vee P(s[\mathbf{x}_1])$ and $C_2[\mathbf{x}_2] = -P(t[\mathbf{x}_2]) \vee C_2'[\mathbf{x}_2]$ where $P(s[\mathbf{x}_1])$ and $-P(t[\mathbf{x}_2])$ are the literals to be resolved upon in the respective clauses. If $s[\mathbf{x}_1]$ and $t[\mathbf{x}_2]$ are unifiable then from Lemma 2, one of the following cases hold:
  - $s[\mathbf{x}_1] = \mathbf{x}_1$ (the case where $t[\mathbf{x}_2] = \mathbf{x}_2$ is treated similarly). From the definition of $\prec$, for $P(s[\mathbf{x}_1])$ to be chosen for resolution, all literals in $C_1'[\mathbf{x}_1]$ are of the form $\pm Q(\mathbf{x}_1)$. The resolvent is $C[\mathbf{x}_2] = C_1'[\mathbf{x}_1]\sigma \cup C_2'$, where $\sigma = \{\mathbf{x}_1 \mapsto t[\mathbf{x}_2]\}$. Each literal in $C_1'[\mathbf{x}_1]\sigma$ is of the form $\pm Q(t[\mathbf{x}_2])$ and each literal in $C_2'[\mathbf{x}_2]$ is of the form $\pm Q(t'[\mathbf{x}_2])$ where $t' \in \mathsf{Ng}$. Hence $C[\mathbf{x}_1] \in S^\dagger$.
  - $s[\mathbf{x}_1] = t[\mathbf{x}_1]$. Then the resolvent is $C_1'[\mathbf{x}_1] \vee C_2'[\mathbf{x}_1]$.
  - $s[\mathbf{x}_1]$ and $t[\mathbf{x}_2]$ have a mgu σ such that $\mathbf{x}_1\sigma, \mathbf{x}_2\sigma \in \mathsf{Ngs}[\mathsf{G}]$. The resolvent $C_1'[\mathbf{x}_1]\sigma \vee C_2'[\mathbf{x}_2]\sigma$ has only ground atoms of the form $\pm Q(t')$ where $t' \in \mathsf{Ng}[\mathsf{Ngs}[\mathsf{G}]]$.
- Now let the first clause $C_1[\mathbf{x}_1] = C_1'[\mathbf{x}_1] \vee \pm P(t[\mathbf{x}_1])$ be non-ground, and the second clause $C_2 = \mp P(s) \vee C_2'$ be ground with $\pm P(t[\mathbf{x}_1])$ and $\mp P(s)$ being the respective literals chosen from $C_1[\mathbf{x}_1]$ and C_2 for resolution. All literals in $C_1[\mathbf{x}_1]$ are of the form $\pm' Q(t'[\mathbf{x}_1])$ with $t' \in \mathsf{Ng}$. All literals in C_2 are of the form $\pm' Q(t')$ with $t' \in \mathsf{Ng}[\mathsf{Ngs}[\mathsf{G}]]$. Suppose that s and $t[\mathbf{x}_1]$ do unify. We have the following cases:
  - $s \in \mathsf{Ngs}[\mathsf{G}]$. Then the resolvent $C = C_1'[\mathbf{x}_1]\sigma \cup C_2'$ where $\sigma = \{\mathbf{x}_1 \mapsto g\}$ where g is subterm of s. As $s \in \mathsf{Ngs}[\mathsf{G}]$ hence $g \in \mathsf{Ngs}[\mathsf{G}]$. Hence all literals in $C_1'[\mathbf{x}_1]\sigma$ are of the form $\pm Q(t')$ where $t' \in \mathsf{Ng}[\mathsf{Ngs}[\mathsf{G}]]$. Hence $C \in S^\dagger$.

– Now suppose $s \in \mathsf{Ng}[\mathsf{Ngs}[\mathsf{G}]] \setminus \mathsf{Ngs}[\mathsf{G}]$. We must have $s = s_1[s_2]$ for some non-trivial $s_1[\mathbf{x}_1] \in \mathsf{Ng}$ and some $s_2 \in \mathsf{Ngs}[\mathsf{G}]$. This is the interesting case which shows why the terms remain in the required form during resolution. The resolvent is $C = C'_1[\mathbf{x}_1]\sigma \vee C'_2$ where $\sigma = \{\mathbf{x}_1 \mapsto g\}$ is the mgu of $t[\mathbf{x}_1]$ and s for some ground term g. As $t[g] = s_1[s_2]$, $\sigma_1 = \{\mathbf{x}_1 \mapsto g, \mathbf{x}_2 \mapsto s_2\}$ is a unifier of the terms $t[\mathbf{x}_1]$ and $s_1[\mathbf{x}_2]$. By Lemma 2 we have the following cases:
  - $t[\mathbf{x}_1] = \mathbf{x}_1$, so that $g = s \in \mathsf{Ng}[\mathsf{Ngs}[\mathsf{G}]]$. By definition of $\prec$, for $\pm P(t[\mathbf{x}_1])$ to be chosen for resolution, all literals in $C_1[\mathbf{x}_1]$ must be of the form $\pm' Q(\mathbf{x}_1)$. Hence all literals in $C'_1\sigma$ are of the form $\pm' Q(g)$. Hence $C \in S^\dagger$.
  - $t[\mathbf{x}_1] = s_1[\mathbf{x}_1]$. Then $g = s_2 \in \mathsf{Ngs}[\mathsf{G}]$. Hence all literals in $C'_1\sigma$ are of the form $\pm' Q(t'[g])$ where $t'[\mathbf{x}_1] \in \mathsf{Ng}$. Hence $C \in S^\dagger$.
  - $g = \mathbf{x}_1\sigma \in \mathsf{Ngs}[\mathsf{G}]$. Hence all literals in $C'_1\sigma$ are of the form $\pm' Q(t'[g])$ where $t' \in \mathsf{Ng}$. Hence $C \in S^\dagger$. $\square$

Hence to decide satisfiability of $S \subseteq S^\dagger$, we keep generating new clauses of $S^\dagger$ by doing ordered binary resolution and ordered factorization wrt the subterm ordering till no new clause can be generated, and then check whether the empty clause has been produced. Also recall that APDS consist of Horn one-variable clauses. Hence:

Theorem 2 *Satisfiability for the classes $\mathcal{V}_1$ and $\mathcal{V}_1 Horn$ is DEXPTIME-complete.*

5 Flat Clauses: Resolution Modulo Propositional Reasoning

Next we show how to decide the class $\mathcal{F}$ of flat clauses in NEXPTIME. This is well known when the maximal arity r is a constant, or when all non-trivial literals in a clause have the same *sequence* (instead of the same *set*) of variables. But we are not aware of a proof of NEXPTIME upper bound in the general case. We show how to obtain NEXPTIME upper bound in the general case, by doing resolution modulo propositional reasoning. While this constitutes an interesting result of its own, the techniques allow us to deal with the full class $\mathcal{C}$ efficiently. Also this shows that the generality of the class $\mathcal{C}$ does not cost more in terms of complexity. An *ϵ-block* is a one-variable clause which contains only trivial literals. A complex clause C is a flat clause $\bigvee_{i=1}^{k} \pm_i P_i(f_i(x_1^i, \ldots, x_{n_i}^i)) \vee \bigvee_{j=1}^{l} \pm_j Q_j(x_j)$ in which $k \geq 1$. Hence a flat clause is either a complex clause, or an *ϵ-clause* which is defined to be a disjunction of ϵ-blocks, i.e. to be of the form $B_1[x_1] \sqcup \ldots \sqcup B_n[x_n]$ where each B_i is an ϵ-block. ϵ-clauses are difficult to deal with, hence we split them to produce ϵ-blocks. Hence define *ϵ-splitting* as the restriction of the splitting rule in which one of the components is an ϵ-block.

Recall that r is the maximal arity of symbols in Σ. Upto renaming, any complex clause C is such that $\mathsf{fv}(C) \subseteq \mathbf{X}_r = \{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$, and any ϵ-block C is such that $\mathsf{fv}(C) \subseteq \{\mathbf{x}_{r+1}\}$. The choice of $\mathbf{x}_{r+1}$ is not crucial. Now notice that ordered resolution between complex clauses and ϵ-blocks only produces flat clauses, which can then be split to be left with only complex and ϵ-blocks. E.g. Resolution between

$$P_1(\mathbf{x}_1) \vee -P_2(\mathbf{x}_2) \vee P_3(f(\mathbf{x}_1, \mathbf{x}_2)) \vee -P_4(g(\mathbf{x}_2, \mathbf{x}_1))$$

and

$$P_4(g(\mathbf{x}_1, \mathbf{x}_1)) \vee -P_5(h(\mathbf{x}_1)) \vee P_6(\mathbf{x}_1)$$

produces

$$P_1(\mathbf{x}_1) \vee -P_2(\mathbf{x}_1) \vee P_3(f(\mathbf{x}_1, \mathbf{x}_1)) \vee -P_5(h(\mathbf{x}_1)) \vee P_6(\mathbf{x}_1)$$

Resolution between

$$P_2(\mathbf{x}_{r+1}) \quad \text{and} \quad -P_2(f(\mathbf{x}_1, \mathbf{x}_2)) \vee P_3(\mathbf{x}_1) \vee P_4(\mathbf{x}_2)$$

produces $P_3(\mathbf{x}_1) \vee P_4(\mathbf{x}_2)$ which can then be split. The point is that we always choose a non-trivial literal from a clause for resolution, if there is one. As there are finitely many complex clauses and ϵ-blocks this gives us a decision procedure. Note however that the number of complex clauses is doubly exponential. This is because we allow clauses of the form $P_1(f_1(\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_2)) \vee P_2(f_2(\mathbf{x}_2, \mathbf{x}_1)) \vee P_3(f_3(\mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_2)) \vee$ ..., i.e. the nontrivial terms contain arbitrary number of repetitions of variables in arbitrary order. The number of such variable sequences of r variables is exponentially many, hence the number of clauses is doubly exponential. Letting the maximal arity r to be a constant, or forcing all non-trivial literals in a clause to have the same variable sequence would have produced only exponentially many clauses. In presence of splitting, this would have given us the well-known NEXPTIME upper bound, which is also optimal. But we are not aware of a proof of NEXPTIME upper bound in the general case. To obtain NEXPTIME upper bound in the general case we introduce the technique of resolution modulo propositional reasoning.

For a clause C, define the set of its projections as $\pi(C) = C[\mathbf{X}_r]$. Essentially projection involves making certain variables in a clause equal. As we saw, resolution between two complex clauses amounts to propositional resolution between their projections. Define the set $\mathsf{U} = \{f(x_1, \ldots, x_n) \mid f \in \Sigma \text{ and each } x_i \in \mathbf{X}_r\}$ of size exponential in r. Resolution between ϵ-block C_1 and a good complex clause C_2 amounts to propositional resolution of a clause from $C[\mathsf{U}]$ with C_2. Also note that propositional resolution followed by further projection is equivalent to projection followed by propositional resolution. Each complex clause has exponentially many projections. This suggests that we can compute beforehand the exponentially many projections of complex clauses and exponentially many instantiations of ϵ-blocks. All new complex clauses generated by propositional resolution are ignored. But after several such propositional resolution steps, we may get an ϵ-clause, which should then be split and instantiated and used for obtaining further propositional resolvents. In other words we only compute such propositionally implied ϵ-clauses, do splitting and instantiation and iterate the process. This generates all resolvents upto propositional implication. We now formalize our approach. We start with the following observation which is used in this and further sections.

Lemma 6 *Let $x_1, \ldots, x_n, y_1, \ldots, y_n$ be variables, not necessarily distinct, but with $\{x_1, \ldots, x_n\} \cap \{y_1, \ldots, y_n\} = \emptyset$. Then the terms $f(x_1, \ldots, x_n)$ and $f(y_1, \ldots, y_n)$ have an mgu σ such that $\{x_1, \ldots, x_n\}\sigma \subseteq \{x_1, \ldots, x_n\}$ and $y_i\sigma = x_i\sigma$ for $1 \leq i \leq n$.*

For a set S of clauses, $\mathsf{comp}(S)$ is the set of complex clauses in S, $\mathsf{eps}(S)$ the set of ϵ-blocks in S, $\pi(S) = \bigcup_{C \in S} \pi(C)$ and $\mathsf{I}(S) = \pi(\mathsf{comp}(S)) \cup \mathsf{eps}(S)[\mathbf{x}_{r+1}] \cup \mathsf{eps}(S)[\mathsf{U}]$. For sets S and T of complex clauses and ϵ-blocks, $S \sqsubseteq T$ means that:
– if $C \in S$ is a complex clause then $\mathsf{I}(T) \models_{\mathrm{p}} \pi(C)$, and
– if $C \in S$ is an ϵ-block then $C[\mathbf{x}_{r+1}] \in \mathsf{eps}(T)[\mathbf{x}_{r+1}]$.
For tableaux $\mathcal{T}_1$ and $\mathcal{T}_2$ involving only complex clauses and ϵ-blocks we write $\mathcal{T}_1 \sqsubseteq \mathcal{T}_2$ if $\mathcal{T}_1$ can be written as $S_1 \mid \ldots \mid S_n$ and $\mathcal{T}_2$ can be written as $T_1 \mid \ldots \mid T_n$ (note same n) such that $S_i \sqsubseteq T_i$ for $1 \leq i \leq n$. Intuitively $\mathcal{T}_2$ is a succinct representation of $\mathcal{T}_1$. Define the splitting strategy ϕ as the one which repeatedly applies ϵ-splitting on a tableau as long as possible. The relation $\Rightarrow_{\prec,\phi}$ provides us a sound and complete method for testing unsatisfiability. We define the alternative procedure for testing unsatisfiability by using succinct representations of tableaux. We define $\blacktriangleright$ by the rule: $\mathcal{T} \mid S \blacktriangleright \mathcal{T} \mid S \cup \{B_1\} \mid \ldots \mid S \cup \{B_k\}$ whenever $\mathsf{I}(S) \models_{\mathrm{p}} C = B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}]$, C is an ϵ-clause, and $1 \leq i_1, \ldots, i_k \leq r+1$. Then $\blacktriangleright$ simulates $\Rightarrow_{\prec,\phi}$:

Lemma 7 *If S is a set of complex clauses and ϵ-blocks, $S \sqsubseteq T$ and $S \Rightarrow_{\prec,\phi} \mathcal{T}$, then all clauses occurring in $\mathcal{T}$ are complex clauses or ϵ-blocks and $T \blacktriangleright^* \mathcal{T}'$ for some $\mathcal{T}'$ such that $\mathcal{T} \sqsubseteq \mathcal{T}'$.*

Proof: We have the following ways in which $\mathcal{T}$ is obtained from S by doing one resolution step followed by splitting:

- We resolve two ϵ-blocks C_1 and C_2 of S to get an ϵ-block C, and $\mathcal{T} = S \cup \{C\}$. Then $\{C_1[\mathbf{x}_{r+1}], C_2[\mathbf{x}_{r+1}]\} \models_{\mathrm{p}} C[\mathbf{x}_{r+1}]$. Also as $S \sqsubseteq T$ we have $\{C_1[\mathbf{x}_{r+1}], C_2[\mathbf{x}_{r+1}]\} \subseteq \mathsf{eps}(T)[\mathbf{x}_{r+1}]$. We have $\mathsf{I}(T) \models_{\mathrm{p}} C[\mathbf{x}_{r+1}]$. Hence $T \blacktriangleright T \cup \{C[\mathbf{x}_{r+1}]\}$ and clearly $S \cup \{C\} \sqsubseteq T \cup \{C\}$.

- We resolve an ϵ-block $C_1[\mathbf{x}_{r+1}]$ with a complex clause $C_2[\mathbf{x}_1, \ldots, \mathbf{x}_r]$, both from S upto renaming, and we have $C_1[\mathbf{x}_{r+1}] \in \mathsf{eps}(T)[\mathbf{x}_{r+1}]$ and $\mathsf{I}(T) \models_{\mathrm{p}} \pi(C_2)$. By ordering constraints, we have $C_1[\mathbf{x}_{r+1}] = C_1'[\mathbf{x}_{r+1}] \vee \pm P(\mathbf{x}_{r+1})$ and $C_2[\mathbf{x}_1, \ldots, \mathbf{x}_r] = \mp P(f(x_1, \ldots, x_n)) \vee C_2'[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ so that resolution produces $C[\mathbf{x}_1, \ldots, \mathbf{x}_r] = C_1'[f(x_1, \ldots, x_n)] \vee C_2'[\mathbf{x}_1, \ldots, \mathbf{x}_r]$. Clearly $C_1[\mathsf{U}] \cup \{C_2[\mathbf{x}_1, \ldots, \mathbf{x}_r]\} \models_{\mathrm{p}} C[\mathbf{x}_1, \ldots, \mathbf{x}_r]$. Also $\pi(C_1[\mathsf{U}]) = C_1[\mathsf{U}]$. Hence $\mathsf{I}(T) C_1[\mathsf{U}] \cup \pi(C_2) \models_{\mathrm{p}} \pi(C) \supseteq \{C[\mathbf{x}_1, \ldots, \mathbf{x}_r]\}$.

  - If C_1' is not empty or if C_2' has some non-trivial literal then C is a complex clause and $\mathcal{T} = S \cup \{C\} \sqsubseteq T$.
  - If C_1' is empty and C_2' has only trivial literals then $C[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ is an ϵ-clause of the form $B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}]$ with $1 \leq i_1, \ldots, i_k \leq r$. $\mathcal{T} = S \cup \{B_1\} \mid \ldots \mid S \cup \{B_k\}$. Since $\mathsf{I}(T) \models_{\mathrm{p}} C[\mathbf{x}_1, \ldots, \mathbf{x}_r]$, hence $T \blacktriangleright \mathcal{T}'$ where $\mathcal{T}' = T \cup \{B_1\} \mid \ldots \mid T \cup \{B_k\}$ and we have $\mathcal{T} \sqsubseteq \mathcal{T}'$.

- We resolve two complex clauses $C_1[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ and $C_2[\mathbf{x}_1, \ldots, \mathbf{x}_r]$, both from S upto renaming, and we have $\mathsf{I}(T) \models_{\mathrm{p}} \pi(C_1)$ and $\mathsf{I}(T) \models_{\mathrm{p}} \pi(C_2)$. First we rename the second clause as $C_2[\mathbf{x}_{r+1}, \ldots, \mathbf{x}_{2r}]$ by applying the renaming $\sigma_0 = \{\mathbf{x}_1 \mapsto \mathbf{x}_{r+1}, \ldots, \mathbf{x}_r \mapsto \mathbf{x}_{2r}\}$. By ordering constraints, $C_1[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ is of the form $C_1'[\mathbf{x}_1, \ldots, \mathbf{x}_r] \vee \pm P(f(x_1, \ldots, x_n))$ and $C_2[\mathbf{x}_{r+1}, \ldots, \mathbf{x}_{2r}]$ is of the form $\mp P(f(y_1,\ldots,y_n)) \vee C_2'[\mathbf{x}_{r+1},\ldots,\mathbf{x}_{2r}]$ so that $\pm P(f(x_1,\ldots,x_n))$ and $\mp P(f(y_1,\ldots,y_n))$ are the literals to be resolved from the respective clauses. By Lemma 6, the resolvent is $C = C_1'[\mathbf{x}_1,\ldots,\mathbf{x}_r]\sigma \vee C_2'[\mathbf{x}_{r+1},\ldots,\mathbf{x}_{2r}]\sigma$ where σ is such that $\{x_1,\ldots,x_n\}\sigma \subseteq \{x_1,\ldots,x_n\}$ and $y_i\sigma = x_i\sigma$ for $1 \le i \le n$. C is obtained by propositional resolution from $C_1[\mathbf{x}_1,\ldots,\mathbf{x}_r]\sigma \in \pi(C_1)$ and $C_2[\mathbf{x}_{r+1},\ldots,\mathbf{x}_{2r}]\sigma = C_2[\mathbf{x}_1,\ldots,\mathbf{x}_r]\sigma_0\sigma \in \pi(C_2)$. Hence $\pi(C_1)\cup\pi(C_2) \models_{\mathrm{p}} C[\mathbf{x}_1,\ldots,\mathbf{x}_r]$. Hence $\pi(\pi(C_1))\cup\pi(\pi(C_2)) = \pi(C_1)\cup\pi(C_2) \models_{\mathrm{p}} \pi(C)$. As $\mathsf{I}(T) \models_{\mathrm{p}} \pi(C_1)$ and $\mathsf{I}(T) \models_{\mathrm{p}} \pi(C_2)$. hence $\mathsf{I}(T) \models_{\mathrm{p}} \pi(C) \supseteq \{C[\mathbf{x}_1,\ldots,\mathbf{x}_r])\}$.

  - If either C_1' or C_2' contains a non-trivial literal then C is a complex clause and $\mathcal{T} = S \cup \{C\} \sqsubseteq T$.
  - If C_1' and C_2' contain only trivial literals then $C[\mathbf{x}_1,\ldots,\mathbf{x}_r]$ is an ϵ-clause of the form $B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}]$ with $1 \le i_1,\ldots,i_k \le r$. $\mathcal{T} = S \cup \{B_1\} \mid \ldots \mid S \cup \{B_k\}$. As $\mathsf{I}(T) \models_{\mathrm{p}} C[\mathbf{x}_1,\ldots,\mathbf{x}_r]$ we have $T \blacktriangleright \mathcal{T}'$ where $\mathcal{T}' = T \cup \{B_1\} \mid \ldots \mid T \cup \{B_k\}$. Also $\mathcal{T} \sqsubseteq \mathcal{T}'$.

- $C[\mathbf{x}_1,\ldots,\mathbf{x}_r]$ is a renaming of a complex clause in S, and we factor $C[\mathbf{x}_1,\ldots,\mathbf{x}_r]$ to get a complex clause $C[\mathbf{x}_1,\ldots,\mathbf{x}_r]\sigma$ where $\mathbf{X}_r\sigma \subseteq \mathbf{X}_r$, and $\mathcal{T} = S \cup \{C[\mathbf{x}_1,\ldots,\mathbf{x}_r]\sigma\}$. $C[\mathbf{x}_1,\ldots,\mathbf{x}_r]\sigma \in \pi(C)$. Hence $\pi(\{C[\mathbf{x}_1,\ldots,\mathbf{x}_r]\sigma\}) \subseteq \pi(\pi(C)) = \pi(C)$. As $S \sqsubseteq T$ hence $\mathsf{I}(T) \models_{\mathrm{p}} \pi(C)$. Hence $\mathsf{I}(T) \models_{\mathrm{p}} \pi(\{C[\mathbf{x}_1,\ldots,\mathbf{x}_r]\sigma\})$. Hence we have $\mathcal{T} = S \cup \{C[\mathbf{x}_1,\ldots,\mathbf{x}_r]\sigma\} \sqsubseteq T$. $\square$

Hence we have completeness of $\blacktriangleright$:

Lemma 8 *If a set S of good complex clauses and ϵ-blocks is unsatisfiable then $S \blacktriangleright^* \mathcal{T}$ for some closed $\mathcal{T}$.*

Proof: By Lemma 1, $S \Rightarrow^*_{\prec,\phi} S_1 \mid \ldots \mid S_n$ such that each $S_i \ni \square$. As $S \sqsubseteq S$, hence by Lemma 7, we have some $T_1,\ldots,T_n$ such that $S \blacktriangleright^* T_1 \mid \ldots \mid T_n$ and $S_i \sqsubseteq T_i$ for $1 \le i \le n$. Since $\square \in S_i$ and $\square$ is an ϵ-block, hence $\square \in T_i$ for $1 \le i \le n$. $\square$

Call a set S of complex clauses and ϵ-blocks *saturated* if the following condition is satisfied: if $\mathsf{I}(S) \models_{\mathrm{p}} B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}]$ with $1 \le i_1,\ldots,i_k \le r+1$, each B_i being an ϵ-block, then there is some $1 \le j \le k$ such that $B_j[\mathbf{x}_{r+1}] \in S[\mathbf{x}_{r+1}]$.

Lemma 9 *If S is a satisfiable set of complex clauses and ϵ-blocks then $S \blacktriangleright^* \mathcal{T} \mid T$ for some $\mathcal{T}$ and some saturated set T of complex clauses and ϵ-blocks, such that $\square \notin T$.*

Proof: We construct a sequence $S = S_0 \subseteq S_1 \subseteq S_2 \subseteq \ldots$ of complex clauses and ϵ-blocks such that S_i is satisfiable and $S_i \blacktriangleright^* S_{i+1} \mid \mathcal{T}_i$ for some $\mathcal{T}_i$ for each i. $S = S_0$ is satisfiable by assumption. Now assume we have already defined $S_0,\ldots,S_i$ and $\mathcal{T}_0,\ldots,\mathcal{T}_{i-1}$. Let $C^l = B_1^l[\mathbf{x}_{i_1^l}] \sqcup \ldots \sqcup B_k^l[\mathbf{x}_{i_{k_l}^l}]$ for $1 \le l \le N$ be all the possible ϵ-clauses such that $\mathsf{I}(S_i) \models_{\mathrm{p}} C^l$, $1 \le i_1^l,\ldots,i_{k_l}^l \le r+1$. Since S_i is satisfiable, $S_i \cup \{C^l \mid 1 \le l \le N\}$ is satisfiable. Since $\mathbf{x}_{i_1^l},\ldots,\mathbf{x}_{i_{k_l}^l}$ are mutually distinct for $1 \le l \le N$, there are $1 \le j_l \le k_l$ for $1 \le l \le N$ such that $S_i \cup \{B_{j_l}^l \mid 1 \le l \le N\}$ is satisfiable. Let $S_{i+1} = S_i \cup \{B_{j_l}^l \mid 1 \le l \le N\}$. S_{i+1} is satisfiable. Also it is clear that $S_i \blacktriangleright^* S_{i+1} \mid \mathcal{T}_i$ for some $\mathcal{T}_i$. If $S_{i+1} = S_i$ then S_i is saturated, otherwise S_{i+1} has strictly more ϵ-blocks upto renaming. As there are only finitely many ϵ-blocks upto renaming, eventually we will end up with a saturated set T in this way. Since T is satisfiable, $\Box \notin T$. From construction it is clear that there is some $\mathcal{T}$ such that $S \blacktriangleright^* \mathcal{T} \mid T$. $\Box$

Theorem 3 *Satisfiability for the class $\mathcal{F}$ is NEXPTIME-complete.*

Proof: The lower bound comes from reduction of satisfiability of positive set constraints which is NEXPTIME-complete [1]. For the upper bound let S be a finite set of flat clauses. Repeatedly apply ϵ-splitting to obtain $f(S) = S_1 \mid \ldots \mid S_m$. S is satisfiable iff some S_i is satisfiable. The number m of branches in $f(S)$ is at most exponential. Also each branch has size linear in the size of S. We non-deterministically choose some S_i and check its satisfiability in NEXPTIME.

Hence wlog we may assume that the given set S has only complex clauses and ϵ-blocks. We non-deterministically choose a certain number of ϵ-blocks $B_1, \ldots, B_N$ and check that $T = S_1 \cup \{B_1, \ldots, B_N\}$ is saturated and $\Box \notin T$. By Lemma 9, if S is satisfiable then clearly there is such a set T. Conversely if there is such a set T, then whenever $T \blacktriangleright^* \mathcal{T}$, we will have $\mathcal{T} = T \mid \mathcal{T}'$ for some $\mathcal{T}'$. Hence we can never have $T \blacktriangleright^* \mathcal{T}$ where $\mathcal{T}$ is closed. Then by Lemma 8 we conclude that T is satisfiable. Hence $S \subseteq T$ is also satisfiable.

Guessing the set T requires non-deterministically choosing from among exponentially many ϵ-blocks. To check that T is saturated, for every ϵ-clause $C = B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}]$, with $1 \leq i_1, \ldots, i_k \leq r+1$, and $B_j[\mathbf{x}_{r+1}] \notin T[\mathbf{x}_{r+1}]$ for $1 \leq j \leq k$, we check that $\mathsf{l}(T) \nvDash_{\mathrm{p}} C$, i.e. $\mathsf{l}(T) \cup \neg C$ is propositionally satisfiable (where $\neg(L_1 \vee \ldots \vee L_n)$ denotes $\{-L_1, \ldots, -L_n\}$). This can be checked in NEXPTIME since propositional satisfiability can be checked in NPTIME. We need to do such checks for at most exponentially many possible values of C. $\Box$

6 Combination: Ordered Literal Replacement

Combining flat and one-variable clauses creates additional difficulties. First observe that resolving a one variable clause $C_1 \vee \pm P(f(s_1[x], \ldots, s_n[x]))$ with a complex clause $\mp P(f(x_1, \ldots, x_n)) \vee C_2$ produces a one-variable clause. If $s_i[x] = s_j[x]$ for all $x_i = x_j$, and if C_2 contains a literal $P(x_i)$ then the resolvent contains a literal $P(s_i[x])$. The problem now is that even if $f(s_1[x], \ldots, s_n[x])$ is reduced, $s_i[x]$ may not be reduced. E.g. $f(g(h(x)), x)$ is reduced but $g(h(x))$ is not reduced. Like in Section 4 we may think of replacing this literal by simpler literals involving fresh predicates. Firstly we have to ensure that in this process we do not generate infinitely many predicates. Secondly it is not clear that mixing ordered resolution steps with replacement of literals is still complete. Correctness is easy to show since the new clause is in some sense equivalent to the old deleted clause. However deletion of clauses arbitrarily can violate completeness of the resolution procedure. The key factor which preserves completeness is that we replace literals by smaller literals wrt the given ordering $<$.

Formally a *replacement rule* is of the form $A_1 \to A_2$ where A_1 and A_2 are (not necessarily ground) atoms. The clause set *associated* with this rule is $\{A_1 \vee -A_2, -A_1 \vee$

$A_2\}$. Intuitively such a replacement rule says that A_1 and A_2 are equivalent. The clause set $cl(\mathcal{R})$ associated with a set $\mathcal{R}$ of replacement rules is the union of the clause sets associated with the individual replacement rules in $\mathcal{R}$. Given a stable ordering $<$ on atoms, a replacement rule $A_1 \rightarrow A_2$ is *ordered* iff $A_2 < A_1$. We define the relation $\rightarrow_{\mathcal{R}}$ as: $S \rightarrow_{\mathcal{R}} (S \setminus \{\pm A_1\sigma \vee C\}) \cup \{\pm A_2\sigma \vee C\}$ whenever S is a set of clauses, $\pm A_1\sigma \vee C \in S$, $A_1 \rightarrow A_2 \in \mathcal{R}$ and σ is some substitution. Hence we replace literals in a clause by smaller literals. The relation is extended to tableaux as usual. This is reminiscent of the well-studied case of resolution with some equational theory on terms. There, however, the ordering $<$ used for resolution is compatible with the equational theory and one essentially works with the equivalence classes of terms and atoms. This is not the case here.

Next note that in the above resolution example, even if $f(s_1[x], \ldots, s_n[x])$ is non-ground, some s_i may be ground. Hence the resolvent may have ground as well as non-ground literals. We avoided this in Section 4 by initial preprocessing. Now we may think of splitting these resolvents during the resolution procedure. This however will be difficult to simulate using the alternative resolution procedure on succinct representations of tableaux because we will generate doubly exponentially many one-variable clauses. To avoid this we use a variant of splitting called *splitting-with-naming* [15]. Instead of creating two branches after splitting, this rule puts both components into the same set, but with tags to simulate branches produced by ordinary splitting. Fix a finite set $\mathbb{P}$ of predicate symbols. $\mathbb{P}$-clauses are clauses whose predicates are all from $\mathbb{P}$. Introduce fresh zero-ary predicates $\overline{C}$ for $\mathbb{P}$-clauses C modulo renaming, i.e. $\overline{C_1} = \overline{C_2}$ iff $C_1\sigma = C_2$ for some renaming σ. Literals $\pm\overline{C}$ for $\mathbb{P}$-clauses C are *splitting literals*. The *splitting-with-naming* rule is defined as: $S \rightarrow_{nspl} (S \setminus \{C_1 \sqcup C_2\}) \cup \{C_1 \vee -\overline{C_2}, \overline{C_2} \vee C_2\}$ where $C_1 \sqcup C_2 \in S$, C_2 is non-empty and has only non-splitting literals, and C_1 has at least one non-splitting literal. Intuitively $\overline{C_2}$ represents the negation of C_2. We will use both splitting and splitting-with-naming according to some predefined strategy. Hence for a finite set $\mathcal{Q}$ of splitting atoms, define *$\mathcal{Q}$-splitting* as the restriction of the splitting-with-naming rule where the splitting atom produced is restricted to be from $\mathcal{Q}$. Call this restricted relation as $\rightarrow_{\mathcal{Q}-nspl}$. This is extended to tableaux as usual. Now once we have generated the clauses $C_1 \vee -\overline{C_2}$ and $\overline{C_2} \vee C_2$ we would like to keep resolving on the second part of the second clause till we are left with the clause $\overline{C_2}$ (possibly with other positive splitting literals) which would then be resolved with the first clause to produce C_1 (possibly with other positive splitting literals) and only then the literals in C_1 would be resolved upon. Such a strategy cannot be ensured by ordered resolution, hence we introduce a new rule. An ordering $<$ over non-splitting atoms is extended to the ordering $<_s$ by letting $q <_s A$ whenever q is a splitting atom and A is a non-splitting atom, and $A <_s B$ whenever A, B are non-splitting atoms and $A < B$. We define *modified ordered binary resolution* by the following rule:

$$\frac{C_1 \vee A \qquad -B \vee C_2}{C_1\sigma \vee C_2\sigma}$$

where $\sigma = mgu(A, B)$ and the following conditions are satisfied:
(1) C_1 has no negative splitting literal, and A is maximal in C_1.
(2) (a) either $B \in \mathcal{Q}$, or

(b) C_2 has no negative splitting literal, and B is maximal in C_2.

As usual we rename the premises before resolution so that they don't share variables. This rule says that we must select a negative splitting literal to resolve upon in any clause, provided the clause has at least one such literal. If no such literal is present in the clause, then the ordering $<_s$ enforces that a positive splitting literal will not be selected as long as the clause has some non-splitting literal. We write $S \Rrightarrow_{<_s} S \cup \{C\}$ to say that C is obtained by one application of the modified binary ordered resolution or the (unmodified) ordered factorization rule on clauses in S. This is extended to tableaux as usual. A *$\mathcal{Q}$-splitting-replacement strategy* is a function ϕ such that $\mathcal{T}(\rightarrow_{\mathcal{Q}-nspl} \cup \rightarrow_{spl} \cup \rightarrow_{\mathcal{R}})^* \phi(\mathcal{T})$ for any tableaux $\mathcal{T}$. Hence we allow both normal splitting and $\mathcal{Q}$-splitting. Modified ordered resolution with $\mathcal{Q}$-splitting-replacement strategy ϕ is defined by the relation: $S \Rrightarrow_{<_s,\phi,\mathcal{R}} \phi(T)$ whenever $S \cup cl(\mathcal{R}) \Rrightarrow_{<_s} T$. This is extended to tableaux as usual. The above modified ordered binary resolution rule can be considered as an instance of *ordered resolution with selection* [2], which is known to be sound and complete even with splitting and its variants. Our manner of extending $<$ to $<_s$ is essential for completeness. We now show that soundness and completeness hold even under arbitrary ordered replacement strategies. It is not clear to the authors if such rules have been studied elsewhere. Wlog we forbid the useless case of replacement rules containing splitting symbols. The relation $<$ is *enumerable* if the set of all ground atoms can be enumerated as $A_1, A_2, \ldots$ such that if $A_i < A_j$ then $i < j$. The subterm ordering is enumerable.

Theorem 4 *Modified ordered resolution, wrt a stable and enumerable ordering, with splitting and $\mathcal{Q}$-splitting and ordered literal replacement is sound and complete for any strategy. I.e. for any set S of $\mathbb{P}$-clauses, for any strict stable and enumerable partial order $<$ on atoms, for any set $\mathcal{R}$ of ordered replacement rules, for any finite set $\mathcal{Q}$ of splitting atoms, and for any $\mathcal{Q}$-splitting-replacement strategy ϕ, $S \cup cl(\mathcal{R})$ is unsatisfiable iff $S \Rrightarrow^*_{<_s,\phi,\mathcal{R}} \mathcal{T}$ for some closed $\mathcal{T}$.*

Proof: See Appendix B.

For the rest of this section fix a set $\mathbb{S}$ of one-variable $\mathbb{P}$-clauses and complex $\mathbb{P}$-clauses whose satisfiability we need to decide. Let Ng be the set of non-ground terms occurring as arguments in literals in the one-variable clauses of $\mathbb{S}$. We rename all terms in Ng to contain only the variable $\mathbf{x}_{r+1}$. Wlog assume $\mathbf{x}_{r+1} \in \mathsf{Ng}$. Let Ngs be the set of non-ground subterms of terms in Ng, and $\mathsf{Ngr} = \{s[\mathbf{x}_{r+1}] \mid s \text{ is non-ground and reduced,and for some } t, s[t] \in \mathsf{Ngs}\}$. Define $\mathsf{Ngrr} = \{s_1[\ldots[s_m] \ldots] \mid s_1[\ldots[s_n]\ldots] \in \mathsf{Ngs}, m \leq n, \text{ and each } s_i \text{ is non-trivial and reduced}\}$. Define the set of predicates $\mathbb{Q} = \{Ps \mid P \in \mathbb{P}, s \in \mathsf{Ngrr}\}$. Note that $\mathbb{P} \subseteq \mathbb{Q}$. Define the set of replacement rules $\mathcal{R} = \{Ps_1 \ldots s_{m-1}(s_m[\mathbf{x}_{r+1}]) \rightarrow Ps_1 \ldots s_m([\mathbf{x}_{r+1}]) \mid Ps_1 \ldots s_m \in \mathbb{Q}\}$. They are clearly ordered wrt $\prec$. Let G be the set of ground subterms of terms occurring as arguments in literals in $\mathbb{S}$. Define the set $\mathcal{Q}_0 = \{\overline{\pm P(t)} \mid P \in \mathbb{P}, t \in \mathsf{G}\}$ of splitting atoms. Their purpose is to remove ground literals from a non-ground clause. All sets defined above have polynomial size. Let $\mathcal{Q} \supseteq \mathcal{Q}_0$ be any set of splitting atoms. For dealing with the class $\mathcal{C}$ we only need $\mathcal{Q} = \mathcal{Q}_0$, but for a more precise analysis of the Horn fragment in the next Section, we need $\mathcal{Q}$ to also contain some other splitting atoms. We also need the set $\mathsf{Ngr}_1 = \{\mathbf{x}_{r+1}\} \cup$

$\{f(s_1,\ldots,s_n) \mid \exists g(t_1,\ldots,t_m) \in \mathsf{Ngr} \cdot \{s_1,\ldots,s_n\} = \{t_1,\ldots,t_m\}\}$ which has exponential size. These terms are produced by resolution of non-ground one-variable clauses with complex clauses, and are also reduced. In the ground case we have the set $\mathsf{G}_1 = \{f(s_1,\ldots,s_n) \mid \exists g(t_1,\ldots,t_m) \in \mathsf{G} \mid \{s_1,\ldots,s_n\} = \{t_1,\ldots,t_m\}\}$ of exponential size. For a set $\mathbb{P}'$ of predicates and a set U of terms, the set $\mathbb{P}'[U]$ of atoms is defined as usual. For a set V of atoms the set $-V$ and $\pm V$ of literals is defined as usual. The following types of clauses will be required during resolution:

(C1) clauses $C \vee D$, where C is an ϵ-block with predicates from $\mathbb{Q}$, and $D \subseteq \pm\mathcal{Q}$.

(C2) clauses $C \vee D$ where C is a renaming of a one-variable clause with literals from $\pm\mathbb{Q}(\mathsf{Ngr}_1)$, C has at least one non-trivial literal, and $D \subseteq \pm\mathcal{Q}$.

(C3) clauses $C \vee D$ where C is a non-empty clause with literals from $\pm\mathbb{Q}(\mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]])$, and $D \subseteq \pm\mathcal{Q}$.

(C4) clauses $C \vee D$ where $C = \bigvee_{i=1}^{k} \pm_i P_i(f_i(x_1^i,\ldots,x_{n_i}^i)) \vee \bigvee_{j=1}^{l} \pm_j Q_j(x_j)$ is a complex clause with each $P_i \in \mathbb{Q}$, each $n_i \geq 2$, each $Q_j \in \mathbb{P}$ and $D \subseteq \pm\mathcal{Q}$

We have already argued why we need splitting literals in the above clauses, and why we need Ngr_1 instead of Ngr in type C2. In type C3 we have Ngrr in place of the set Ngs that we had in Section 4, to take care of interactions between one-variable clauses and complex clauses. In type C4 the trivial literals involve predicates only from $\mathbb{P}$ (and not $\mathbb{Q}$). This is what ensures that we need only finitely many fresh predicates (those from $\mathbb{Q} \setminus \mathbb{P}$) because these are the literals that are involved in replacements when this clause is resolved with a one-variable clause. We have also required that each $n_i \geq 2$. This is only to ensure that types C2 and C4 are disjoint. The clauses that are excluded because of this condition are necessarily of type C2.

The $\mathcal{Q}_0$-splitting steps that we use in this section consist of replacing a tableau $\mathcal{T} \mid S$ by the tableau $\mathcal{T} \mid (S \setminus \{C \vee L\}) \cup \{C \vee -\overline{L}, \overline{L} \vee L\}$, where C is non-ground, $L \in \pm\mathbb{P}(\mathsf{G})$ and $C \vee L \in S$. The replacement steps we are going to use are of the following kind:

(1) replacing clause $C_1[x] = C \vee \pm P(t_1[\ldots[t_n[s[x]]]\ldots])$ by clause $C_2[x] = C \vee \pm Pt_1 \ldots t_n(s[x])\}$ where $P \in \mathbb{P}$, $s[\mathbf{x}_{r+1}] \in \mathsf{Ngr}$ is non-trivial, and $t_1[\ldots[t_n]\ldots] \in \mathsf{Ngrr}$. We have $\{C_1[\mathbf{x}_{r+1}]\} \cup cl(\mathcal{R})[\mathsf{Ngrr}] \vDash_{\mathrm{p}} C_2[\mathbf{x}_{r+1}]$.

(2) replacing ground clause $C_1 = C \vee \pm P(t_1[\ldots[t_n[g]]\ldots])$ by clause $C_2 = C \vee \pm Pt_1 \ldots t_n[g]\}$ where $P \in \mathbb{P}, g \in \mathsf{Ngrr}[\mathsf{G}_1]$ and $t_1[\ldots[t_n]\ldots] \in \mathsf{Ngrr}$. This replacement is done only when $t_1[\ldots[t_n[g]]\ldots] \in \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]] \setminus \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. We have $\{C_1\} \cup cl(\mathcal{R})[\mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] \vDash_{\mathrm{p}} C_2$.

Define the $\mathcal{Q}_0$-splitting-replacement strategy ϕ as one which repeatedly applies first ϵ-splitting, then the above $\mathcal{Q}_0$-splitting steps, then the above two replacement steps till no further change is possible. Then $\Rrightarrow_{\prec_s,\phi,\mathcal{R}}$ gives us a sound and complete method for testing unsatisfiability.

As in Section 5 we now define a succinct representation of tableaux and an alternative resolution procedure for them. As we said, a literal $\overline{L} \in \mathcal{Q}_0$ represents $-L$. Hence for a clause C we define $\underline{C}$ as the clause obtained by replacing every $\pm\overline{L}$ by the literal $\mp L$. This is extended to sets of clauses as usual. Observe that if $S \vDash_{\mathrm{p}} C$ then $\underline{S} \vDash_{\mathrm{p}} \underline{C}$. As before $\mathsf{U} = \{f(x_1,\ldots,x_n) \mid f \in \Sigma,$ and each $x_i \in \mathbf{X}_r\}$. The functions eps and comp of Section 5 are now modified to return clauses of type C1 and C2 respectively. For a set S of clauses, define $\mathsf{ov}(S)$ as the set of clauses of type C2 in S. The function π is as before. We need to define which kinds of instantiations are to be used to generate propositional implications. For a clause C, define

$$\begin{array}{ll} \mathsf{I}_1(C) = & C[\mathsf{U}[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G_1}]]]] \cup C[\mathsf{Ngr_1}] \cup C[\mathsf{Ngr_1}[\mathsf{Ngrr}[\mathsf{G_1}]]] \\ \mathsf{I}_2(C) = & \{C[\mathbf{x}_{r+1}]\} \cup C[\mathsf{Ngrr}[\mathsf{G_1}]] \\ \mathsf{I}_3(C) = & \{C\} \\ \mathsf{I}_4(C) = & \pi(C) \cup C[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G_1}]]] \end{array}$$

The instantiations defined by I_i are necessary for clauses of type Ci. Observe that $C[U] \subseteq \mathsf{I}_1(C)$. For a set S of clauses, define $\mathsf{I}_i(S) = \bigcup_{C \in S} \mathsf{I}_i(C)$. For a set S of clauses of type C1-C4 define $\mathsf{I}(S) = \mathsf{I}_1(\underline{\mathsf{eps}(S)}) \cup \mathsf{I}_2(\underline{\mathsf{ov}(S)}) \cup \mathsf{I}_3(\underline{\mathsf{gr}(S)}) \cup \mathsf{I}_4(\underline{\mathsf{comp}(S)}) \cup \underline{cl(\mathcal{R})[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G_1}]]]}$. Note that instantiations of clauses in $cl(\mathcal{R})$ are necessary for the replacement rules, as argued above. For a set T of clauses define the following properties:

- C satisfies property P1$_T$ iff $C[\mathbf{x}_{r+1}] \in T$.
- C satisfies property P2$_T$ iff $\mathsf{I}(T) \models_{\mathrm{p}} \mathsf{I}_2(\underline{C})$.
- C satisfies property P3$_T$ iff $\mathsf{I}(T) \models_{\mathrm{p}} \mathsf{I}_3(\underline{C})$.
- C satisfies property P4$_T$ iff $\mathsf{I}(T) \models_{\mathrm{p}} \mathsf{I}_4(\underline{C})$.

For sets of clauses S and T, define $S \sqsubseteq T$ to mean that every $C \in S$ is of type Ci and satisfies property Pi_T for some $1 \leq i \leq 4$. This is extended to tableaux as usual. We first consider the effect of one step of the above resolution procedure without splitting. Accordingly let ϕ_0 be the variant of ϕ which applies replacement rules and $\mathcal{Q}_0$-splitting, but no ϵ-splitting.

Lemma 10 *Let S be a set of clauses of type C1-C4. If $S \Rrightarrow_{\prec_s, \phi_0, \mathcal{R}} S'$ then one of the following statements holds.*

- $S' \sqsubseteq S$
- *$S' = S \cup \{C\} \cup S''$, C is a renaming of $B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}] \sqcup D$, each B_i is an ϵ-block, $1 \leq i_1, \ldots, i_k \leq r$, $D \subseteq \pm\mathcal{Q}$, $\mathsf{I}(S) \models_{\mathrm{p}} \underline{C}$, and S'' is a set of clauses of type C3 and $\emptyset \models_{\mathrm{p}} \underline{S''}$. If $k \geq 2$ then D has no literals $-q$ with $q \in \mathcal{Q} \setminus \mathcal{Q}_0$.*

Proof: The set S'' in the second statement will contain the clauses $\overline{L} \vee L$ added by $\mathcal{Q}_0$-splitting, while C will be the clause produced by binary resolution or factoring, possibly followed by applications of replacement rules and by replacement of ground literals L by $-\overline{L}$. Hence $S'' = \emptyset$ in all cases except when we need to perform $\mathcal{Q}_0$-splitting.

First we consider resolution steps where splitting literals are resolved upon. A positive splitting literal cannot be chosen to resolve upon in a clause unless the clause has no literals other than positive splitting literals. Hence this clause is $C_1 = q \vee q_1 \vee \ldots \vee q_m$ of type C1, The other clause must be $C_2 = C_2' \vee -q$ of type Ci for some

$1 \leq i \leq 4$. Resolution produces clause $C = C_2' \vee q_1 \vee \ldots \vee q_m$ of type Ci, and no replacement or splitting rules apply. We have $\{C_1, C_2\} \models_{\mathrm{p}} C$ and $\{\underline{C_1}, \underline{C_2}\} \models_{\mathrm{p}} \underline{C}$. Hence $\mathsf{I}(S) \supseteq \underline{C_1} \cup \mathsf{I}_i(\underline{C_2}) \models_{\mathrm{p}} \mathsf{I}_i(\underline{C})$. If $i = 1$ then the second statement of the lemma holds because $\mathsf{I}_i(\underline{C})$ contains a renaming of $\underline{C}$. If $i > 1$ then the first statement holds.

Now we consider binary resolution steps where no splitting literals are resolved upon. This is possible only when no negative splitting literals are present in the premises. Then the resolvent has no negative splitting literals. $\mathcal{Q}_0$ splitting may create negative splitting literals, but none of them are from $\mathcal{Q} \setminus \mathcal{Q}_0$. Hence the last part of the second statement of the lemma is always true. In the following $D, D_1, \ldots$ denote subsets of $\mathcal{Q}_0$. When we write $C \vee D$, it is implicit that C has no splitting literals. We have the following cases:

1. We do resolution between two clauses C_1 and C_2 from S, both of type C1, and the resolvent C is of type C1. Hence no splitting or replacement rules apply, $S' = S \cup \{C\}$, $\mathsf{I}(S) \supseteq \{\underline{C_1}[\mathbf{x}_{r+1}], \underline{C_2}[\mathbf{x}_{r+1}]\} \models_{\mathrm{p}} \underline{C}[\mathbf{x}_{r+1}]$. Hence the second statement holds.

2. We do resolution between a clause $C_1[\mathbf{x}_{r+1}] = C_1'[\mathbf{x}_{r+1}] \vee D_1 \vee \pm P(\mathbf{x}_{r+1})$, of type C1, and a clause $C_2[\mathbf{x}_{r+1}] = \mp P(t[\mathbf{x}_{r+1}]) \vee C_2'[\mathbf{x}_{r+1}] \vee D_2$, of type C2, both from S upto renaming, and the resolvent is $C[\mathbf{x}_{r+1}] = C_1'[t[\mathbf{x}_{r+1}]] \vee C_2'[\mathbf{x}_{r+1}] \vee D_1 \vee D_2$. By ordering constraints $t[\mathbf{x}_{r+1}] \in \mathsf{Ngr}_1$ is non-trivial. All literals in $C_1'[t[\mathbf{x}_{r+1}]] \vee C_2'[\mathbf{x}_{r+1}]$ are of the form $\pm' Q(t'[\mathbf{x}_{r+1}])$ with $t'[\mathbf{x}_{r+1}] \in \mathsf{Ngr}_1$. Hence no splitting or replacement rules apply and $S' = S \cup \{C\}$. $\underline{C_1}[\mathsf{Ngr}_1] \cup \{\underline{C_2}[\mathbf{x}_{r+1}]\} \models_{\mathrm{p}} \underline{C}[\mathbf{x}_{r+1}]$. Hence $\mathsf{I}(S) \supseteq \mathsf{I}_1(\underline{C_1}) \cup \mathsf{I}_2(\underline{C_2}) \supseteq \underline{C_1}[\mathsf{Ngr}_1] \cup \underline{C_2}[\mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]] \cup \{\underline{C_2}[\mathbf{x}_{r+1}]\} \cup \underline{C_2}[\mathsf{Ngrr}[\mathsf{G}_1]] \models_{\mathrm{p}} \{\underline{C}[\mathbf{x}_{r+1}]\} \cup \underline{C}[\mathsf{Ngrr}[\mathsf{G}_1]] = \mathsf{I}_2(\underline{C}[\mathbf{x}_{r+1}])$. If C_1' is non-empty or C_2' has some non-trivial literal then $C[\mathbf{x}_{r+1}]$ is of type C2, $S' \sqsubseteq S$ and the first statement holds. If C_1' is empty and C_2' has only trivial literals, then C is of type C1 and the second statement holds.

3. We do resolution between a clause $C_1[\mathbf{x}_{r+1}] = C_1'[\mathbf{x}_{r+1}] \vee D_1 \vee \pm P(\mathbf{x}_{r+1})$ of type C1, and a clause $C_2 = \mp P(t) \vee C_2' \vee D_2$ of type C3, both from S upto renaming, and the resolvent is $C = C_1'[t] \vee C_2' \vee D_1 \vee D_2$. We know that $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. Hence no splitting or replacement rules apply, and $S' = S \cup \{C\}$. $\{C_1[t], C_2\} \models_{\mathrm{p}} C$. Hence $\mathsf{I}(S) \supseteq \mathsf{I}_1(\underline{C_1}[\mathbf{x}_{r+1}]) \cup \mathsf{I}_3(\underline{C_2}) \supseteq \underline{C_1}[\mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]] \cup \{\underline{C_2}\} \models_{\mathrm{p}} \mathsf{I}_3(\underline{C}) = \{\underline{C}\}$. If C_1' or C_2' is non-empty. then $C[\mathbf{x}_{r+1}]$ is of type C3, $S' \sqsubseteq S$ and the first statement holds. If C_1' and C_2' are empty then C is of type C1 and the second statement holds.

4. We do resolution between a clause $C_1[\mathbf{x}_{r+1}] = C_1'[\mathbf{x}_{r+1}] \vee D_1 \vee \pm P(\mathbf{x}_{r+1})$ of type C1, and a clause $C_2[\mathbf{x}_1, \ldots, \mathbf{x}_r] = \mp P(x_1, \ldots, x_n) \vee C_2'[\mathbf{x}_1, \ldots, \mathbf{x}_r] \vee D_2$ of type C4, both from S upto renaming, and the resolvent is $C[\mathbf{x}_1, \ldots, \mathbf{x}_r] = C_1'[f(x_1, \ldots, x_n)] \vee C_2'[\mathbf{x}_1, \ldots, \mathbf{x}_r] \vee D_1 \vee D_2$. (By ordering constraints we have chosen a non trivial literal from C_2 for resolution). No splitting or replacement rules apply and $S' = S \cup \{C\}$. We have $\underline{C_1}[\mathsf{U}] \cup \{\underline{C_2}[\mathbf{x}_1, \ldots, \mathbf{x}_r]\} \supseteq \{\underline{C_1}[f(x_1, \ldots, x_n)], \underline{C_2}[\mathbf{x}_1, \ldots, \mathbf{x}_r]\} \models_{\mathrm{p}} \underline{C}[\mathbf{x}_1, \ldots, \mathbf{x}_r]$. Hence $\underline{C_1}[\mathsf{U}] \cup \pi(\underline{C_2}[\mathbf{x}_1, \ldots, \mathbf{x}_r]) \models_{\mathrm{p}} \pi(\underline{C}[\mathbf{x}_1, \ldots, \mathbf{x}_r])$ and $\underline{C_1}[\mathsf{U}[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] \cup \underline{C_2}[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]]) \models_{\mathrm{p}} \underline{C}[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]]$. Hence $\mathsf{I}(S) \supseteq \mathsf{I}_1(\underline{C_1}) \cup \mathsf{I}_4(\underline{C_2}) \models_{\mathrm{p}} \mathsf{I}_4(\underline{C})$.

- Suppose C_1' is non-empty or C_2' has some non-trivial literal. Then C is of type C4. The only trivial literals in $C[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ are those in $C_2'[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ and hence they involve predicates from $\mathbb{P}$. Hence $C[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ if of type C4 and the first statement holds.
- Suppose C_1' is empty and C_2' has only trivial literals. Then $C[\mathbf{x}_1, \ldots, \mathbf{x}_r] = B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}] \vee D_1 \vee D_2$ where $1 \leq i_1, \ldots, i_k \leq r$, and each B_i is an ϵ-block. The second statement holds.

5. We do resolution between a clause $C_1[\mathbf{x}_{r+1}] = C_1'[\mathbf{x}_{r+1}] \vee D_1 \vee \pm P(s[\mathbf{x}_{r+1}])$ and a clause $C_2[\mathbf{x}_{r+1}] = \mp P(t[\mathbf{x}_{r+1}]) \vee C_2'[\mathbf{x}_{r+1}] \vee D_2$, both of type C2, and both from S upto renaming, and the resolvent is $C[\mathbf{x}_{r+1}] = C_1'[\mathbf{x}_{r+1}]\sigma \vee C_2'[\mathbf{x}_{r+2}]\sigma \vee D_1 \vee D_2$ where $\sigma = mgu(s[\mathbf{x}_{r+1}], t[\mathbf{x}_{r+2}])$ (we renamed the second clause before resolution). We know that $s[\mathbf{x}_{r+1}], t[\mathbf{x}_{r+1}] \in \mathsf{Ngr}_1$, and by ordering constraints both s and t are non-trivial. By Lemma 2 one of the following cases holds:

   - $\mathbf{x}_{r+1}\sigma = \mathbf{x}_{r+2}\sigma = \mathbf{x}_{r+1}$. $C[\mathbf{x}_{r+1}] = C_1'[\mathbf{x}_{r+1}] \vee C_2'[\mathbf{x}_{r+1}]$. Hence no splitting or replacement rules apply and $S' = S \cup \{C\}$. We have $\{C_1[\mathbf{x}_{r+1}], C_2[\mathbf{x}_{r+1}]\} \models_{\mathrm{p}} C[\mathbf{x}_{r+1}]$. Hence $\mathsf{I}_2(\underline{C_1}[\mathbf{x}_{r+1}]) \cup \mathsf{I}_2(\underline{C_2}[\mathbf{x}_{r+1}]) \models_{\mathrm{p}} \mathsf{I}_2(\underline{C}[\mathbf{x}_{r+1}]) \ni \underline{C}[\mathbf{x}_{r+1}]$. If C_1' or C_2' contains some non-trivial literal then $C[\mathbf{x}_{r+1}]$ is of type C2 and the first condition holds. If C_1' and C_2' contain only trivial literals then C is of type C1 and the second condition holds.
   - $\mathbf{x}_{r+1}\sigma, \mathbf{x}_{r+2}\sigma \in \mathsf{Ngrr}[\mathsf{G}] \subseteq \mathsf{Ngrr}[\mathsf{G}_1]$. Then every literal in $C[\mathbf{x}_{r+1}]$ is of the form $\pm' Q(u)$ with $u \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. No splitting or replacement rules apply and $S' = S \cup \{C\}$. $\mathsf{I}(S) \supseteq \underline{C_1}[\mathsf{Ngrr}[\mathsf{G}_1]] \cup \underline{C_2}[\mathsf{Ngrr}[\mathsf{G}_1]] \models_{\mathrm{p}} \{\underline{C}\} = \mathsf{I}_3(\underline{C})$. If C_1' or C_2' is non-empty then C is of type C3 and the first statement holds. If C_1' and C_2' are empty then C is of type C1 and the second statement holds.

6. We do resolution between a clause $C_1[\mathbf{x}_{r+1}] = C_1'[\mathbf{x}_{r+1}] \vee D_1 \vee \pm P(s[\mathbf{x}_{r+1}])$ of type C2, and a ground clause $\mp P(t) \vee C_2' \vee D_2$ of type C3, both from S upto renaming, and the resolvent is $C = C_1'[\mathbf{x}_{r+1}]\sigma \vee C_2' \vee D_1 \vee D_2$ where σ is a unifier of $s[\mathbf{x}_{r+1}]$ and t. We know that $s[\mathbf{x}_{r+1}] \in \mathsf{Ngr}_1$, $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$, and by ordering constraints, s is non-trivial. We have the following cases:

   - $t \in \mathsf{G}_1$. Then $\mathbf{x}_{r+1}\sigma$ is a strict subterm of t hence $\mathbf{x}_{r+1}\sigma \in \mathsf{G} \subseteq \mathsf{Ngrr}[\mathsf{G}_1]$.
   - $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]] \setminus \mathsf{G}_1$. Hence we have $t = t_1[t']$ for some non-trivial $t_1[\mathbf{x}_{r+1}] \in \mathsf{Ngr}_1$ and some $t' \in \mathsf{Ngrr}[\mathsf{G}_1]$. Let $s' = \mathbf{x}_{r+1}\sigma$. As $s[s'] = t_1[t']$ hence $s[\mathbf{x}_{r+1}]$ and $t_1[\mathbf{x}_{r+2}]$ have a unifier $\sigma = \{\mathbf{x}_{r+1} \mapsto s', \mathbf{x}_{r+2} \mapsto t'\}$. From Lemma 2, one of the following is true:
     - $s[\mathbf{x}_{r+1}] = t_1[\mathbf{x}_{r+1}]$. Hence we have $\mathbf{x}_{r+1}\sigma = s' = t' \in \mathsf{Ngrr}[\mathsf{G}_1]$.
     - $\mathbf{x}_{r+1}\sigma_1, \mathbf{x}_{r+2}\sigma_1 \in \mathsf{Ngrr}[\mathsf{G}] \subseteq \mathsf{Ngrr}[\mathsf{G}_1]$. Hence $s' \in \mathsf{Ngrr}[\mathsf{G}_1]$.

   In each case we have $\mathbf{x}_{r+1}\sigma = s' \in \mathsf{Ngrr}[\mathsf{G}_1]$. Hence all literals in $C_1'[\mathbf{x}_{r+1}]\sigma$ are of the form $\pm Q(t)$ with $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. All literals in C_2' are of the form $\pm' Q(t)$ with $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. Hence no splitting or replacement rules apply and $S' = S \cup \{C\}$. $\mathsf{I}(S) \supseteq \mathsf{I}_2(\underline{C_1}[\mathbf{x}_{r+1}]) \cup \mathsf{I}_3(\underline{C_2}) \supseteq \underline{C_1}[\mathsf{Ngrr}[\mathsf{G}_1]] \cup \{\underline{C_2}\} \models_{\mathrm{p}} \{\underline{C}\} = \mathsf{I}_3(\underline{C})$. If C'_1 or C'_2 is non-empty then C is of type C3 and the first statement holds. If C'_1 and C'_2 are empty then C is of type C1 and the second statement holds.

7. We do resolution between a clause $C_1[\mathbf{x}_{r+1}] = C'_1[\mathbf{x}_{r+1}] \vee D_1 \vee \pm P(s[\mathbf{x}_{r+1}])$ of type C2, and a clause $C_2[\mathbf{x}_1, \ldots, \mathbf{x}_r] = \mp P(f(x_1, \ldots, x_n)) \vee C'_2[\mathbf{x}_1, \ldots, \mathbf{x}_r] \vee D_2$ of type C4, both from S upto renaming, and $\pm P(s[\mathbf{x}_{r+1}])$ and $\mp P(f(x_1, \ldots, x_n))$ are the literals resolved upon from the respective clauses. (By ordering constraints we have chosen a non-trivial literal to resolve upon in the second clause). By ordering constraints $s[\mathbf{x}_{r+1}] \in \mathsf{Ngr}_1$ is non-trivial. Hence we have the following two cases for $s[\mathbf{x}_{r+1}] = f(s_1[\mathbf{x}_{r+1}], \ldots, s_n[\mathbf{x}_{r+1}])$.

   - We have some $1 \leq i, j \leq n$ such that $x_i = x_j$ but $s_i[\mathbf{x}_{r+1}] \neq s_j[\mathbf{x}_{r+1}]$. By Lemma 3, the only possible unifier of the terms $s[\mathbf{x}_{r+1}]$ and $f(x_1, \ldots, x_n)$ is σ such that $\mathbf{x}_{r+1}\sigma = g$ is a ground subterm of s_i or s_j and $x_k\sigma = s_k[g]$ for $1 \leq k \leq n$. As $s[\mathbf{x}_{r+1}] \in \mathsf{Ngr}_1$, we have $g \in \mathsf{G}$ and each $s_k[\mathbf{x}_{r+1}] \in \mathsf{Ngrr} \cup \mathsf{G}$. Hence $\mathbf{x}_{r+1}\sigma \in \mathsf{G}$ and each $x_k\sigma \in \mathsf{Ngrr}[\mathsf{G}] \cup \mathsf{G} \subseteq \mathsf{Ngrr}[\mathsf{G}_1]$. The resolvent $C = C'_1[\mathbf{x}_{r+1}]\sigma \cup C'_2[\mathbf{x}_1, \ldots, \mathbf{x}_r]\sigma \vee D_1 \vee D_2$ is ground. Each literal in $C'_1[\mathbf{x}_{r+1}]\sigma$ is of the form $\pm' Q(t)$ with $t \in \mathsf{Ngr}_1[\mathsf{G}] \subseteq \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. Each literal in $C'_2[\mathbf{x}_1, \ldots, \mathbf{x}_r]\sigma$ is of the form $\pm' Q(t)$ where the following cases can arise:
     - $t = f'(x_{i_1}, \ldots, x_{i_m})\sigma$ such that $\{x_{i_1}, \ldots, x_{i_m}\} = \{x_1, \ldots, x_n\}$. Then $t = f'(s_{i_1}, \ldots, s_{i_m})[g] \in \mathsf{Ngr}_1[\mathsf{G}_1] \subseteq \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$.
     - $t = x_k\sigma \in \mathsf{Ngrr}[\mathsf{G}_1] \subseteq \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$ for some $1 \leq k \leq n$, where the literal $\pm' Q(x_k)$ is from C_2.

     We conclude that all non-splitting literals in C are of the form $\pm' Q(t)$ with $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$, and no splitting or replacement rules apply. We have $S' = S \cup \{C\}$. $\mathsf{I}(S) \supseteq \mathsf{I}_2(\underline{C_1}[\mathbf{x}_{r+1}]) \cup \mathsf{I}_4(\underline{C_2}[\mathbf{x}_1, \ldots, \mathbf{x}_r]) \supseteq \underline{C_1}[\mathsf{Ngrr}[\mathsf{G}_1]] \cup \underline{C_2}[\mathsf{Ngrr}[\mathsf{G}_1]] \models_{\mathrm{p}} \{C\} = \mathsf{I}_3(C)$. If C'_1 or C'_2 is non-empty then C is of type C3, and the first statement holds. If C'_1 and C'_2 are empty then C of type C1 and the second condition holds.

   - For all $1 \leq i, j \leq n$, if $x_i = x_j$ then $s_i[\mathbf{x}_{r+1}] = s_j[\mathbf{x}_{r+1}]$. Then $s[\mathbf{x}_{r+1}]$ and $f(x_1, \ldots, x_n)$ have mgu σ such that $x_k\sigma = s_k[\mathbf{x}_{r+1}] \in \mathsf{Ngrr} \cup \mathsf{G}$ for $1 \leq k \leq n$ and $x\sigma = x$ for $x \notin \{x_1, \ldots, x_n\}$. The resolvent $C[\mathbf{x}_{r+1}] = C'_1[\mathbf{x}_{r+1}] \vee C'_2\sigma \vee D_1 \vee D_2$ is a one-variable clause. $\{C_1[\mathbf{x}_{r+1}]\} \cup C_2[\mathsf{Ngrr} \cup \mathsf{G}] \models_{\mathrm{p}} C[\mathbf{x}_{r+1}]$. All literals in $C'_1[\mathbf{x}_{r+1}]$ are of the form $\pm' Q(t)$ with $t \in \mathsf{Ngr}_1$, and no replacement rules apply on them. All literals in $C'_2[\mathbf{x}_1, \ldots, \mathbf{x}_r]\sigma$ are of the form $\pm' Q(t[\mathbf{x}_{r+1}])$ where the following cases can arise:
     - $t[\mathbf{x}_{r+1}] = f'(x_{i_1}, \ldots, x_{i_m})\sigma$ such that $\{x_{i_1}, \ldots, x_{i_m}\} = \{x_1, \ldots, x_n\}$. Then $t[\mathbf{x}_{r+1}] \in \mathsf{Ngr}_1$. No replacement rules apply on such a literal.
     - $t[\mathbf{x}_{r+1}] = x_k\sigma = s_k[\mathbf{x}_{r+1}] \in \mathsf{Ngrr}$ for some $1 \leq k \leq n$, where the literal $\pm' Q(x_k)$ is from C_2. Hence we must have $Q \in \mathbb{P}$. Let $s_k[\mathbf{x}_{r+1}] = t_1[\ldots[t_p[\mathbf{x}_{r+1}]]\ldots]$ for some $p \geq 0$ where each $t_i[\mathbf{x}_{r+1}] \in \mathsf{Ngr}$ is

non-trivial and reduced. Such a literal is replaced by the literal $\pm' Qt_1 \dots t_{p-1}(t_p[\mathbf{x}_{r+1}])$ and we know that $t_p \in \mathsf{Ngr} \subseteq \mathsf{Ngr}_1$. This new clause is obtained by propositional resolution between the former clause and clauses from $cl(\mathcal{R})[\mathsf{Ngrr}]$.

– $t[\mathbf{x}_{r+1}] = x_k\sigma = s_k \in \mathsf{G}$ for some $1 \leq k \leq n$, where the literal $\pm' Q(x_k)$ is from C_2. Hence we must have $Q \in \mathbb{P}$. No replacement rules apply on such a literal. If C contains only ground literals then this literal is left unchanged. Otherwise we perform $\mathcal{Q}_0$-splitting and this literal is replaced by the literal $-\overline{\pm' Q(s_k)}$ and also a new clause $C'' = \overline{\pm' Q(s_k)} \vee \pm' Q(s_k)$ of type C3 is added to S. If C' is the new clause obtained by this splitting then $\underline{C'}$ is clearly propositionally equivalent to the former clause. Also $\underline{C''} = \mp' Q(s_k) \vee \pm' Q(s_k)$ is a propositionally valid statement.

We conclude that after zero or more replacement and splitting rules, we obtain a clause $C'[\mathbf{x}_{r+1}]$, together with a set S'' of clauses of type C3, $\{\underline{C}[\mathbf{x}_{r+1}]\} \cup cl(\mathcal{R})[\mathsf{Ngrr}] \models_{\mathrm{p}} \{\underline{C}[\mathbf{x}_{r+1}]\}$, $\emptyset \models_{\mathrm{p}} \underline{S''}$, and $S' = S \cup \{C'\} \cup S''$. $\{\underline{C_1}[\mathbf{x}_{r+1}]\} \cup \underline{C_2}[\mathsf{Ngrr} \cup \mathsf{G}] \cup cl(\mathcal{R})[\mathsf{Ngrr}] \models_{\mathrm{p}} \underline{C'}[\mathbf{x}_{r+1}]$. Hence $\mathsf{I}(S) \supseteq \mathsf{I}_2(\underline{C_1}) \cup \mathsf{I}_4(\underline{C_2}) \supseteq \{\underline{C_1}[\mathbf{x}_{r+1}]\} \cup \underline{C_1}[\mathsf{Ngrr}[\mathsf{G}_1]] \cup \underline{C_2}[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] \cup cl(\mathcal{R})[\mathsf{Ngrr}] \cup cl(\mathcal{R})[\mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] \models_{\mathrm{p}} \mathsf{I}_2(\underline{C'}) \cup \mathsf{I}_3(\underline{S''}) = \underline{C'}[\mathbf{x}_{r+1}] \cup \underline{C'}[\mathsf{Ngrr}[\mathsf{G}_1]] \cup \underline{S''}$. If C' is of type C2 or C3 then the first statement holds. Otherwise C' is of type C1 and the second statement holds.

8. We do resolution between a clause $C_1 = C_1' \vee D_1 \vee \pm P(s)$ and a clause $C_2 = \mp P(s) \vee C_2' \vee D_2$, both ground clauses of type C3 from S, and the resolvent is $C = C_1' \vee C_2' \vee D_1 \vee D_2$. No replacement or splitting rules apply and we have $S' = S \cup \{C\}$. $\mathsf{I}(S) \supseteq \{\underline{C_1}, \underline{C_2}\} \models_{\mathrm{p}} \mathsf{I}_3(\underline{C}) = \{\underline{C}\}$. If C_1' or C_2' is non-empty then C is of type C3, and the first statement holds. If C_1' and C_2' are empty then C is of type C1 and the second statement holds.

9. We do resolution between a ground clause $C_1 = C_1' \vee D_1 \vee \pm P(s)$ of type C3, and a clause $C_2[\mathbf{x}_1, \dots, \mathbf{x}_r] = \mp P(f(x_1, \dots, x_n)) \vee C_2'[\mathbf{x}_1, \dots, \mathbf{x}_r] \vee D_2$ of type C4, both from S upto renaming, and $\pm P(s)$ and $\mp P(f(x_1, \dots, x_n))$ are the literals resolved upon from the respective clauses. We know that $s \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. Hence we have the following two cases for s.

   - $s \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]] \setminus \mathsf{G}_1$. Hence s must be of the form $f(s_1, \dots, s_n)[g]$ for some $f(s_1, \dots, s_n) \in \mathsf{Ngr}_1$ and some $g \in \mathsf{Ngrr}[\mathsf{G}_1]$ (The symbol f is same as in the literal $\mp P(f(x_1, \dots, x_n))$ otherwise this resolution step would not be possible). We have each $s_i \in \mathsf{Ngrr} \cup \mathsf{G}$. The mgu σ of s and $f(x_1, \dots, x_n)$ is such that $x_i\sigma = s_i[g] \in \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]$. The resolvent $C = C_1' \vee C_2'[\mathbf{x}_1, \dots, \mathbf{x}_r]\sigma \vee D_1 \vee D_2$ is a ground clause. All literals in C_1' are of the form $\pm' Q(t)$ with $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$ hence no replacement rules apply on them. The literals in $C_2'[\mathbf{x}_1, \dots, \mathbf{x}_r]\sigma$ are of the form $\pm' Q(t)$ where the following cases are possible:
     – $t = f'(x_{i_1}, \dots, x_{i_m})\sigma$ where $\{x_{i_1}, \dots, x_{i_m}\} = \{x_1, \dots, x_n\}$. Then $f'(s_{i_1}, \dots, s_{i_m}) \in \mathsf{Ngr}_1$. Hence $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. No replacement rules apply on such a literal.

– $t = x_i\sigma \in \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]$ for some $1 \leq i \leq n$. If $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$ then no replacement rules apply on this literal. Otherwise suppose $t \in \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]] \backslash \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. We have $t = t_1[\ldots[t_p[t']]\ldots]$ for some reduced non-trivial non-ground terms $t_1, \ldots, t_p \in \mathsf{Ngr}$ with $p \geq 0$ such that $t_1[\ldots[t_p[y]]] \in \mathsf{Ngrr}$ and $t' \in \mathsf{Ngrr}[\mathsf{G}_1]$, and the replacement strategy replaces this literal by the literal $\pm' Q t_1 \ldots t_{p-1}(t_p[t'])$, and we know that $t_p \in \mathsf{Ngr} \subseteq \mathsf{Ngr}_1$ so that $t_p[t'] \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. This new clause can be obtained by propositional resolution between the former clause and clauses from $cl(\mathcal{R})[\mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]]$

We conclude that after zero or more replacement rules, we obtain a ground clause C', all of whose non-splitting literals are of the form $\pm' Q(t)$ with $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$, and which is obtained by propositional resolution from $\{C\} \cup cl(\mathcal{R})[\mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]]$. No splitting rules apply and $S' = S \cup \{C'\}$. $\{\underline{C_1}\} \cup \underline{C_2}[\mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] \models_{\mathrm{p}} \underline{C}$ hence $\mathsf{I}(S) \supseteq \mathsf{I}_3(\underline{C_1}) \cup \mathsf{I}_4(\underline{C_2}) \cup cl(\mathcal{R})[\mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] \models_{\mathrm{p}} \mathsf{I}_3(\underline{C'}) = \{\underline{C'}\}$. If C'_1 or C'_2 is non-empty then C is of type C3, and the first statement holds. If C'_1 and C'_2 are empty then C is of type C1 and the second statement holds.

- $s \in \mathsf{G}_1$. For the resolution step to be possible we must have $s = f(s_1, \ldots, s_n)$. Each $s_i \in \mathsf{G}$. The mgu σ of s and $f(x_1, \ldots, x_n)$ is such that each $x_i\sigma = s_i$. The resolvent $C = C'_1 \vee C'_2[\mathbf{x}_1, \ldots, \mathbf{x}_r]\sigma \vee D_1 \vee D_2$ is a ground clause. All literals in C'_1 are of the form $\pm' Q(t)$ with $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. The literals in $C'_2[\mathbf{x}_1, \ldots, \mathbf{x}_r]\sigma$ are of the form $\pm' Q(t)$ where the following cases are possible:

  – $t = f'(x_{i_1}, \ldots, x_{i_m})\sigma$ where $\{x_{i_1}, \ldots, x_{i_m}\} = \{x_1, \ldots, x_n\}$. Then $t = f'(s_{i_1}, \ldots, s_{i_m}) \in \mathsf{G}_1 \subseteq \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$.

  – $t = x_i\sigma = s_i \in \mathsf{G} \subseteq \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$ for some $1 \leq i \leq n$.

  Hence all non-splitting literals in C are of the form $\pm' Q(t)$ with $t \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. No replacement rules or splitting rules apply and $S' = S \cup \{C\}$. $\{C_1\} \cup C_2[\mathsf{G}] \models_{\mathrm{p}} C$ hence $\mathsf{I}(S) \models_{\mathrm{p}} \mathsf{I}_3(\underline{C}) = \{\underline{C}\}$. If C'_1 or C'_2 is non-empty then C is of type C3 and the first statement holds. If C'_1 and C'_2 are empty then C is of type C1 and the second statement holds.

10. We do resolution between two clauses $C_1[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ and $C_2[\mathbf{x}_1, \ldots, \mathbf{x}_r]$, both of type C4, and both from S upto renaming. First we rename the second clause as $C_2[\mathbf{x}_{r+1}, \ldots, \mathbf{x}_{2r}]$ by applying the renaming $\sigma_0 = \{\mathbf{x}_1 \mapsto \mathbf{x}_{r+1}, \ldots, \mathbf{x}_r \mapsto \mathbf{x}_{2r}\}$. By ordering constraints, $C_1[\mathbf{x}_1, \ldots, \mathbf{x}_r] = C'_1[\mathbf{x}_1, \ldots, \mathbf{x}_r] \vee D_1 \vee P(f(x_1, \ldots, x_n))$ and $C_2[\mathbf{x}_{r+1}, \ldots, \mathbf{x}_{2r}] = -P(f(y_1, \ldots, y_n)) \vee C'_2[\mathbf{x}_{r+1}, \ldots, \mathbf{x}_{2r}] \vee D_2$ and the resolvent is $C[\mathbf{x}_1, \ldots, \mathbf{x}_r] = C'_1[\mathbf{x}_1, \ldots, \mathbf{x}_r]\sigma \vee C'_2[\mathbf{x}_{r+1}, \ldots, \mathbf{x}_{2r}]\sigma \vee D_1 \vee D_2$ where, by Lemma 6, σ is such that $\{x_1, \ldots, x_n\}\sigma \subseteq \{x_1, \ldots, x_n\}$ and $y_i\sigma = x_i$ for $1 \leq i \leq n$. $\pi(C_1) \cup \pi(C_2) \models_{\mathrm{p}} C[\mathbf{x}_1, \ldots, \mathbf{x}_r]$. Hence $\mathsf{I}(S) \supseteq \mathsf{I}_4(\underline{C_1}[\mathbf{x}_1, \ldots, \mathbf{x}_r]) \cup \mathsf{I}_4(\underline{C_2}[\mathbf{x}_1, \ldots, \mathbf{x}_r]) = \pi(\underline{C_1}[\mathbf{x}_1, \ldots, \mathbf{x}_r]) \cup \underline{C_1}[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] \cup \pi(\underline{C_2}[\mathbf{x}_1, \ldots, \mathbf{x}_r]) \cup \underline{C_2}[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] \models_{\mathrm{p}} \pi(\underline{C}[\mathbf{x}_1, \ldots, \mathbf{x}_r]) \cup \underline{C}[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] = \mathsf{I}_4(\underline{C}[\mathbf{x}_1, \ldots, \mathbf{x}_r])$.

    - Suppose C'_1 or C'_2 has a non-trivial literal. Then C is of type C4, no replacement or splitting rules apply, $S' = S \cup \{C\}$ and the first statement holds.

- Suppose C'_1 and C'_2 contain no non-trivial literal. Then $C[\mathbf{x}_1, \ldots, \mathbf{x}_r] = B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}] \vee D_1 \vee D_2$ with $1 \leq i_1, \ldots, i_k \leq r$, each B_i being an ϵ-block. No splitting or replacement rules apply (ϵ-splitting is forbidden by ϕ_0), and $S' = S \cup \{C\}$. The second statement holds.

11. We do a resolution step in which one of the premises is a clause from $cl(\mathcal{R})$. Every clause in $cl(\mathcal{R})$ is of type C2. Also trivially $\mathsf{I}_2(C) \subseteq \mathsf{I}(T)$. Hence this case can be dealt with in the same way as in the case where one of the premises of resolution is a clause of type C2.

Next we consider factoring steps. Factoring on a clause of type C1 or C3 is possible only if the two involved literals are the same, hence this is equivalent to doing nothing.

1. We do factoring on a clause $C_1[\mathbf{x}_{r+1}] = C'_1[\mathbf{x}_{r+1}] \vee \pm P(s[\mathbf{x}_{r+1}]) \vee \pm P(t[\mathbf{x}_{r+1}])$ of type C2, and from S upto renaming. We know that $s[\mathbf{x}_{r+1}], t[\mathbf{x}_{r+1}] \in \mathsf{Ngr}_1$, and by ordering constraints s and t are non trivial. The clause obtained is $C[\mathbf{x}_{r+1}] = C'_1[\mathbf{x}_{r+1}]\sigma \vee \pm P(s[\mathbf{x}_{r+1}])\sigma$ where σ is a unifier of $s[\mathbf{x}_{r+1}]$ and $t[\mathbf{x}_{r+1}]$. If $s[\mathbf{x}_{r+1}] \neq t[\mathbf{x}_{r+1}]$ then by Lemma 3 $\mathbf{x}_{r+1}\sigma$ is a ground strict subterm of s or t, hence $\mathbf{x}_{r+1}\sigma \in \mathsf{G} \subseteq \mathsf{Ngrr}[\mathsf{G}_1]$. Each literal in C is of the form $\pm' Q(t')$ where $t' \in \mathsf{Ngr}_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. Hence C is of type C3. No splitting or replacement rules apply and $S' = S \cup \{C\}$. We have $C \in C_1[\mathsf{Ngrr}[\mathsf{G}_1]]$. $\mathsf{I}(S) \supseteq \mathsf{I}_2(\underline{C_1}[\mathbf{x}_{r+1}]) \supseteq \underline{C_1}[\mathbf{x}_{r+1}][\mathsf{Ngrr}[\mathsf{G}_1]] \supseteq \mathsf{I}_3(C) = \{\underline{C}\}$. The first statement holds.

2. We do factoring on a clause $C_1[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ of type C4, and from S upto renaming, to obtain the clause $C[\mathbf{x}_1, \ldots, \mathbf{x}_r]$. By ordering constraints non-trivial literals must be chosen for factoring. Then $C[\mathbf{x}_1, \ldots, \mathbf{x}_r]$ is again of type C4 and $C[\mathbf{x}_1, \ldots, \mathbf{x}_r] \in \pi(C_1)$. $\mathsf{I}(S) \supseteq \mathsf{I}_4(\underline{C_1}) = \pi(\underline{C_1}) \cup \underline{C_1}[\mathsf{Ngrr} \cup \mathsf{Ngrr}[\mathsf{Ngrr}[\mathsf{G}_1]]] \models_{\mathrm{p}} \mathsf{I}_4(\underline{C})$. The first statement holds. $\square$

The alternative resolution procedure for testing unsatisfiability by using succinct representations of tableaux is now defined by the rule: $\mathcal{T} \mid S \blacktriangleright \mathcal{T} \mid S \cup \{B_1 \sqcup D\} \mid S \cup \{B_2\} \mid \ldots \mid S \cup \{B_k\}$ whenever $\mathsf{I}(S) \models_{\mathrm{p}} B_1 \sqcup \ldots \sqcup B_k \sqcup \underline{D}$, each B_i is an ϵ-block, $1 \leq i_1, \ldots, i_k \leq r$ and $D \subseteq \pm\mathcal{Q}$. The simulation property now states:

Lemma 1 *If $S \sqsubseteq T$ and $S \Rrightarrow_{\prec_s,\phi,\mathcal{R}} \mathcal{T}$ then $T \blacktriangleright^* \mathcal{T}'$ for some $\mathcal{T}'$ such that $\mathcal{T} \sqsubseteq \mathcal{T}'$.*

Proof: As $S \Rrightarrow_{\prec_s,\phi,\mathcal{R}} \mathcal{T}$, we have some S' such that $S \Rrightarrow_{\prec_s,\phi_0,\mathcal{R}} S'$ and $\mathcal{T}$ is obtained from S' by ϵ-splitting steps. From Lemma 10, one of the following cases holds.

- $S' \sqsubseteq S$. Then S' contains only clauses of type C1-C4 and no ϵ-splitting is applicable. Hence $\mathcal{T} = S' \sqsubseteq S$. As $\mathcal{T} \sqsubseteq S$ and $S \sqsubseteq T$ hence $\mathcal{T} \sqsubseteq T$ because of transitivity of $\sqsubseteq$. Thus T is the required $\mathcal{T}'$.

- $S' = S \cup \{C\} \cup S''$, C is a renaming of $B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}] \sqcup D$ where each B_i is an ϵ-block, $1 \leq i_1, \ldots, i_k \leq r$, $D \subseteq \pm\mathcal{Q}$, $\mathsf{I}(S) \models_{\mathrm{p}} \underline{C}$ and S'' is a set of clauses of type C3 and $\emptyset \models_{\mathrm{p}} \underline{S''}$. We have $\mathcal{T} = S \cup S'' \cup \{B_1 \sqcup D\} \mid$

$S \cup S'' \cup \{B_2\} \mid \ldots \mid S \cup S'' \cup \{B_k\}$. We have $S \cup S'' \cup \{B_1 \sqcup D\} \sqsubseteq T \cup \{B_1 \sqcup D\}$ and $S \cup S'' \cup \{B_i\} \sqsubseteq T \cup \{B_i\}$ for $1 \leq i \leq k$. We show that the required $\mathcal{T}'$ is $T \cup \{B_1 \sqcup D\} \mid T \cup \{B_1\} \mid \ldots \mid S \cup S'' \cup \{B_k\}$. As $S \sqsubseteq T$ hence $\mathsf{I}(T) \models_{\mathrm{p}} \mathsf{I}(S) \models_{\mathrm{p}} \underline{C}$. Hence $T \blacktriangleright \mathcal{T}'$. $\square$

Hence as for flat clauses we obtain:

Theorem 5 *Satisfiability for the class $\mathcal{C}$ is NEXPTIME-complete.*

Proof: Let S be a finite set in $\mathcal{C}$ whose satisfiability we want to show. We proceed as in the proof of Theorem 3. Wlog if $C \in S$ then C is either a complex clause or a one-variable clause. Clearly S is satisfiable iff $S \cup cl(\mathcal{R})$ is satisfiable. At the beginning we apply the replacement steps using $\mathcal{R}$ as long as possible and then $\mathcal{Q}_0$-splitting as long as possible. Hence wlog all clauses in S are of type C1-C4. Then we non-deterministically add a certain number of clauses of type C1 to S. Then we check that the resulting set S' does not contain $\square$, and is saturated in the sense that: if $C = B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}] \sqcup D$, each B_i is an ϵ-block, $1 \leq i_1, \ldots, i_k \leq r$, $D \subseteq \pm\mathcal{Q}_0$, and $B_j[\mathbf{x}_{r+1}] \notin S'$ for $1 \leq j \leq k$, then $\mathsf{I}(S') \not\models_{\mathrm{p}} \underline{C}$. There are exponentially many such C to check for since the number of splitting literals in polynomially many. The size of $\mathsf{I}(S')$ is exponential. $\square$

7 The Horn Case

We show that in the Horn case, the upper bound can be improved to DEXPTIME. The essential idea is that propositional satisfiability of Horn clauses is in PTIME instead of NPTIME. But now we need to eliminate the use of tableaux altogether. To this end, we replace the ϵ-splitting rule of Section 6 by splitting-with-naming. Accordingly we instantiate the set $\mathcal{Q}$ used in Section 6 as $\mathcal{Q} = \mathcal{Q}_0 \cup \mathcal{Q}_1$ where $\mathcal{Q}_1 = \{\overline{C} \mid C$ is a non-empty negative ϵ-block with predicates from $\mathbb{P}\}$. We know that binary resolution and factorization on Horn clauses produces Horn clauses. Replacements on Horn clauses using the rules from $\mathcal{R}$ produces Horn clauses. $\mathcal{Q}_1$-splitting on Horn clauses produces Horn clauses. E.g. clause $P(\mathbf{x}_1) \vee -Q(\mathbf{x}_1) \vee -R(\mathbf{x}_2)$ produces $P(\mathbf{x}_1) \vee -Q(\mathbf{x}_1) \vee -\overline{-R(\mathbf{x}_2)}$ and $\overline{-R(\mathbf{x}_2)} \vee -R(\mathbf{x}_2)$. $\mathcal{Q}_0$-splitting on $P(f(x)) \vee -Q(a)$ produces $P(f(\mathbf{x}_1)) \vee -\overline{-Q(a)}$ and $\overline{-Q(a)} \vee -Q(a)$ which are Horn. However $\mathcal{Q}_0$-splitting on $C = -P(f(\mathbf{x}_1)) \vee Q(a)$ produces $C_1 = -P(f(\mathbf{x}_1)) \vee -\overline{Q(a)}$ and $C_2 = \overline{Q(a)} \vee Q(a)$. C_2 is not Horn. However $\underline{C_1} = C$ and $\underline{C_2} = -Q(a) \vee Q(a)$ are Horn. Finally, as $\mathcal{Q}_1$ has exponentially many atoms, we must restrict their occurrences in clauses. Accordingly, for $1 \leq i \leq 4$, define clauses of type Ci' to be clauses C of the type Ci, such that $\underline{C}$ is Horn and has at most r negative literals from $\mathcal{Q}_1$. ($\underline{C}$ is defined as before, hence it leaves atoms from $\mathcal{Q}_1$ unchanged). Now the $\mathcal{Q}$-splitting-replacement strategy ϕ_h first applies the replacement steps of Section 6 as long as possible, then applies $\mathcal{Q}_0$-splitting as long as possible and then applies $\mathcal{Q}_1$-splitting as long as possible. Succinct representations are now defined as: $S \sqsubseteq_h T$ iff for each $C \in S$, C is of type Ci' and satisfies Pi_T for some $1 \leq i \leq 4$. The abstract resolution procedure is defined as: $T \blacktriangleright_h T \cup \{B_1 \vee -q_2 \vee \ldots \vee -q_k \sqcup D \sqcup E\} \cup \{\overline{B_i} \vee B_i \mid 2 \leq i \leq k\}$ whenever $\mathsf{I}(T) \models_{\mathrm{p}} \underline{C}$, $C = B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}] \sqcup D \sqcup E$, $\underline{C}$ is Horn, $1 \leq i_1, \ldots, i_k \leq r$, B_1

is an ϵ-block, B_i is a negative ϵ-block and $2 \leq i \leq k$, $D \subseteq \pm\mathcal{Q}_0$ and $E \subseteq \pm\mathcal{Q}_1$ such that if $k = 1$ then E has at most r negative literals, and if $k > 1$ then E has no negative literal. The $\sqsubseteq$ and $\blacktriangleright$ relations are as in Section 6.

Lemma 2 *If $S \sqsubseteq_h T$ and $S \Rrightarrow_{\prec_s,\phi_h,\mathcal{R}} S_1$ then $T \blacktriangleright_h^* T_1$ and $S_1 \sqsubseteq_h T_1$ for some T_1.*

Proof: Let ϕ_0 be as in Section 6. As $S \Rrightarrow_{\prec_s,\phi_h,\mathcal{R}} S_1$ hence we have some S' such that $S \Rrightarrow_{\prec_s,\phi_0,\mathcal{R}} S'$ and S_1 is obtained from S' by applying $\mathcal{Q}_1$-splitting steps. As discussed above, all clauses $C \in S_1 \cup S'$ are such that $\underline{C}$ is also Horn. If S' is obtained by resolving upon splitting literals, then one of the premises must be just a positive splitting literal. The other premise has at most r literals of the form $-q$ with $q \in \mathcal{Q}_1$, hence the resolvent has at most r literals of the form $-q$ with $q \in \mathcal{Q}_1$. In case non-splitting literals are resolved upon then the premises cannot have any negative splitting literal and the resolvent has no negative splitting literal. $\mathcal{Q}_0$-splitting does not create literals from $\pm\mathcal{Q}_1$. Hence all clauses in S' have at most r literals of the form $-q$ with $q \in \mathcal{Q}_1$. Now by Lemma 10, one of the following conditions holds.

- $S' \sqsubseteq S$. Then $\mathcal{Q}_1$-splitting is not applicable on clauses in S' and $S_1 = S' \sqsubseteq S$. From transitivity of $\sqsubseteq$ we have $S_1 \sqsubseteq T$. Then from the above discussion we conclude that $S_1 \sqsubseteq_h T$.

- $S' = S \cup \{C\} \cup S''$, C is a renaming of $B_1[\mathbf{x}_{i_1}] \sqcup \ldots \sqcup B_k[\mathbf{x}_{i_k}] \sqcup D$, each B_i is an ϵ-block, $1 \leq i_1, \ldots, i_k \leq r$, $D \subseteq \pm\mathcal{Q}$, $\mathsf{I}(S) \vDash_{\mathrm{p}} \underline{C}$, and S'' is a set of clauses of type C3 and $\emptyset \vDash_{\mathrm{p}} \underline{S''}$. Also if $k \geq 2$ then D has no literals $-q$ with $q \in \mathcal{Q}_1$. As C is Horn, wlog B_i is negative for $i \geq 2$. Hence $S_1 = S' \cup \{B_1 \vee -q_2 \vee \ldots \vee -q_k \sqcup D\} \cup \{\overline{B_i} \cup B_i \mid 2 \leq i \leq k\}$. We show that the required T_1 is $T \cup \{B_1 \vee -q_2 \vee \ldots \vee -q_k \sqcup D\} \cup \{\overline{B_i} \cup B_i \mid 2 \leq i \leq k\}$. Each $\overline{B_i} \cup B_i$ is of type C1'. As $C \in S'$ hence D has at most r literals $-q$ with $q \in \mathcal{Q}_1$. Hence if $k = 1$ then $B_1 \vee -q_2 \vee \ldots \vee -q_k \sqcup D$ is also of type C1'. If $k \geq 2$ then D has no negative literals $-q$ with $q \in \mathcal{Q}_1$, and $B_1 \vee -q_2 \vee \ldots \vee -q_k \sqcup D$ is again of type C1' since $k \leq r$. As $S \sqsubseteq_h T$ we have $\mathsf{I}(T) \vDash_{\mathrm{p}} \mathsf{I}(S) \vDash_{\mathrm{p}} \underline{C}$. Hence $T \blacktriangleright_h T_1$. Finally, clearly $S_1 \sqsubseteq T_1$ hence $S_1 \sqsubseteq_h T_1$. $\square$

Now for deciding satisfiability of a set of flat and one-variable clauses we proceed as in the non-Horn case. But now instead of non-deterministically adding clauses, we compute a sequence $S = S_0 \blacktriangleright_h S_1 \blacktriangleright_h S_2 \ldots$ starting from the given set S, and proceeding don't care non-deterministically, till no more clauses can be added, and then check whether $\square$ has been generated. The length of this sequence is at most exponential. Computing S_{i+1} from S_i requires at most exponential time because the number of possibilities for C in the definition of $\blacktriangleright$ above is exponential. (Note that this idea of $\mathcal{Q}_1$-splitting would not have helped in the non-Horn case because we cannot bound the number of positive splitting literals in a clause in the non-Horn case, whereas Horn clauses by definition have at most one positive literal). Also note that APDS can be encoded using flat Horn clauses. Hence:

Theorem 6 *Satisfiability for the classes $\mathcal{C}Horn$ and $\mathcal{F}Horn$ is DEXPTIME-complete.*

Together with Theorem 1, this gives us optimal complexity for protocol verification:

Theorem 7 *Secrecy of cryptographic protocols with single blind copying, with bounded number of nonces but unbounded number of sessions is DEXPTIME-complete.*

7.1 Alternative Normalization Procedure

While Theorem 6 gives us the optimum complexity for the Horn case, we outline here an alternative normalization procedure for deciding satisfiability in the Horn case, in the style of [14]. Our goal is to show that the Horn case can be dealt with using simpler techniques. This may also be interesting for implementations, since it avoids exhaustive generation of instantiations of clauses. Since we already have the optimum complexity from Theorem 6, we restrict ourselves to giving only the important ideas here. Define *normal* clauses to be clauses which have no function symbol in the body, have no repetition of variables in the body, and have no variables in the body other than those in the head. Sets of normal definite clauses involving unary predicates can be thought of as generalizations of tree automata, by adopting the convention that term t is *accepted* at *state* P iff atom $P(t)$ is reachable. I.e. states are just unary predicates. *(Intersection-)emptiness* and *membership* properties are defined as usual.

Lemma 3 *Emptiness and membership properties are decidable in polynomial time for sets of normal definite clauses.*

Proof: Let S be the set of clauses. To test emptiness of a state P, we remove arguments of predicate symbols in clauses, and treat predicates as proposition symbols. Then we add the clause $-P$ and check satisfiability of the resulting propositional Horn clause set.

To test if t is accepted at P, let T be the set of subterms of t. Define a set S' of clauses as follows. If $Q(s) \vee -Q_1(x_1) \vee \ldots \vee -Q_n(x_n) \in S$ and $s\sigma \in T$ for some substitution σ then we add the Horn clause $Q(s\sigma) \vee -Q_1(x_1\sigma) \vee \ldots \vee -Q_n(x_n\sigma)$ to S'. Finally we add $-P(t)$ to S' and test its unsatisfiability. S' is computable in polynomial time. Also S' has only ground clauses, hence satisfiability is equivalent to propositional unsatisfiability, by treating each ground literal as a propositional symbol. $\square$

The intuition behind the normalization procedure is as follows. We use new states which are sets $\{P_1, P_2, \ldots,\}$, where $P_1, P_2, \ldots$ are states in the given clauses set. The state $\{P_1, P_2, \ldots,\}$ represents intersection of the states $P_1, P_2, \ldots$. These new states are denoted by $p, q, p_1, \ldots$. The states P in clauses are replaced by $\{P\}$. We try to make non-normal clauses redundant by resolving them with normal clauses. Hence a clause $C \vee -p(t)$, where t has some function symbol, is resolved with a normal clause $p(s) \vee D$ to obtain a clause $C\sigma \vee D\sigma$ where $\sigma = mgu(s,t)$. Normal clauses $p(s) \vee C$ and $p(t) \vee D$ are used to produce clause $(p \cup q)(s\sigma) \vee C\sigma \vee D\sigma$ where $\sigma = mgu(s,t)$. In this process if we get a clause $C \vee -p(t)$ where t is ground, then either t is accepted at p using the normal clauses and we remove the literal $-p(t)$ from the clause, or t is not accepted at p using the normal clauses, and we reject the clause.

From clauses $C \vee -p(x) \vee -q(x)$ we derive the clause $C \vee -(p \vee q)(x)$. If a clause $p(x_1) \vee -q(x_1) \vee -q_1(x_2) \vee \ldots \vee -q_n(x_n)$ is produced where the x_i are mutually distinct, then either each q_i is non-empty using the normal clauses and we replace this clause by $p(x) \vee -q(x)$, or we reject this clause. The normal clauses $p(x) \vee -q(x)$ and $q(t) \vee C$ produce the clause $q(t) \vee C$. Replacement rules are also applied as in the non-Horn case. We continue this till no more new clauses can be produced. Then we remove all non-normal clauses. We claim that this process takes exponential time and each state p in the resulting clause set accepts exactly the terms accepted by each $P \in p$ in the original clause set. This also gives us a DEXPTIME algorithm for the satisfiability problem for the class $\mathcal{C}$.

Example 1 *Consider the set $S = \{C_1, \ldots, C_5\}$ of clauses where*

$$\begin{array}{lrl}
C_1 = & P(a) & \\
C_2 = & Q(a) & \\
C_3 = & P(f(g(\mathbf{x}_1, a), g(a, \mathbf{x}_1), a)) & \vee - P(\mathbf{x}_1) \\
C_4 = & P(f(g(\mathbf{x}_1, a), g(a, \mathbf{x}_1), b)) & \vee - P(\mathbf{x}_1) \\
C_5 = & R(\mathbf{x}_1) & \vee - P(f(\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_2)) \vee -Q(\mathbf{x}_2)
\end{array}$$

We first get the following normal clauses.

$$\begin{array}{lrl}
C_1' = & \{P\}(a) & \\
C_2' = & \{Q\}(a) & \\
C_3' = & \{P\}(f(g(\mathbf{x}_1, a), g(a, \mathbf{x}_1), a)) & \vee - \{P\}(\mathbf{x}_1) \\
C_4' = & \{P\}(f(g(\mathbf{x}_1, b), g(a, \mathbf{x}_1), b)) & \vee - \{P\}(\mathbf{x}_1)
\end{array}$$

The clause

$$C_5' = \{R\}(\mathbf{x}_1) \vee -\{P\}(f(\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_2)) \vee -\{Q\}(\mathbf{x}_2)$$

is not normal. Resolving it with C_3' gives the clause

$$\{R\}(g(a, a)) \vee -\{P\}(a) \vee -\{Q\}(a)$$

As a is accepted at $\{P\}$ and $\{Q\}$ using the normal clauses C_1' and C_2', hence we get a new normal clause

$$C_6 = \{R\}(g(a, a))$$

Resolving C_5' with C_4' gives

$$\{R\}(g(a, a)) \vee -\{P\}(a) \vee -\{Q\}(b)$$

But b is not accepted at $\{Q\}$ using the normal clauses hence this clause is rejected. Finally C_1' and C_2' also give the normal clause

$$C_7 = \{P, Q\}(a)$$

The resulting set of normal clauses is $\{C_1', \ldots, C_4', C_6, C_7\}$.

8 Conclusion

We have proved DEXPTIME-hardness of secrecy for cryptographic protocols with single blind copying, and have improved the upper bound from 3-DEXPTIME to DEXPTIME. We have improved the 3-DEXPTIME upper bound for satisfiability for the class $\mathcal{C}$ to NEXPTIME in the general case and DEXPTIME in the Horn case, which match known lower bounds. For this we have invented new resolution techniques like ordered resolution with splitting modulo propositional reasoning, ordered literal replacements and decompositions of one-variable terms. As byproducts we obtained optimum complexity for several fragments of $\mathcal{C}$ involving flat and one-variable clauses. Security for several other decidable classes of protocols with unbounded number of sessions and bounded number of nonces is in DEXPTIME, suggesting that DEXPTIME is a reasonable complexity class for such classes of protocols.

References

[1] A. Aiken, D. Kozen, M. Vardi, and E. Wimmers. The complexity of set constraints. In *CSL'93*, volume 832 of *LNCS*, pages 1–17. Springer-Verlag, 1993.

[2] L. Bachmair and H. Ganzinger. Resolution theorem proving. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. North-Holland, 2001.

[3] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Computer Society Press, Cape Breton, Nouvelle-Écosse, Canada, 2001.

[4] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1), Jan. 1981.

[5] H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *Theoretical Computer Science*, 331(1):143–214, 2005.

[6] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In R. Nieuwenhuis, editor, *14th International Conference on Rewriting Techniques and Applications (RTA'03)*, volume 2706 of *LNCS*, pages 148–164, Valencia, Spain, June 2003. Springer-Verlag.

[7] H. Comon-Lundh and V. Cortier. Security properties: Two agents are sufficient. In *12th European Symposium on Programming (ESOP'03)*, volume 2618 of *LNCS*, pages 99–113, Warsaw, Poland, Apr. 2003. Springer-Verlag.

[8] V. Cortier. *Vérification Automatique des Protocoles Cryptographiques*. PhD thesis, ENS Cachan, France, 2003.

[9] N. A. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Workshop on Formal Methods and Security Protocols (FMSP'99)*, Trento, Italy, 1999.

[10] H. Ganzinger and K. Korovin. New directions in instantiation-based theorem proving. In P. G. Kolaitis, editor, *18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 55–64, Ottawa, Canada, June 2003. IEEE Computer Society Press.

[11] J. Goubault-Larrecq. Résolution ordonnée avec sélection et classes décidables de la logique du premier ordre. Lecture notes for the course "démonstration automatique et vérification de protocoles cryptographiques" (with Hubert Comon-Lundh), DEA "programmation", 2004. 71 pages, `http://www.lsv.ens-cachan.fr/~goubault/SOresol.ps`.

[12] J. Goubault-Larrecq, M. Roger, and K. N. Verma. Abstraction and resolution modulo AC: How to verify Diffie-Hellman-like protocols automatically. *Journal of Logic and Algebraic Programming*, 64(2):219–251, Aug. 2005.

[13] D. Monniaux. Abstracting cryptographic protocols with tree automata. In A. Cortesi and G. Filé, editors, *6th International Static Analysis Symposium (SAS'99)*, volume 1694 of *LNCS*, pages 149–163, Venice, Italy, September 1999. Springer-Verlag.

[14] F. Nielson, H. R. Nielson, and H. Seidl. Normalizable Horn clauses, strongly recognizable relations and Spi. In *9th Static Analysis Symposium (SAS'02)*, volume 24477 of *LNCS*, pages 20–35. Springer-Verlag, 2002.

[15] A. Riazanov and A. Voronkov. Splitting without backtracking. In *IJCAI'01*, pages 611–617, 2001.

[16] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In P. Pandya and J. Radhakrishnan, editors, *14th IEEE Computer Security Foundations Workshop (CSFW'01)*, Cape Breton, Nova-Scotia, Canada, June 2001. IEEE Computer Society Press.

[17] C. Weidenbach. Towards an automatic analysis of security protocols. In H. Ganzinger, editor, *16th International Conference on Automated Deduction (CADE'99)*, number 1632 in LNAI, pages 378–382. Springer-Verlag, 1999.

A Proofs of Section 4

We use the following unification algorithm, due to Martelli and Montanari. It is described by the following rewrite rules on finite multisets of equations between terms; we let M be any such multiset, and comma denote multiset union:

(Delete) $M, u \doteq u \to M$

(Decomp) $M, f(u_1, \ldots, u_n) \doteq f(v_1, \ldots, v_n) \to M, u_1 \doteq v_1, \ldots, u_n \doteq v_n$

(Bind) $M, x \doteq v \to M[x := v], x \doteq v$ provided x is not free in v, but is free in M.

(Fail1) $M, x \doteq v \to \perp$ provided x is free in v and $x \neq v$.

(Fail2) $M, f(u_1, \ldots, u_m) \doteq g(v_1, \ldots, v_n) \to \perp$ provided $f \neq g$.

We consider that equations $u \doteq v$ are unordered pairs of terms u, v, so that in particular $u \doteq v$ and $v \doteq u$ are the same equation. $\perp$ represents failure of unification. If s and t are unifiable, then this rewrite process terminates, starting from $s \doteq t$, on a so-called solved form $z_1 \doteq u_1, \ldots, z_k \doteq u_k$; then $\sigma = \{z_1 \mapsto u_1, \ldots, z_k \mapsto u_k\}$ is an mgu of $s \doteq t$.

Lemma 4 *Let $s[x]$ and $t[y]$ be two non-ground non-trivial one-variable terms, and $x \neq y$. Let U be the set of non-ground strict subterms of s and t and let V be the set of ground strict subterms of s and t. If $s[x]$ and $t[y]$ are unifiable then they have a mgu σ such that one of the following is true:*

- $\sigma = \{x \mapsto u[y]\}$ *where* $u \in U$.
- $\sigma = \{y \mapsto u[x]\}$ *where* $u \in U$.
- $\sigma = \{x \mapsto u, y \mapsto v\}$ *where* $u, v \in U[V]$.

Proof: Note that $V \subseteq U[V]$ since U contains the trivial terms also. We use the above unification algorithm. We start with the multiset $M_0 = s \doteq t$. We claim that if $M_0 \to^+ M$ then M is of one of the following forms:

1. $s_1[x] = t_1[y], \ldots, s_n[x] = t_n[y]$, where each $s_i, t_i \in U \cup V$, some $s_i \in U$ and some $t_j \in U$.
2. $s_1[u[y']] = t_1[y'], \ldots, s_n[u[y']] = t_n[y'], x' = u[y']$ where $u \in U$, each $s_i, t_i \in U \cup V$, $x' \in \{x, y\}$ and $y' \in \{x, y\} \setminus \{x'\}$.
3. $s_1[u] = t_1[y'], \ldots, s_n[u] = t_n[y'], x' = u$ where $u \in V$, each $s_i, t_i \in U \cup V$, some $t_i \in U$, $x' \in \{x, y\}$ and $y' \in \{x, y\} \setminus \{x'\}$.
4. $M', x = u, y = v$ where $u, v \in U[V]$, and no variables occur in M'.
5. $\perp$.

As s and t are non-trivial, and x and y are distinct, hence (Delete) and (Bind) don't apply on M_0. Applying (Decomp) on M_0 leads us to type (1). Applying (Fail1) or (Fail2) on any M leads us to $\perp$. Applying (Delete) and (Decomp) on type (1) keeps us in type (1). Applying (Bind) on type (1) leads to type (2) or (3) depending on whether the concerned variable is replaced by a non-ground or ground term. Applying (Delete) on type (2) leads to type (2) itself. Applying (Decomp) on type (2) leads to type (2) itself. (Bind) applies on M of type (2) only if M contains some $y' \doteq v$ where v is ground. We must have $v \in V$. The result is of type (4). Applying (Delete) and (Decomp) rules on type (3) leads to type (3) itself. (Bind) applies on M of type (3) only if M contains some $y' \doteq v$ where v is ground. We must have $v \in U[V]$. The result is of type (4). Applying (Delete) and (Decomp) on type (4) leads to type (4) itself, and (Bind) does not apply.

Now we look at the solved forms. Solved forms of type (1) are of the form either $x \doteq u[y]$ with $u \in U$, or $y \doteq u[x]$ with $u \in U$, or $x \doteq u, y \doteq v$ with $u, v \in V \subseteq U[V]$.

M of type (2) is in solved form only if $n = 0$. Hence the solved forms are again of the form $x \doteq u[y]$ or $y \doteq u[x]$ with $u \in U$. M of type (3) is in solved form only if $n = 1$, hence M is of the form $x \doteq u, y \doteq v$ with $u, v \in U[V]$. Solved forms of type (4) are again of type $x \doteq u, y \doteq v$ with $u, v \in U[V]$ (i.e. M' is empty). □

Lemma 2 *Let $s[x]$ and $t[y]$ be reduced, non-ground and non-trivial terms where $x \neq y$ and $s[x] \neq t[x]$. If s and t have a unifier σ then $x\sigma, y\sigma \in U[V]$ where U is the set of non-ground (possibly trivial) strict subterms of s and t, and V is the set of ground strict subterms of s and t.*

Proof: By Lemma 4, $s[x]$ and $t[y]$ have a mgu σ' such that one of the following is true:

- $\sigma' = \{x \mapsto u[y]\}$ where $u \in U$. We have $s[u[y]] = t[y]$. As t is reduced, this is possible only if u is trivial. Hence $s[y] = t[y]$, so $s[x] = t[x]$. This is a contradiction.
- $\sigma' = \{y \mapsto u[x]\}$ where $u \in U$. This case is similar to the previous case.
- $\sigma' = \{x \mapsto u, y \mapsto v\}$ where $u, v \in U[V]$. As σ' is the mgu and maps x and y to ground terms, hence $\sigma = \sigma'$. □

Lemma 3 *Let σ be a unifier of two non-trivial, non-ground and distinct one-variable terms $s[x]$ and $t[x]$. Then $x\sigma$ is a ground strict subterm of s or of t.*

Proof: We use the above unification algorithm. We start with the multiset $M_0 = s[x] = t[x]$. If $M_0 \to^+ M$ then M is of one of the following forms:

1. $s_1[x] = t_1[x], \ldots, s_n[x] = t_n[x]$ where each s_i is a strict subterm of s and each t_i is a strict subterm of t
2. $M, x = u$ where u is a ground strict subterm of s or t, and no variables occur in M
3. $\perp$.

Then it is easy to see that the only possible solved form is $x \doteq u$ where u is a ground strict subterm of s or t. □

B Proofs of Section 6

Theorem 4 *Modified ordered resolution, wrt a stable and enumerable ordering, with splitting and $\mathcal{Q}$-splitting and ordered literal replacement is sound and complete for any strategy. I.e. for any set S of $\mathbb{P}$-clauses, for any strict stable and enumerable partial order $<$ on atoms, for any set $\mathcal{R}$ of ordered replacement rules, for any finite set $\mathcal{Q}$ of splitting atoms, and for any $\mathcal{Q}$-splitting-replacement strategy ϕ, $S \cup cl(\mathcal{R})$ is unsatisfiable iff $S \Rrightarrow^*_{<_s,\phi,\mathcal{R}} \mathcal{T}$ for some closed $\mathcal{T}$.*

Proof: A *standard Herbrand interpretation* is a Herbrand interpretation $\mathcal{H}$ such that $\overline{C} \in \mathcal{H}$ iff $\mathcal{H}$ does not satisfy C. This leads us to the notion of *standard satisfiability* as expected. The given set S of $\mathbb{P}$-clauses is satisfiable iff it is standard-satisfiable. Ordered resolution, factorization and splitting preserve satisfiability in any given Herbrand interpretation, and $\mathcal{Q}$-splitting preserves satisfiability in any given standard-Herbrand interpretation. Also if $T \rightarrow_{\mathcal{R}} T'$ then $T \cup cl(\mathcal{R})$ is satisfiable in a Herbrand interpretation iff $T' \cup cl(\mathcal{R})$ is satisfiable in that interpretation. This proves correctness: if $S \Rrightarrow^*_{<_s,\phi,\mathcal{R}} \mathcal{T}$ and $\mathcal{T}$ is closed then $S \cup cl(\mathcal{R})$ is unsatisfiable.

For completeness we replay the proof of [11] for ordered resolution with selection specialized to our case, and insert the arguments required for the replacement rules. Since $<$ is enumerable, hence we have an enumeration $A'_1, A'_2, \ldots$ of all ground atoms such that if $A'_i < A'_j$ then $i < j$. Also there are only finitely many splitting atoms in $\mathcal{Q}$, all of which are smaller than non-splitting atoms. Hence the set of all (splitting as well as non-splitting) atoms can be enumerated as $A_1, A_2, \ldots$ such that if $A_i <_s A_j$ then $i < j$. Clearly all the splitting atoms occur before the non-splitting atoms in this enumeration. Consider the infinite binary tree $\mathbb{T}$ whose nodes are literal sequences of the form $\pm_1 A_1 \pm_2 A_2 \ldots \pm_k A_k$ for $k \geq 0$. The two successors of the node N are $N + A_{k+1}$ (the left child) and $N - A_{k+1}$ (the right child). If $k = 0$ then N is a root node. Furthermore we write $-N = \mp_1 A_i \mp_2 A_2 \ldots \mp_k A_k$. A clause *fails* at a node N if there is some ground substitution σ such that for every literal $L \in C$, $L\sigma$ is in $-N$. For any set T of clauses define $\mathbb{T}_T$ as the tree obtained from $\mathbb{T}$ by deleting the subtrees below all nodes of $\mathbb{T}$ where some clause of T fails. A failure-witness for a set T of clauses is a tuple $(\mathbb{T}', C_\bullet, \theta_\bullet)$ such that $\mathbb{T}' = \mathbb{T}_T$ is finite, C_N is a clause for each leaf node N of $\mathbb{T}'$, and θ_N is a ground substitution for each leaf node N of $\mathbb{T}'$ such that for $-N$ contains every $L \in C_N\theta_N$. We define $\nu(\mathbb{T}')$ as the number of nodes in $\mathbb{T}'$. For any failure witness of the form $(\mathbb{T}', C_\bullet, \theta_\bullet)$ and for any leaf node $N = \pm_1 A_1 \pm_2 A_2 \ldots \pm_k A_k$ of $\mathbb{T}'$, define $\mu_1(C_N, \theta_N)$ as follows:

– If $C_N \notin cl(\mathcal{R})$ then $\mu_1(C_N, \theta_N)$ is the multiset of integers which contains the integer i as many times as there are literals $\pm A' \in C_N$ such that $A'\theta_N = A_i$.

– If $C_N \in cl(\mathcal{R})$ then $\mu_1(C_N, \theta_N)$ is the empty multiset.

We define $\mu^-(\mathbb{T}', C_\bullet, \theta_\bullet)$ as the multiset of the values $\mu_1(C_N, \theta_N)$ where N ranges over all leaf nodes of $\mathbb{T}'$. We define $\mu(\mathbb{T}', C_\bullet, \theta_\bullet) = (\nu(\mathbb{T}'), \mu^-(\mathbb{T}', C_\bullet, \theta_\bullet))$. We consider the lexicographic ordering on pairs, i.e. $(x_1, y_1) < (x_2, y_2)$ iff either $x_1 < x_2$, or $x_1 = x_2$ and $y_1 < y_2$. Since $S \cup cl(\mathcal{R})$ is unsatisfiable, from König's Lemma:

Lemma 5 *$S \cup cl(\mathcal{R})$ has a failure witness.*

Lemma 6 *If T has a failure witness $(\mathbb{T}_T, C_\bullet, \theta_\bullet)$ such that $\mathbb{T}_T$ is not just the root node, then there is some T' with a failure witness $(\mathbb{T}_{T'}, C'_\bullet, \theta'_\bullet)$ such that $T \Rrightarrow_{<_s} T'$ and $\mu(\mathbb{T}_{T'}, C'_\bullet, \theta'_\bullet) < \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$.*

Proof: In the following the notion of mgu is generalized and we write $mgu(s_1 \doteq \ldots \doteq s_n)$ for the most general substitution which makes $s_1, \ldots, s_n$ equal. We iteratively define a sequence $R_0, R_1, \ldots$ of nodes, none of which is a leaf node. R_0 is the empty sequence which is not a leaf node. Suppose we have already defined R_i. As R_i is not a leaf node, R_i has a descendant N_i such that $N_i - B_i$ is rightmost leaf node in the subtree of $\mathbb{T}_T$ rooted at R_i.

(1) If B_i is a non-splitting atom then stop the iteration.

(2) Otherwise B_i is a splitting atom.

(2a) If the subtree rooted at $N_i + B_i$ has some leaf node N such that $-B_i \in C_N$ then stop the iteration.

(2b) Otherwise $N_i + B_i$ cannot be a leaf node. Define $R_{i+1} = N_i + B_i$ and continue the iteration.

$\mathbb{T}_T$ is finite hence the iteration terminates. Let k be the largest integer for which R_k, and hence N_k and B_k are defined. For $0 \leq i \leq k-1$, B_i is a splitting literal. The only positive literals in the sequence N_k are from the set $\{B_0, \ldots, B_{k-1}\}$. $N_k - B_k$ is a leaf node of $\mathbb{T}_T$.

Suppose the iteration stopped in case (1) above. Then N_k has some descendant N such that its two children $N - B$ and $N + B$ are leaf nodes of $\mathbb{T}_T$, and B is a non-splitting literal. As B_k is a non-splitting literal, no negative splitting literals are present in C_{N-B} or C_{N+B}. C_{N-B} is of the form $C_1 \vee B'_1 \vee \ldots \vee B'_m (m \geq 1)$ such that $B'_1\theta_{N-B} = \ldots = B'_m\theta_{N-B} = B$ and each literal in $C_1\theta_{N-B}$ is present in $-N$. The literals $B'_1, \ldots, B'_m$ are then maximal in C_{N-B} and can be selected for resolution. C_{N+B} is of the form $C_2 \vee -B''_1 \vee \ldots \vee -B''_n (n \geq 1)$ such that $B''_1\theta_{N+B} = \ldots = B''_n\theta_{N+B} = B$ and each literal in $C_2\theta_{N+B}$ is present in $-N$. The literals $B''_1, \ldots, B''_n$ are then maximal in C_{N+B} and can be selected for resolution. We assume that C_{N-B} and C_{N+B} are renamed apart so as not to share variables. Let θ be a ground substitution which maps each $x \in \mathsf{fv}(C_{N-B})$ to $x\theta_{N-B}$ and $x \in \mathsf{fv}(C_{N+B})$ to $x\theta_{N+B}$. We have $B'_1\theta = \ldots = B'_m\theta = B''_1\theta = \ldots = B''_n\theta$. Then $\sigma = mgu(B'_1 \doteq \ldots \doteq B'_m \doteq B''_1 \doteq \ldots \doteq B''_n)$ exists. Hence we have some ground substitution θ' such that $\sigma\theta' = \theta$. Hence by repeated applications of the ordered factorization and ordered binary resolution rule, we obtain the resolvent $C = C_1\sigma \vee C_2\sigma$, and $T \Rrightarrow_{<_s} T' = T \cup \{C\}$. We have $C\theta' = C_1\theta \vee C_2\theta$. Hence C fails at node N. Then $\mathbb{T}_{T'}$ is finite and $\nu(\mathbb{T}_{T'}) < \nu(\mathbb{T}_T)$. Hence by choosing any $C'_\bullet$ and $\theta'_\bullet$ such that $(\mathbb{T}_{T'}, C'_\bullet, \theta'_\bullet)$ is a failure witness for T', we have $\mu(\mathbb{T}_{T'}, C'_\bullet, \theta'_\bullet) < \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$.

If the iteration didn't stop in case (1) but in case (2a) then it means that B_k is a splitting literal. Then $C_{N_k - B_k} = C_1 \vee +B_k$ (with $B_k \notin C_1$). C_1 has no negative splitting literals. Hence the only literals in C_1 are positive splitting literals. Hence the literal B_k can be chosen from $C_{N_k - B_k}$ for resolution. The subtree rooted at $N_k + B_k$ has some leaf node N such that $-B_k \in C_N$. Then $C_N = C_2 \vee -B_k$ (and $-B_k \notin C_2$). Hence $-B_k$ can be selected from C_N for resolution. We obtain the resolvent $C_2 \vee C_1$ which fails at N. Let $T' = T \cup \{C_1 \vee C_1\}$. We have $\nu(\mathbb{T}_{T'}) \leq \nu(\mathbb{T}_T)$. If N' is the highest ancestor of N where $C_2 \vee C_1$ fails then N' is a leaf of $\mathbb{T}_{T'}$ and we define $C'_{N'} = C_2 \vee C_1$ and $\theta'_{N'} = \theta_N$. We have $\mu_1(C'_{N'}, \theta'_{N'}) < \mu_1(C_N, \theta_N)$ since all literals in C_1 are splitting literals $\pm q$ such that q occurs strictly before B_k in the enumeration $A_1, A_2, \ldots$. (Also note that $C_N \notin cl(\mathcal{R})$ because C_N contains a splitting literal). All other leaf nodes N'' of $\mathbb{T}_{T'}$ are also leaf nodes of $\mathbb{T}_T$ and we define $C'_{N''} = C_{N''}$ and $\theta'_{N''} = \theta_{N''}$. Then $(\mathbb{T}_{T'}, C'_\bullet, \theta'_\bullet)$ is a failure witness for T' and we have $\mu^-(\mathbb{T}_{T'}, C'_\bullet, \theta'_\bullet) < \mu^-(\mathbb{T}_T, C_\bullet, \theta_\bullet)$. Hence we have $\mu(\mathbb{T}_{T'}, C'_\bullet, \theta'_\bullet) < \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$. $\square$

Lemma 7 *If T has a failure witness $(\mathbb{T}_T, C_\bullet, \theta_\bullet)$ and $T \rightarrow_{\mathcal{Q}-nspl} T'$ then $T' \cup cl(\mathcal{R})$ has a failure witness $(\mathbb{T}_{T'\cup cl(\mathcal{R})}, C'_\bullet, \theta'_\bullet)$ with $\mu(\mathbb{T}_{T'\cup cl(\mathcal{R})}, C'_\bullet, \theta'_\bullet) \leq \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$.*

Proof: Let $C = C_1 \sqcup C_2 \in T$, C_2 is a non-empty $\mathbb{P}$-clause, C_1 has at least one non-splitting literal, and $T \rightarrow_{\mathcal{Q}-nspl} T' = (T \setminus \{C\}) \cup \{C_1 \vee -\overline{C_2}, \overline{C_2} \vee C_2\}$. If $C \neq C_N$ for any leaf node N of $\mathbb{T}_T$ then there is nothing to show. Now suppose $C = C_N$ where N is a leaf node of $\mathbb{T}_T$. If $C_N \in cl(\mathcal{R})$ then there is nothing to prove. Now suppose $C_N \notin cl(\mathcal{R})$. As C is constrained to contain at least one non-splitting literal, hence the literal sequence N has at least one non-splitting literal. By the chosen enumeration $A_1, A_2, \ldots$, either $\overline{C_2}$ or $-\overline{C_2}$ occurs in the literal sequence N.

- If $\overline{C_2}$ occurs in N then $C_1 \vee -\overline{C_2}$ fails at N. Let N' be the highest ancestor of N where it fails. N' is a leaf node of $\mathbb{T}_{T'}$. We define $C''_{N'} = C_1 \vee -\overline{C_2}$ and $\theta''_{N'} = \theta_N$. All other leaf nodes N'' of $\mathbb{T}_{T'}$ are also leaf nodes of $\mathbb{T}_T$ and we define $C''_{N''} = C_{N''}$ and $\theta''_{N''} = \theta_{N''}$. $(\mathbb{T}_{T'}, C''_\bullet, \theta''_\bullet)$ is a failure witness for T'. As C_2 has at least one non-splitting literal, we have $\mu_1(C''_{N'}, \theta''_{N'}) < \mu_1(C_N, \theta_N)$ (recall that $C_N \notin cl(\mathcal{R})$) so that $\mu(\mathbb{T}_{T'}, C''_\bullet, \theta''_\bullet) \leq \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$. As $T' \subseteq T' \cup cl(\mathcal{R})$ hence the result follows.
- If $-\overline{C_2}$ occurs in N then $C_2 \vee \overline{C_2}$ fails at N. Since C_1 has at least one non-splitting literal, as in the previous case, we obtain a failure witness $(\mathbb{T}_{T'}, C''_\bullet, \theta''_\bullet)$ such that $\mu(\mathbb{T}_{T'}, C''_\bullet, \theta''_\bullet) \leq \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$. □

Lemma 8 *If T has a failure witness $(\mathbb{T}_T, C_\bullet, \theta_\bullet)$ and $T \rightarrow_{spl} T_1 \mid T_2$ then $T_1 \cup cl(\mathcal{R})$ and $T_2 \cup cl(\mathcal{R})$ have failure witnesses $(\mathbb{T}_{T_1\cup cl(\mathcal{R})}, C'_\bullet, \theta'_\bullet)$ and $(\mathbb{T}_{T_2\cup cl(\mathcal{R})}, C''_\bullet, \theta''_\bullet)$ such that $\mu(\mathbb{T}_{T_1\cup cl(\mathcal{R})}, C'_\bullet, \theta'_\bullet) \leq \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$ and $\mu(\mathbb{T}_{T_2\cup cl(\mathcal{R})}, C''_\bullet, \theta''_\bullet) \leq \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$.*

Proof: Let $C = C_1 \sqcup C_2 \in T$ such that C_1 and C_2 share no variables, and we have $T \rightarrow_{spl} T_1 \mid T_2$ where $T_i = T \cup \{C_i\}$. We prove the required result for T_1, the other part is symmetric. If $C \neq C_N$ for any leaf node N of $\mathbb{T}_T$ then there is nothing to show. Now suppose $C = C_N$ for some leaf node N of $\mathbb{T}_T$. If $C_N \in cl(\mathcal{R})$ then there is nothing to show. Now suppose $C_N \notin cl(\mathcal{R})$. Since $C_1 \subseteq C$, hence C_1 also fails at N. Let N' be the highest ancestor of N where C_1 fails. N' is a leaf node of $\mathbb{T}_{T_1}$. We define $C'''_{N'} = C$ and $\theta'''_{N'} = \theta$. All other leaf nodes N'' of $\mathbb{T}_{T_1}$ are also leaf nodes of $\mathbb{T}_T$, and we define $C'''_{N''} = C_{N''}$ and $\theta'''_{N''} = \theta_{N''}$. $(\mathbb{T}_{T_1}, C'''_\bullet, \theta'''_\bullet)$ is a failure witness for T_1. Also $\mu_1(C'''_{N'}, \theta'''_{N'}) \leq \mu_1(C_N, \theta_N)$ (recall that $C_N \notin cl(\mathcal{R})$). Hence $\mu(\mathbb{T}_{T'}, C'''_\bullet, \theta'''_\bullet) \leq \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$. As $T_1 \subseteq T_1 \cup cl(\mathcal{R})$, hence the result follows.

The following arguments are the ones that take care of replacement steps.

Lemma 9 *If T has a failure witness $(\mathbb{T}_T, C_\bullet, \theta_\bullet)$ and $T \rightarrow_{\mathcal{R}} T'$ then $T' \cup cl(\mathcal{R})$ has a failure witness $(\mathbb{T}_{T'\cup cl(\mathcal{R})}, C'_\bullet, \theta'_\bullet)$ with $\mu(\mathbb{T}_{T'\cup cl(\mathcal{R})}, C'_\bullet, \theta'_\bullet) \leq \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$.*

Proof: Let $C_1 = C'_1 \vee \pm A\sigma \in T$, $R = A \rightarrow B \in \mathcal{R}$, and $T \rightarrow_{\mathcal{R}} T' = (T \setminus \{C_1\}) \cup \{C\}$ where $C = C'_1 \vee \pm B\sigma$. If $C_1 \neq C_N$ for any leaf node of $\mathbb{T}_T$ then there is nothing to prove. Now suppose that $C_1 = C_N$ for some leaf node N of $\mathbb{T}_T$. Let $N = \pm_1 A_1 \ldots \pm_k A_k$. If $C_1 \in cl(\mathcal{R})$ then $T \subseteq T' \cup cl(\mathcal{R})$, and there is nothing to prove. Now suppose $C_1 \notin cl(\mathcal{R})$. We have a ground substitution θ

such that $C_1\theta = C_1'\theta \vee \pm A\sigma\theta \subseteq \{\mp_1 A_1, \ldots, \mp_k A_k\}$. As R is ordered we have $A \geq B$. Hence $A\sigma\theta \geq B\sigma\theta$. Hence either $\pm B\sigma\theta \in \{\mp_1 A_1, \ldots, \mp_k A_k\}$ or $\mp B\sigma\theta \in \{\mp_1 A_1, \ldots, \mp_k A_k\}$.

- Suppose $\pm B\sigma\theta \in \{\mp_1 A_1, \ldots, \mp_k A_k\}$. Since $C_1\theta = C_1'\theta \vee \pm A\sigma\theta \subseteq \{\mp_1 A_1, \ldots, \mp_k A_k\}$, hence $C\theta = C_1'\theta \vee \pm B\sigma\theta \subseteq \{\mp_1 A_1, \ldots, \mp_k A_k\}$. Hence C fails at N. Let N' be the highest ancestor of N where C fails. N' is a leaf node of $\mathbb{T}_{T'}$. We define $C''_{N'} = C$ and $\theta''_{N'} = \theta$. All other leaf nodes N'' of $\mathbb{T}_{T'}$ are also leaf nodes of $\mathbb{T}_T$, and we define $C''_{N''} = C_{N''}$ and $\theta''_{N''} = \theta_{N''}$. $(\mathbb{T}_{T'}, C''_\bullet, \theta''_\bullet)$ is a failure witness for T'. Also $\mu_1(C''_{N'}, \theta''_{N'}) \leq \mu_1(C_N, \theta_N)$ (recall that $C_N \notin cl(\mathcal{R})$). Hence $\mu(\mathbb{T}_{T'}, C'_\bullet, \theta'_\bullet) \leq \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$. As $T' \subseteq T' \cup cl(\mathcal{R})$, hence the result follows.

- Suppose $\mp B\sigma\theta \in \{\mp_1 A_1, \ldots, \mp_k A_k\}$. Since $\pm A\sigma\theta = \in \{\mp_1 A_1, \ldots, \mp_k A_k\}$, hence the clause $\mp A \vee \pm B \in cl(\mathcal{R})$ fails at N. Let N' be the highest ancestor of N where $\mp A \vee \pm B$ fails. N' is a leaf node of $\mathbb{T}_{T' \cup \{\mp A \vee \pm B\}}$. We define $C''_{N'} = C$ and $\theta''_{N'} = \theta$. All other leaf nodes N'' of $\mathbb{T}_{T' \cup \{\mp A \vee \pm B\}}$ are also leaf nodes of $\mathbb{T}_T$, and we define $C''_{N''} = C_{N''}$ and $\theta''_{N''} = \theta_{N''}$. $(\mathbb{T}_{T' \cup \{\mp A \vee \pm B\}}, C''_\bullet, \theta''_\bullet)$ is a failure witness for $T' \cup \{\mp A \vee \pm B\}$. Also $\mu_1(C''_{N'}, \theta''_{N'}) \leq \mu_1(C_N, \theta_N)$ since $\mu_1(C''_{N'}, \theta''_{N'})$ is the empty multiset. Hence $\mu(\mathbb{T}_{T' \cup \{\mp A \vee \pm B\}}, C''_\bullet, \theta''_\bullet) \leq \mu(\mathbb{T}_T, C_\bullet, \theta_\bullet)$. As $T' \cup \{\mp A \vee \pm B\} \subseteq T' \cup cl(\mathcal{R})$, hence the result follows. □

For a tableaux $\mathcal{T} = S_1 \mid \ldots \mid S_n$, define $\mathcal{T} \cup S = S_1 \cup S \mid \ldots \mid S_n \cup S$. We define a failure witness for such a $\mathcal{T}$ to be a multiset $\{(\mathbb{T}_{S_1}, C^1_\bullet, \theta^1_\bullet), \ldots, (\mathbb{T}_{S_n}, C^1_\bullet, \theta^n_\bullet)\}$ where each $(\mathbb{T}_{S_i}, C^i_\bullet, \theta^i_\bullet)$ is a failure witness of S_i. We define

$\mu(\{\mathbb{T}_{S_1}, C^1_\bullet, \theta^1_\bullet), \ldots, (\mathbb{T}_{S_n}, C^1_\bullet, \theta^n_\bullet\}) = \{\mu(\mathbb{T}_{S_1}, C^1_\bullet, \theta^1_\bullet), \ldots, \mu(\mathbb{T}_{S_n}, C^1_\bullet, \theta^n_\bullet)\}$.

Then it is clear that $S \cup cl(\mathcal{R})$ has a failure witness and whenever any $\mathcal{T}$ has a failure witness in which one of the trees has at least two nodes, then $\mathcal{T} \Rrightarrow_{<_s, \phi, \mathcal{R}} \mathcal{T}'$ for some $\mathcal{T}'$ such that $\mathcal{T}' \cup cl(\mathcal{R})$ has a strictly smaller failure witness. Hence we have some $\mathcal{T}$ such that $S \Rrightarrow^*_{<_s, \phi, \mathcal{R}} \mathcal{T}$ and $\mathcal{T} \cup cl(\mathcal{R})$ has a failure witness in which each tree is a root node. Then $\mathcal{T} \cup cl(\mathcal{R})$ is closed. Hence $\mathcal{T}$ is closed. □