A Resolution Calculus for Propositional Schemata

Vincent Aravantinos, Mnacho Echenim, Nicolas Peltier

A Resolution Calculus for Propositional Schemata

Vincent Aravantinos, Mnacho Echenim and Nicolas Peltier

February 7, 2011

1 Introduction

This report describes a resolution calculus handling families of propositional formulae. The calculus is proven to be sound, refutationally complete and terminating. The class of propositional schemata we consider is more or less equivalent to the class of *regular schemata* introduced in [1] although the definitions differ. The core of the resolution calculus (namely the resolution and factoring rule and the encoding of arithmetic parameters) is close to the one of [5] (although we use a different presentation). The main originality of our approach is the loop detection rule ensuring termination and refutational completeness.

2 Preliminaries

Informally, the syntax and semantics of our logic are almost identical to a very simple subclass of clausal logic, except that we consider a particular set of constant symbols, called *parameters*, that must be interpreted as natural numbers. Every predicate symbol is monadic and the only function symbols are $0, succ$ (natural numbers).

Let Ω be a set of unary predicate symbols, or (indexed) *propositional variables*. Let $\mathcal{P}$ and $\mathcal{V}$ be two disjoint sets of *variables*. The elements in $\mathcal{P}$ are the *parameters*, and the ones in $\mathcal{V}$ are the *index variables*. Throughout this paper, parameters are denoted by n, m and index variables by x, y, z. The letters i, j, k, l will denote natural numbers (meta-variables).

The set of *(arithmetic) terms* $\mathcal{T}$ is the least set satisfying $\mathcal{V} \subseteq \mathcal{T}$, $0 \in \mathcal{T}$ and $t \in \mathcal{T} \Rightarrow succ(t) \in \mathcal{T}$. Notice that $\mathcal{T}$ does *not* contain the parameters. A term is *ground* iff it is of the form $succ^i(0)$ (with $i \in \mathbb{N}$), in which case it may be viewed as a natural number and simply denoted by i. The set of ground terms is denoted by $\mathbb{N}$. If x is a variable and i is a natural number then $x + i$ denotes the term $succ^i(x)$.

An *atom* is either of the form $n \approx t$ where $n \in \mathcal{P}$ and $t \in \mathcal{T}$ or of the form p_t where $p \in \Omega$ and $t \in \mathcal{T}$. It is *ground* if t is ground. Notice that parameters only occur in equational atoms. Atoms of the form $n \approx t$ (resp. p_t) are called *equational atoms* (resp. *indexed atoms*).

A *literal* is either an atom (positive literal) or the negation of an atom (negative literal). A *clause* is a finite set (or disjunction) of literals (the empty clause is denoted by $\Box$). An *equational literal* (resp. *indexed literal*) is a literal whose atom is an equational atom (resp. an indexed atom). We denote by C^{eq} and C^{in} the sets (disjunctions) of equational literals (resp. indexed literals) in C. If S is a set of clauses then S^{in} denotes the set of clauses $\{C^{in} \mid C \in S\}$. A clause C is *purely equational* iff $C^{in} = \emptyset$ and *parameter-free* iff $C^{eq} = \emptyset$.

For every expression (term, atom, literal or clause) $\mathcal{E}$, $var(\mathcal{E})$ denotes the set of index variables occurring in $\mathcal{E}$.

The *depth* of an expression is defined as usual: $depth(x) \stackrel{\text{def}}{=} 0$ if $x \in \mathcal{V}$, $depth(0) = 0$, $depth(succ(t)) = 1 + depth(t)$, $depth(\neg p_t) \stackrel{\text{def}}{=} depth(p_t) \stackrel{\text{def}}{=} depth(t)$, $depth(n \not\approx t) \stackrel{\text{def}}{=} depth(n \approx t) \stackrel{\text{def}}{=} depth(t)$ and $depth(\bigvee_{i=1}^{n} l_i) \stackrel{\text{def}}{=} \max_{i \in [1,n]} depth(l_i)$ (by convention $depth(\Box) \stackrel{\text{def}}{=} 0$).

A *substitution* σ is a function mapping every index variable x to a term $x\sigma \in \mathcal{T}$ (notice that parameters cannot be mapped). The *domain* $dom(\sigma)$ of σ is the set of index variables x such that $x\sigma \neq x$. For every expression $\mathcal{E}$, $\mathcal{E}\sigma$ denotes the expression obtained from $\mathcal{E}$ by replacing every variable x by $x\sigma$. A substitution σ is *ground* iff for every $x \in dom(\sigma)$, $x\sigma$ is ground. A *renaming* is an injective substitution σ such that $x\sigma \in \mathcal{V}$ for every $x \in dom(\sigma)$. A σ is *flat* if for every $x \in \mathcal{V}$, $x\sigma \in \mathcal{V} \cup \{0\}$.

σ is a *unifier* of $t_1, \ldots, t_n$ iff $t_1\sigma = \ldots = t_n\sigma$. As is well known, any unifiable set of terms has a most general unifier (unique up to a renaming), denoted by $\mathrm{mgu}(t_1, \ldots, t_n)$. Notice that in our simple case, all terms are of the form $succ^i(t)$ where $t \in \{0\} \cup \mathcal{V}$. Then any unifier of $succ^i(t)$ and $succ^j(s)$ where $t, s \in \{0\} \cup \mathcal{V}$ (if it exists) is necessarily either empty or of the form $t \mapsto succ^{j-i}(s)$ (if $t \in \mathcal{V}$, $t \neq s$ and $j > i$) or of the form $s \mapsto succ^{i-j}(t)$ (if $t \in \mathcal{V}$, $t \neq s$ and $j < i$).

An *interpretation* I is a function mapping:

- Every parameter to a ground term in $\mathcal{T}$ (i.e. a natural number).
- Every ground indexed atom to a truth value `true` or `false`.

An interpretation I *validates*:

- A ground atom $n \approx t$ iff $I(n) = t$.
- A ground atom p_t iff $I(p_t) =$ `true`.
- A ground literal $\neg a$ iff I does not validate a.
- A ground clause C iff it validates at least one literal in C.
- A non ground clause C iff for every substitution σ of domain $var(C)$, I validates $C\sigma$.
- A set of clauses S iff it validates every clause in S.

We write $I \models S$ iff I validates S (then I is called a *model* of S). If S, S' are two sets of clauses, we write $S \models S'$ iff for every interpretation I, we have $I \models S \Rightarrow I \models S'$. By definition, an interpretation I validates a clause $n \not\approx succ^i(x)$ iff $I(n) < i$. Thus the notation $n < i$ is a shorthand for the clause $n \not\approx succ^i(x)$.

Proposition 1 *The satisfiability problem is decidable for (finite) purely equational clause sets.*

PROOF. By definition of the semantics, a purely equational clause set $\{C_1, \ldots, C_k\}$ is equivalent to the formula $\bigwedge_{i=1}^{k} \forall \vec{x_i} C_i$ where $\vec{x_i}$ is the set of index variables in C_i. The only symbols occurring in this formula (beside the variables in $\vec{x_i}$) are 0, *succ*, $\approx$, $\not\approx$ and the parameters. Since 0, *succ* are non interpreted, $\{C_1, \ldots, C_k\}$ is satisfiable iff the formula $\exists \vec{n}. \bigwedge_{i=1}^{k} \forall \vec{x_i} C_i$ holds in the empty theory, where $\vec{n}$ is the set of parameters in $C_1, \ldots, C_n$. The satisfiability problem is well known to be decidable for such formulae (by the results in [6, 4] or even [3]). ∎

Remark 2 *Notice that by definition, any clause of the form $n \not\approx u \vee n \not\approx v \vee C$ such that u and v are not unifiable is valid. For instance, $n \not\approx succ(x) \vee n \not\approx x$ holds in any interpretation, since n cannot be equal to both $succ(x)$ and x.*

3 The Calculus

3.1 Definition

If u and v are two terms, we write $u \lhd v$ iff $v = succ^i(u)$ for some $i > 0$. This ordering is extended into a pre-ordering on atoms as follows: $p_u \lhd q_v$ iff $u \lhd v$.

We assume that a total ordering $\prec$ is given on elements of Ω. $\prec$ is extended into an ordering on atoms as follows: $a \prec a'$ iff either $a \lhd a'$ or there exists a term u such that $a = p_u$, $a' = q_u$ and $p \prec q$. By definition, the equational atoms are not comparable.

Notice that $\prec$ is stable by substitution, i.e. $a \prec a' \Rightarrow a\sigma \prec a'\sigma$, for every substitution σ. $\lhd$ and $\prec$ are extended to index literals by ignoring negation symbols and to clauses by the multiset extension.

Let sel be a *selection function* mapping every clause C to a (*possibly empty*) set of *selected index literals* in C. We assume that sel satisfies the following conditions: for every clause C such that $C^{in} \neq \emptyset$, either sel(C) is a nonempty set of $\lhd$-maximal negative literals or sel(C) is the set of $\prec$-maximal literals in C^{in}. Notice that by definition equational literals are never selected (thus sel(C) $= \emptyset$ if $C^{in} = \emptyset$).

For instance, consider the clause

$$n \not\approx succ(succ(x)) \vee \neg p_x \vee \neg p_{succ(x)} \vee q_{succ(x)} \vee r_{succ(x)}.$$

Assume that $p \prec q \prec r$. $p_{succ(x)}$, $q_{succ(x)}$ and $r_{succ(x)}$ are $\lhd$-maximal. $r_{succ(x)}$ is $\prec$-maximal. sel can contain the literals $\neg p_{succ(x)}$ or $r_{succ(x)}$ but neither $\neg p_x$ nor $q_{succ(x)}$.

The Resolution calculus is defined, as usual, by the following rules (we externalize factorization for technical convenience).

$$\frac{p_u \vee C \qquad \neg p_v \vee D}{(C \vee D)\sigma}$$

If: $\sigma = \mathrm{mgu}(u, v)$, $p_u\sigma$ and $\neg p_v\sigma$ are selected.

$$\frac{p_u \vee p_v \vee C}{(p_u \vee C)\sigma} \qquad \frac{\neg p_u \vee \neg p_v \vee C}{(\neg p_u \vee C)\sigma}$$

If: $\sigma = \mathrm{mgu}(u, v)$, $p_u\sigma$ or $\neg p_u$ is selected.

We denote by $\mathcal{R}(S)$ the set of clauses that can be deduced from (pairwise index variable disjoint renamings of) clauses in S by resolution or factorization (in one step).

A clause C is *redundant* w.r.t. a clause set S (written $C \sqsubseteq S$) iff for every ground substitution σ of domain $var(C)$, there exist $D_1, \ldots, D_n \in S$ and $\sigma_1, \ldots, \sigma_n$ such that $D_1\sigma_1, \ldots, D_n\sigma_n \models C\sigma$ and $D_1\sigma_1, \ldots, D_n\sigma_n \preceq C\sigma$. If S is a clause set, we write $S \sqsubseteq S'$ iff every clause in S is redundant w.r.t. S'.

A clause set S is *saturated* iff $\mathcal{R}(S) \sqsubseteq S$.

A *derivation* from a clause set S is a sequence of clauses $C_1, \ldots, C_n$ such that for every $i \in [1, n]$, $C_i \in S \cup \mathcal{R}(\{C_1, \ldots, C_{i-1}\})$. We write $S \vdash C$ iff there exists a derivation $C_1, \ldots, C_n$ from S such that $C = C_n$.

A derivation *of* S is a derivation containing every clause in S. For instance $p_0, \neg p_x \vee p_{succ(x)}, p_{succ(0)}$ is a derivation of $\{p_{succ}(0)\}$ from $\{p_0, \neg p_x \vee p_{succ(x)}\}$.

The notion of refutation differs from the usual one: since no rule can be applied on the equational part of the clauses, these literals cannot be eliminated:

Definition 3 A *refutation* of S is a derivation from S of a finite unsatisfiable set of purely equational clauses. ♢

Notice that the satisfiability of this set can be tested by Proposition 1.

3.2 Basic properties of the calculus

It is very easy to check that the above rules are sound, i.e. the conclusion are logical consequences of the premises.

Proposition 4 *Let S be a set of clauses. $S \models \mathcal{R}(S)$.*

Proof. Immediate. ∎

We now prove a weak form of refutational completeness. We first notice that the calculus is obviously complete for parameter-free clause sets:

Proposition 5 *Let S be a saturated set of parameter-free clauses. If S is unsatisfiable then $\square \in S$.*

PROOF. When the clauses contain no equational literals, our Resolution calculus coincides with the usual one. Hence the result. ∎

Theorem 6 *Let S be a saturated set of clauses. If S is unsatisfiable then there exists an unsatisfiable set of purely equational clauses S' such that $S' \subseteq S$.*

PROOF. Let I be an interpretation of the parameters in S. Let S_g be the set of ground instances of the clauses in S. Let S_I be the set of clauses obtained from S_g by:

- Deleting every clause containing an equational literal that is true in I.
- Removing all equational literals in the remaining clauses.

By construction, S_I contains no equational literal, hence no parameter. If S_I has a model J then obviously $I \cup J$ is a model of S. Thus S_I is unsatisfiable. We now check that S_I is saturated.

Since S is saturated, so is S_g (since the notion of saturatedness only depends on ground instances). Let $p_u \vee C$ and $\neg p_u \vee D$ be two clauses in S_I on which the Resolution rule applies. By definition S_g contains two clauses $p_u \vee C \vee C'$ and $\neg p_u \vee D \vee D'$ where C', D' are purely equational clauses that are false in I. Since S_g is saturated, there exist n clauses $E_1, \ldots, E_n \in S_g$ such that $E_1, \ldots, E_n \models C \vee D \vee C' \vee D'$ and $E_1, \ldots, E_n \prec C \vee D \vee C' \vee D'$. Assume w.l.o.g. that there is a $k \in [0, n]$ such that for every $i \in [1, k]$, $I \models E_i^{eq}$ and for every $i \in [k+1, n]$, $I \not\models E_i^{eq}$ (i.e. the k clauses such that $I \models E_i^{eq}$ are assumed to be at the beginning of the sequence $E_1, \ldots, E_n$). By definition, we have $E_{k+1}^{in}, \ldots, E_n^{in} \in S_I$.

We show that $E_{k+1}^{in}, \ldots, E_n^{in} \models C \vee D$. Let J be an interpretation satisfying $E_{k+1}^{in}, \ldots, E_n^{in}$. Let J' be the interpretation defined as follows: $J'(n) \stackrel{\text{def}}{=} I(n)$ and $J'(A) \stackrel{\text{def}}{=} J(A)$ for every ground atom A. By definition, $J' \models E_i^{eq}$, for every $i \in [1, k]$ thus $\forall i \in [1, k], J' \models E_i$. Moreover, since $J \models E_i^{in}$, for every $i \in [k+1, n]$ we also have $\forall i \in [k+1, n], J' \models E_i$. Therefore $J' \models E_1, \ldots, E_n$, hence $J' \models C \vee D \vee C' \vee D'$. But $I \not\models C' \vee D'$, thus $J \models C \vee D$.

Moreover, since $E_1, \ldots, E_n \prec C \vee D \vee C' \vee D'$, we must have $E_{k+1}^{in}, \ldots, E_n^{in} \prec C \vee D$ (since index literals and equational literals are not comparable). Consequently, $C \vee D$ is redundant in S_I.

By the same reasoning we can prove that any factor of a clause in S_I is redundant.

Thus $\mathcal{R}(S_I) \sqsubseteq S_I$ and S_I is saturated. Thus by Proposition 5, $\square \in S_I$. Consequently, S contains a purely equational clause C_I that is false in I. This holds for every interpretation I, thus S contains a set of purely equational clauses S' such that for every interpretation I of the parameters in S', $I \not\models S'$. This implies that S' is unsatisfiable. ∎

Theorem 6 does *not* imply semi-decidability because S' may well be infinite:

Example 7 Let $S = \{n \not\approx x \vee p_x, p_y \vee \neg p_{succ(y)}, \neg p_0\}$. The reader can easily check that S is unsatisfiable and that our calculus generates an infinite number of purely equational clauses of the form $n \not\approx succ^i(0)$:

1	$n \not\approx x \vee p_x$	(given)
2	$p_y \vee \neg p_{succ(y)}$	(given)
3	$\neg p_0$	(given)
4	$n \not\approx succ(y) \vee p_y$	(resolution $1, 2$, $x \mapsto succ(y)$)
$4'$	$n \not\approx succ(y') \vee p_{y'}$	(renaming, 4)
5	$n \not\approx 0$	(resolution $1, 3$, $x \mapsto 0$)
6	$n \not\approx succ(0)$	(resolution $4, 3$, $y \mapsto 0$)
7	$n \not\approx succ(succ(y)) \vee p_y$	(resolution $4', 2$, $y' \mapsto succ(y)$)
8	$n \not\approx succ(succ(0))$	(resolution $7, 3$, $y \mapsto 0$)
	$\ldots$	

This set of clauses is clearly unsatisfiable, but every finite subset it contains is satisfiable. ♣

4 Restricting the language

We restrict ourselves to a particular class of clause sets. This class is very convenient from a theoretical point of view and, as we shall see in Section 8, it is expressive enough for our purpose.

We shall firstly assume that the clauses contain at most one variable (e.g. we discard clauses such as $p_x \vee p_y$ where x, y are distinct variables). Furthermore, the clauses containing a variable should not contain any ground term (i.e. no occurrence of 0). For instance, a clause such as $p_x \vee q_0$ is forbidden.

Definition 8 Let $t \in \{0\} \cup \mathcal{V}$. A clause C is a *t-clause* iff every atom occurring in it is of the form $p_{succ^i(t)}$ for some $i \in \mathbb{N}$. ◊

Next, we assume that the non equational part of the clause is of depth 0 or 1 (the depth of the equational part is arbitrary). Finally, we strongly restrict the form of the equational part of the clauses:

Definition 9 A clause C is *normalized* iff there exists a term $t \in \mathcal{V} \cup \{0\}$ such that the following conditions hold:

- C^{eq} is either empty or of the form $n \not\approx succ^k(t)$.
- C^{in} is a t-clause of depth 0 or 1.
- If $t = 0$ then $depth(C^{in}) = 0$.

A set of clauses S is *normalized* iff the following conditions hold:

1. S contains at most one parameter (always denoted by n in the following)
2. Every clause in S is normalized. ◊

For instance $p_x \vee \neg p_{succ(x)}$, $n \not\approx succ(succ(x)) \vee p_x$ or $n \not\approx 0 \vee q_0$ are normalized, but $n \not\approx 0 \vee p_{succ(0)}$, $n \not\approx succ(x) \vee \neg p_0 \vee q_x$ or $p_{succ(succ(x))}$ are not.

Definition 10 The *level* of a normalized clause is an element of $\mathbb{N} \cup \{\bot\}$, defined as follows:

- If $C^{eq} = \Box$ then the level of C is $\bot$.
- If $C^{eq} \neq \Box$ then $level(C) \stackrel{\text{def}}{=} level(C^{eq}) - level(C^{in}) + 1$.

$S|_i$ denotes the set of clauses in S of level i (where $i \in \mathbb{N} \cup \{\bot\}$). If I is a subset of $\mathbb{N} \cup \{\bot\}$, $S|_I$ denotes the set of clauses whose levels are in I. $\Diamond$

For instance, the levels of $n \not\approx succ(0) \vee p_0$, $n \not\approx succ(succ(x)) \vee p_{succ(x)}$ and $n \not\approx succ(x) \vee p_x$ are all equal to 2. For any parameter-free clause C, the level of $n \not\approx 0 \vee C$ is 1, the one of $n \not\approx succ(0) \vee C$ is 2, etc. (because C necessarily has depth 0 in this case). The level of $n \not\approx succ(succ(x)) \vee p_x$ is 3 (as the one of its instance $n \not\approx succ(succ(succ(x))) \vee p_{succ(x)}$), the one of $n \neq succ(succ(x)) \vee p_{succ(x)}$ is 2.

Intuitively, a clause of level $\bot$ expresses either some universal property such as $\forall x \neg p_x \vee p_{succ(x)}$ or a ground property, e.g. p_0, $p_{succ(0)}$ etc. A clause of level distinct from $\bot$ is of the form $n \not\approx succ^k(t) \vee C$ where the only indices occurring in C are t or $succ(t)$. It can be viewed as an implication: $n \approx succ^k(t) \Rightarrow C$. If $t = 0$ then the clause states a ground property that must be true if n is equal to $succ^k(0)$. If t is a variable x, the clause expresses an assertion that must hold for the natural number $x = n - l$, assuming that $n \geq l$. This could be written $n \geq l \Rightarrow C\{x \mapsto n - l\}$ but of course this last formula is not a clause.

Remark 11 *For any interpretation I, the level of a clause C (if distinct from $\bot$) is always equal to $I(n) + 1 - i$, where i denotes the maximal index of the (only) ground instance of C that is relevant in I (this instance depends on the value of n).*

For example, consider the clause $C : n \not\approx succ(x) \vee p_x$. The level of C is 2. The only ground instance of C that is relevant for the interpretation $I = \{n \mapsto k\}$ is of the form $n \not\approx succ(succ^{k-1}(0)) \vee p_{succ^{k-1}(0)}$ (since $n \not\approx succ(x)$ is true in I if $x \neq succ^{k-1}(0)$) where the value of n is $k + 1$. We have $2 = k + 1 - (k - 1)$.

We order the levels in $\mathbb{N} \cup \{\bot\}$ by using the usual ordering on natural numbers and by assuming that $\bot < i$ for every $i \in \mathbb{N}$. Thus if i is a natural number then $[\bot, i]$ denotes the set $\{\bot\} \cup [0, i]$.

Proposition 12 *For every $l \in \mathbb{N} \cup \{\bot\}$, the number of normalized clauses of level l (on a given finite signature) is finite (up to renaming and up to duplication of literals).*

Proof. A normalized clause is of the form $n \not\approx t \vee C$ or C where C is a t-clause of depth 0 or 1. Furthermore, if t is fixed and if Ω is finite, the number of distinct t-clauses of depth 0 or 1 is finite (up to duplication of literals), since every literal in C is of the form p_t, $\neg p_t$, $p_{succ(t)}$ or $\neg p_{succ(t)}$, for some $p \in \Omega$. ∎

Definition 13 A set of clauses is *k-normalized* if it is normalized and if the level of every clause in S is in $[\bot, k]$. $\Diamond$

Section 8 gives hints on the expressive power of normalized clause sets.

The following propositions states straightforward properties of t-clauses:

Proposition 14 *If C is a t-clause (where $t \in \mathcal{V} \cup \{0\}$) and if σ is flat then $C\sigma$ is a $t\sigma$-clause.*

Proposition 15 *If C, D are two t-clauses then $C \vee D$ is a t-clause.*

Proposition 16 *If C is a $succ(t)$-clause of depth 0 then C is a t-clause.*

The next lemma states that the class of normalized clauses is preserved by the inference rules.

Lemma 17 *If S is set of normalized non valid clauses then any non valid clause in $\mathcal{R}(S)$ is normalized.*

PROOF. Let C be a (non valid) clause deduced from two normalized clauses D_1 and D_2 by resolution. By definition, D_1^{in} and D_2^{in} are of the form $p_{u_1} \vee D_1'$ and $\neg p_{u_2} \vee D_1'$ where u_1 and u_2 are unifiable. Let $D_i'' \stackrel{\text{def}}{=} D_i^{eq}$. Since S is normalized, there exists a term $t_i \in \mathcal{V} \cup \{0\}$ such that $l^i_{u_i} \vee D_i'$ is a t_i-clause. Furthermore, D_i'' is either empty or of the form $n \not\approx succ^{k_i}(t_i)$. Then C is of the form $(D_1' \vee D_2' \vee D_1'' \vee D_2'')\sigma$ where $\sigma = \text{mgu}(u_1, u_2)$. By definition, u_i is either t_i or $succ(t_i)$. Thus σ is of one of the following forms: $\emptyset$, $\{t_2 \mapsto t_1\}$, $\{t_2 \mapsto succ(t_1)\}$ (with $t_2 \in \mathcal{V}$ and $t_1 \in \mathcal{V} \cup \{0\}$) or $\{t_1 \mapsto t_2, t_1 \mapsto succ(t_2)\}$ (with $t_1 \in \mathcal{V}$ and $t_2 \in \mathcal{V} \cup \{0\}$).

Any atom occurring in $D_i'\sigma$ is of the form $q_{t_i\sigma}$ or $q_{succ(t_i\sigma)}$.

If σ is empty or of the form $t_2 \mapsto t_1$ or $t_1 \mapsto t_2$ then $t_1\sigma = t_2\sigma$ (notice that if σ is empty then u_1, u_2 are ground thus, since D_1, D_2 are normalized we must have $u_1 = u_2 = t_1 = t_2 = 0$). Since σ is flat, by Proposition 14, $D_1'\sigma$ is a $t_i\sigma$-clause. By Proposition 15, $D_1'\sigma \vee D_2'\sigma$ is a $t_1\sigma$-clause. D_i'' is either empty or of the form $n \not\approx succ^{k_i}(t_i)$. Assume that D_1'' and D_2'' are not empty and that $k_1 \neq k_2$. Then C contains the disjunction $n \not\approx succ^{k_1}(t_1\sigma) \vee n \not\approx succ^{k_2}(t_2\sigma)$. Since $t_1\sigma = t_2\sigma$ and $k_1 \neq k_2$, $succ^{k_1}(t_1\sigma)$ and $succ^{k_2}(t_2\sigma)$ are not unifiable, hence C is valid. Otherwise, $D_1'' \vee D_2''$ is either empty or of the form $n \not\approx succ^{k_1}(t_1\sigma)$. Thus C is normalized.

Assume that σ is $t_2 \mapsto succ(t_1)$. If $D_2'\sigma$ contains a atom of the form $q_{succ(t_2)}$ then we have $p_{t_2} \lhd q_{succ(t_2)}$ thus by definition of the selection function, the literal $\neg p_{t_2}$ would not be selected, which is impossible. Hence every atom in $D_2'\sigma$ is of the form $q_{t_2\sigma} = q_{succ(t_1)}$, hence $D_2\sigma'$ is a t_1-clause. $D_1'\sigma$ is obviously a t_1-clause (since σ is the identity on D_1'), thus by Proposition 15, $D_1'\sigma \vee D_2'\sigma$ is a t_1-clause. Assume that D_1'' and D_2'' are not empty and that $k_1 \neq k_2 + 1$. Then C contains the disjunction $n \not\approx succ^{k_1}(t_1\sigma) \vee n \not\approx succ^{k_2}(t_2\sigma)$. Since $t_2\sigma = succ(t_1)$ and $k_1 \neq k_2 + 1$, $succ^{k_1}(t_1\sigma)$ and $succ^{k_2}(t_2\sigma)$ are not unifiable, hence C is valid. Otherwise, $D_1'' \vee D_2''$ is either empty or of the form $n \not\approx succ^{k_2+1}(t_1)$. Thus C is normalized.

The proof is similar if σ is $t_1 \mapsto succ(t_2)$.

Now assume that C is deduced by factorisation, from a clause $p_u \vee p_v \vee C'$ (resp. $\neg p_u \vee \neg p_v \vee C'$). By definition, u and v are either ground or of the form $succ^i(x)$ and $succ^j(x)$ for some variable x. Since $p_u \vee p_v \vee C'$ (resp. $\neg p_u \vee \neg p_v \vee C'$) is non valid, u and v must be unifiable. Thus we must have $u = v$ hence the mgu of u and v is empty. Hence $C = p_u \vee C'$ (resp. $C = \neg p_u \vee C'$) is normalized. Factorisation cannot be applied in a non trivial way on normalized clauses (it simply removes a literal). ∎

Lemma 18 relates the level of any clause to that of its parents.

Lemma 18 *Let S be a set of normalized clauses. Let C be a non valid clause of level $j \in \mathbb{N}$ in $\mathcal{R}(S)$ deduced from two parent clauses D_1, D_2 in S of level k_1, k_2 respectively. The following conditions hold:*

- $k_1, k_2 \in \{\bot, j, j-1\}$.
- *If C is ground then $k_1, k_2 \in \{\bot, j\}$.*
- *If $k_1 \neq \bot$ and $k_2 \neq \bot$ then $k_1 = k_2$.*

Proof. By definition, there exists a term $t \in \mathcal{V} \cup \{0\}$ such that C is of the form $n \not\approx succ^{j-1+\epsilon}(t) \vee C'$ and C' is a t-clause of depth ϵ (notice that $j \neq \bot$ by hypothesis). Similarly, If D_i is not parameter-free, then D_i is of the form $n \not\approx succ^{k_i-1+\epsilon_i}(t_i) \vee D_i'$ where D_i' is a t_i'-clause of depth ϵ_i. We must have $succ^{j-1+\epsilon}(t) = succ^{k_i-1+\epsilon_i}(t_i)\sigma$ where σ is the mgu of two terms u_1 and u_2 occurring in selected literals in D_1 and D_2 respectively.

u_1 and u_2 are of the form $0, x_i$ or $succ(x_i)$ where x_i is a variable, thus σ is either empty (if $u_1 = u_2 = 0$) or of the form $x_1 \to 0$ (if $u_1 = x_1$ and $u_2 = 0$) or $x_2 \to 0$ (if $u_1 = 0$, $u_2 = x_2$) or $x_1 \to x_2$ (if either $u_1 = x_1$ and $u_2 = x_2$ or $u_1 = succ(x_1)$ or $u_2 = succ(x_2)$) or $x_1 \to succ(x_2)$ (if $u_1 = x_1$ and $u_2 = succ(x_2)$) or $x_2 \to succ(x_1)$ (if $u_1 = succ(x_1)$ and $u_2 = x_2$). Several cases must be distinguished according to the form of σ.

- If σ is empty then we must have $u_1 = u_2 = 0$. Then D_1, D_2 are ground, hence C is also ground. Furthermore, if D_i is not parameter-free, we have $j - 1 + \epsilon = k_i - 1 + \epsilon_i$. By definition of normalized clauses we have $\epsilon = \epsilon_i = 0$, thus $j = k_i$.
- If σ is of the form $x_1 \mapsto x_2$ or $x_1 \mapsto 0$ or $x_2 \mapsto 0$ then D_1^{in} and D_2^{in} must be of the same depth, thus $\epsilon_1 = \epsilon_2$. If D_i is not parameter-free, we have $j - 1 + \epsilon = k_i - 1 + \epsilon_i$, thus $k_i = j + \epsilon - \epsilon_i$. If $\epsilon = 1$ then this means that C' is of depth 1. But every literal in C' occurs in $D_j'\sigma$ for some $j = 1, 2$, thus $D_j'\sigma$ is of depth 1 for some $j = 1, 2$. Since σ is flat, D_j is of depth 1, hence $\epsilon_j = 1$. Since $\epsilon_1 = \epsilon_2$ we deduce that $\epsilon_i = 1$, hence $\epsilon - \epsilon_i \leq 0$ and $k_i \in \{0, j, j-1\}$.

  If C is ground then we must have $u_1 = 0$ or $u_2 = 0$, hence $\epsilon_1 = \epsilon_2 = 0$. Moreover, $\epsilon = 0$ by definition of the notion of normalized clause. Thus $k_i = j$.

Finally if D_1 and D_2 are both not parameter-free we remark that, since u_1 and u_2 are of same depth we must have $\epsilon_1 = \epsilon_2$ hence $k_1 = k_2$.

- If σ is of the form $x_1 \mapsto succ(x_2)$ then $u_1 = x_1$ and $u_2 = succ(x_2)$. In this case C is not ground. If D_1 is not parameter-free, we have $j - 1 + \epsilon = k_1 - 1 + \epsilon_1 + 1$, thus $k_1 = j + \epsilon - \epsilon_1 - 1$. Moreover we have $\epsilon_1 = 0$ since $u_1 = x_1$ is maximal. Thus $k_1 = j + \epsilon - 1 \in \{j - 1, j\}$. If D_2 is not parameter-free, we have $j - 1 + \epsilon = k_2 - 1 + \epsilon_2$, thus $k_2 = j + \epsilon - \epsilon_2$. Moreover we have $\epsilon_2 = 1$ since $u_2 = succ(x_2)$ occurs in D_2. Thus $k_2 = j - 1 + \epsilon \in \{j - 1, j\}$.
- The case where σ is of the form $\{x_2 \mapsto succ(x_1)\}$ is symmetric. ∎

5 Loop Detection

In this section we refine the calculus proposed in Section 3 by defining a *loop detection rule* that is capable of pruning infinite derivations. To help the reader to grasp the next definitions and lemmata, we first provide an informal high level description of the rule. Let S be a clause set. The set of clauses $S' = \{C \mid S \vdash C\}$ generated from S can be partitioned into an infinite sequence of clause sets $S'|_\perp$, $S'|_1, \ldots, S'|_k, \ldots$, where for every $k \in \mathbb{N} \cup \{\perp\}$, $S'|_k$ contains exactly the clauses of level k in S' (see Definition 10) . Notice that by Lemma 18, the clauses in $S'|_k$ (where $k > 0$) must be generated either from clauses in $S'|_k \cup S'|_\perp$ or from $S'|_{k-1} \cup S'|_\perp$ (except of course those that already occur in the original clause set S). By definition of the notion of level, each of these sets $S'|_k$ can be viewed as a formula of the form $n \not\approx succ^k(x) \vee S_k$, where S_k is a conjunction of x-clauses[1]. Since the set of x-clauses is finite there must exist, by the pigeonhole argument, two natural numbers i and $j > 0$ such that $S_i = S_{i+j}$. This means that the set of clauses generated at level i is equivalent to the set of clauses generated at level $i + j$, up to a shift on the parameter n: indeed if $S_i = S_{i+j}$ then $n \not\approx succ^{i+j}(x) \vee S_{i+j}$ is equivalent to $n \not\approx succ^i \vee S_i$, up to a shift $n \mapsto n + j$. We shall show that, under some particular conditions, the existence of such a "cycle" in the derivation allows one to derive an upper bound on the value of the parameter n: if S is satisfiable then it has a model I such that $I(n) < i + j$. The intuition is that if a model I such that $I(n) > i + j$ exists, then one can obtain another model J such that $J(n) = I(n) - j$ by applying the translation $n \mapsto n - j$ on I. This implies that the clause $n < i + j$ (written $n \not\approx succ^{i+j}(x)$) can be derived. Then it is easy to see that every clause of level strictly greater than $i + j$ is redundant w.r.t. $n < i + j$ (since $n \not\approx succ^{i+j}(t) \vee C$ is obviously subsumed by $n \not\approx succ^{i+j}(x)$). Consequently, once the *pruning clause* $n < i + j$ has been generated, only finitely many non redundant clauses can be deduced.

Obviously, this result does not hold for any derivation: for instance if the initial clause set S contains clauses of arbitrary level, then the value of parameter must also be arbitrary. Now we give the formal definition of the conditions that must be satisfied by S. We start by slightly restricting the notion of redundancy:

[1] to simplify the presentation we assume that every clause $C \in S_k$ is of depth 1.

Definition 19 A clause C is *level-redundant* w.r.t. S (written $C \sqsubseteq_l S$) iff for every ground substitution σ of domain $var(C)$ there exist n clauses $C_1, \ldots, C_n$ such that:

- If C is non ground then $C_1, \ldots, C_n$ are non ground.
- $C_1, \ldots, C_n$ are either of level $\bot$ or of the same level as C.
- There exist n substitutions $\theta_1, \ldots, \theta_n$ such that $C_1\theta_1, \ldots, C_n\theta_n \models C\sigma$ and $C_1\theta_1, \ldots, C_n\theta_n \preceq C\sigma$.

If S' is a set of clauses, we write $S' \sqsubseteq_l S$ iff $\forall C \in S, C \sqsubseteq_l S$. $\Diamond$

For instance, p_x is redundant w.r.t. $\{p_0, p_{succ(x)}\}$ but not level-redundant (since p_0 is ground and p_x is not). $n \not\approx succ(x) \vee p_x$ is redundant w.r.t. $\{p_x, \neg p_x\}$ but not level-redundant since $level(n \not\approx succ(x) \vee p_x) = 2$ and $level(p_x) = level(\neg p_x) = \bot < 2$. Notice that every tautological clause and every clause subsumed by a clause in S is level-redundant w.r.t. S, thus in practice these two notions coincide.

Definition 20 A set of clauses S is *saturated up to level* i iff $\mathcal{R}(S|_{[\bot,i[}) \sqsubseteq_l S$.$\Diamond$

If S is a set of clauses and i is a natural number, $shift(S, i)$ denotes the set of clauses of the form $n \not\approx succ^{i+j}(x) \vee C$, where $x \in \mathcal{V}$ and $n \not\approx succ^j(x) \vee C \in S$.

Example 21 Consider the clause set

$$S = \{n \not\approx succ(succ(x)) \vee \neg p(x) \vee p(succ(x)), n \not\approx succ(x) \vee q(x), n \not\approx 0 \vee p(x), \neg r(x)\}.$$

We have $shift(S, 0) = \{n \not\approx succ(succ(x)) \vee \neg p(x) \vee p(succ(x)), n \not\approx succ(x) \vee q(x)\}$, $shift(S, 1) = \{n \not\approx succ(succ(succ(x))) \vee \neg p(x) \vee p(succ(x)), n \not\approx succ(succ(x)) \vee q(x)\}$, $shift(S, 2) = \{n \not\approx succ(succ(succ(succ(x)))) \vee \neg p(x) \vee p(succ(x)), n \not\approx succ(succ(succ(x))) \vee q(x)\}$ etc. Notice that $\neg r(x)$ or $n \not\approx 0 \vee p(x)$ cannot occur in $shift(S, i)$ because they contain no literal of the form $n \not\approx succ^j(x)$, where $x \in \mathcal{V}$. ♣

Lemma 22 will set the foundations for defining the loop detection rule. This lemma applies when we detect that the clauses of a certain level i in S are logically entailed by the ones of a level $i + j > i$, up to a shift on the parameter n. This means that any model of $S|_{i+j}$ is also a model of $S|_i$, up to a shift $n \mapsto n - j$. If, moreover, we assume that S is saturated up to level i, this implies as we shall see that for any model I of S such that $I(n) \geq i + j$, one can construct a model J of S such that $J(n) = I(n) - j < I(n)$. To get a model of $S|_{[i,\infty[}$, it suffices to apply the translation $n \mapsto n - j$ on the model I, namely to fix the truth values of the variables $p_i, p_{i+1}, p_{i+2}, \ldots$ in J as the ones of $p_{i+j}, p_{i+j+1}, p_{i+j+2}, \ldots$ in I; then, since S is saturated up to level i, it is possible to show that this interpretation can be extended into a model of S. Thus, satisfiability is preserved when we state that the value of n is strictly lower than $i + j$. As explained before, adding this assertion into S makes every clause of level greater than $i + j$ redundant since it is subsumed by $n < i + j$.

Lemma 22 *Let S be a set of normalized clauses of parameter n and let i, j be natural numbers satisfying the following conditions:*

1. *S is saturated up to level i.*
2. *$S|_i \cup S|_\bot \models S|_{[i,\infty[}$*
3. *$S|_i$ contains no ground clause.*
4. *$j \neq 0$.*
5. *$S|_\bot \cup S|_{i+j} \models \mathit{shift}(S|_i, j)$.*

Then: S is satisfiable iff $S \cup \{n < i + j\}$ is.

Proof. Let I be an interpretation satisfying S. W.l.o.g. we assume that the value of n is minimal (w.r.t. the usual ordering on natural numbers), i.e. for every interpretation J such that $J(n) < I(n)$ we have $J \not\models S$.

Assume that $I \not\models \{n < i + j\}$ (i.e. $I(n) \geq i + j$ which implies that $I(n) > 0$ since $j \neq 0$ by assumption).

We construct an interpretation J as follows.

- $J(n) \stackrel{\text{def}}{=} I(n) - j$. By Condition 4 this implies that $J(n) < I(n)$ thus $J \not\models S$.
- The value of the ground atom p_k (for $k \in \mathbb{N}$) is defined by induction on the ordering $\prec$:
  - If $k \leq I(n) - i - j + 1$ then $J(p_k) \stackrel{\text{def}}{=} I(p_k)$.
  - If $k > I(n) - i + 1$, then suppose $J(q_{k'})$ is already defined for every literal $q_{k'}$ lower than p_k. Thus, for any clause made only of such literals, its value under J is completely defined. Notice also that the value of every purely equational clause only depends on $J(n)$ and is thus also completely defined. Consequently, we can define $J(p_k)$ as follows: if $S|_{[\bot,i[}$ contains a clause which has an instance of the form $p_k \vee C$, where all indexed atoms of C are lower than p_k and $J \not\models C$, then we set $J(p_k)$ to `true`. Otherwise we set it to `false`.

Let $S' = S|_{[\bot,i]}$. By Condition 2 we have $S|_i \cup S|_\bot \models S|_{[i,\infty[}$ thus $S' \models S$ (since by definition $S = S|_{[\bot,i]} \cup S|_{[i,\infty[}$). We now show that $J \models S'$ (thus contradicting the fact that $J \not\models S$).

Let D be a clause in S' and let θ be a ground substitution such that $J \not\models D\theta$. By definition θ is either empty (if D is ground) or of the form $x \mapsto k$ where x is the unique variable in D.

First, assume that $D \in S|_i$. By Condition 5 $S|_\bot \cup S|_{i+j} \models \mathit{shift}(S|_i, j)$. Since $I \models S \supseteq S|_\bot \cup S|_{i+j}$ this implies that $I \models \mathit{shift}(S|_i, j)$. By definition, D is of the form $n \not\approx \mathit{succ}^{i-1+\epsilon}(t) \vee D'$, where D' is a t-clause of depth ϵ. By condition 3 D is not ground, thus t is a variable x. Then $\mathit{shift}(S|_i, j)$ contains the clause

$n \not\approx succ^{i-1+\epsilon+j}(x) \vee D'$ and $I \models n \not\approx succ^{i-1+\epsilon+j}(x) \vee D'$. Since $J \not\models D\theta$ we must have $J(n) = i - 1 + \epsilon + k$. Thus $I(n) = J(n) + j = i - 1 + \epsilon + k + j$, hence (since $I \models n \not\approx succ^{i-1+\epsilon+j}(x)\theta \vee D'\theta$), we have $I \models D'\theta$. The indices occurring in D' are either x or $succ(x)$. Thus the indices occurring in $D'\theta$ are either k or $k+1$. Furthermore if an index $k+1$ occurs in $D'\theta$ then D is of depth 1, thus $\epsilon = 1$ (otherwise $\epsilon = 0$). Hence the maximal index that can occur in D' is $I(n) - i - j + 1$. By definition of J, I and J coincide on the propositions p_l such that $l \leq I(n) - i - j + 1$. Thus we have $J \models D'\theta$, hence $J \models D\theta$, a contradiction.

Now, assume that $D \in S|_{[\bot,i[}$. W.l.o.g. we assume that $D\theta$ is the least instance (w.r.t. $\prec$) of a clause in $S|_{[\bot,i[}$ such that $J \not\models D\theta$. Note that D cannot be empty (otherwise S would be unsatisfiable). If D is purely equational then, since D cannot be valid, D must be of the form $n \not\approx succ^l(t)$ for some $l < i$ and $t \in \mathcal{V} \cup \{0\}$ (since $D \in S|_{[0,i[}$ and D is normalized). Then D is equivalent to either $n < l$ (if $t \in \mathcal{V}$) or $n \neq l$ (if $t = 0$), where $l < i$. In the first case, we have $I \not\models D$ which is impossible since I is a model of S and $D \in S' \subseteq S$. In the second case, since $l < i$ and $J(n) = I(n) - j \geq i$ we have $J \models D$.

Thus D contains at least one index literal. We denote by l_u the maximal index literal in D (there is only one maximal index literal since D contains only one variable and since $\prec$ is total). D is of the form $l_u \vee D' \vee D''$, where D' is parameter-free, D'' is purely equational (with possibly $D', D'' = \Box$), and for any literal $l \in D'\theta$, $l \preceq l_u\theta$. By definition of the ordering, this implies that the index of every literal $l \in D'\theta$ is smaller or equal to $u\theta$.

We distinguish two cases according to the value of $u\theta$.

- If $u\theta \leq I(n) - i - j + 1$, then every index in $D'\theta$ must be lower or equal to $I(n) - i - j + 1$. By definition of J, this implies that I and J coincide on $(l_u \vee D')\theta$. If D'' is empty, then we have $D = l_u \vee D'$, and the proof is completed since by hypothesis $J \models D\theta$, thus $J \models (l_u \vee D')\theta$ and $I \models (l_u \vee D')\theta$. Otherwise, D'' is of the form $n \not\approx succ^l(t)$. Since $J \not\models D''\theta$ we have $J(n) = l + k$ for some $k \in \mathbb{N}$ (if $t = x$) or $J(n) = l$ (if $t = 0$). Because D'' is not empty, we cannot have $D \in S|_\bot$ thus $D \in S|_{[0,i[}$, hence $l < i$. Thus we cannot have $J(n) = l$ (since by hypothesis $I(n) > i + j$ and by definition of J, $J(n) = I(n) - j$, thus $J(n) \geq i$), hence we are in the first case and $k > J(n) - i$. u is either x or $succ(x)$. Thus $u\theta$ is either k or $k + 1$. Furthermore, if u is x then $l_u \vee D'$ is an x-clause of depth 0 (otherwise l_u would not be maximal), hence $l + 1 < i$ and in this case we have $k > J(n) - i + 1$. Thus in every case $u\theta \geq J(n) - i + 1 \geq I(n) - j - i + 1$ which contradicts our assumption.

- Now assume that $u\theta > I(n) - i - j + 1$. Since $D\theta$ is the minimal instance of a clause in $S|_{[\bot,i[}$ that is false in J and since S is saturated up to level i we can assume that $D'\theta \prec l_u\theta$ (indeed, every literal $D'\theta$ must be smaller than $l_u\theta$, furthermore, if $l_u\theta$ occurs in $D'\theta$ then the factorization rule applies on D and generates a clause with a strictly smaller instance that is false in J). By definition of J this implies that l_u is a negative literal $\neg p_u$. But then since $J(p_{u\theta}) = \texttt{true}$, $S|_{[\bot,i[}$ contains a clause of the

form $p_v \vee E' \vee E''$ such that E' is parameter-free, E'' is purely equational, there exists a substitution θ' such that $v\theta' = u\theta$, $J \not\models (E' \vee E'')\theta'$ and $\forall l \in E', l\theta' \prec p_{v\theta'}$.

The resolution rule applies on D and $p_v \vee E' \vee E''$ and yields a clause of the form $(E' \vee E'' \vee D' \vee D'')\sigma$, where σ is the mgu of u and v. An instance of this clause is $E''\theta' \vee D''\theta \vee E'\theta' \vee D'\theta$. This clause is strictly smaller than $D\theta$ and false in J. Since S is saturated up to level i, $(E' \vee E'' \vee D' \vee D'')\sigma$ is level-redundant in S. Thus there exist m clauses $C_1, \ldots, C_m \in S$ and m substitutions $\sigma_1, \ldots, \sigma_n$ such that $C_1\sigma_1, \ldots, C_m\sigma_m \preceq (E' \vee E'' \vee D' \vee D'')\sigma$ and $C_1\sigma_1, \ldots, C_m\sigma_m \models (E' \vee E'' \vee D' \vee D'')\sigma$. By Lemma 18, $(E' \vee E'' \vee D' \vee D'')\sigma$ is of level at most i, thus so are $C_1, \ldots, C_m$. If $C_1, \ldots, C_m$ are of level i then we must have $J \models C_1, \ldots, C_m$ by the first item above. Otherwise, by the minimality condition on $D\theta$, we have $J \models C_1\sigma_1, \ldots, C_m\sigma_m$. In every case we deduce that $J \models (E' \vee E'' \vee D' \vee D'')\sigma$ which is impossible. ■

Lemma 22 by itself is not sufficient to define a suitable loop detection rule. Indeed, Condition 2 is hard to check and Condition 3 is not guaranteed. Fortunately, we show that these two conditions always hold for set of clauses that are obtained from a k-normalized clause set:

Definition 23 A set of clauses S is *k-reducible* iff there exists a k-normalized set of clauses S' such that for every clause $C \in S$ there exists a derivation $C_1, \ldots, C_n$ from S' such that $C_n = C$ and $C_1, \ldots, C_n$ are level-redundant w.r.t. S. ◊

Lemma 24 *Let S be a k-reducible set of normalized clauses. For every $i \geq k$, $S|_i \cup S|_\perp \models S|_{i+1}$. Hence $S|_i \cup S|_\perp \models S|_{[i,\infty[}$.*

Proof. Since S is k-reducible, there exists a k-normalized set of clauses S' such that C is derivable from S'. Furthermore, every clause in the derivation is level-redundant w.r.t. S.

We show, by induction on the length of the derivation that for every clause C' of level $i+1$ occurring in this derivation, we have $S|_i \cup S|_\perp \models C'$. Since C' is of level $i+1 > k$, the length of the derivation is greater than 0. C' is deduced from two clauses D_1, D_2 (resp. from a unique clause D_1) by resolution (resp. factorization). By Lemma 18, D_1, D_2 (resp. D_1) are either in $S|_\perp$ or of level i or $i+1$. If D_1 is of level i or $\perp$ then $S|_i \cup S|_\perp \models D_1$ since D_1 is level-redundant w.r.t. S. Otherwise by the induction hypothesis, we must have $S|_i \cup S|_\perp \models D_1$. The same property holds for D_2. Consequently, the parents of C' are logical consequences of $S|_i \cup S|_\perp$ hence $S|_i \cup S|_\perp \models C'$.

The second part of the lemma follows immediately by a straightforward induction (since $S|_{[i,\infty[} \stackrel{\text{def}}{=} \bigcup_{j \geq i} S|_j$). ■

Definition 25 If $i > 0$, we denote by $S|_i^\star$ the set of non ground clauses of level i in S. If $i = 0$ then we define $S|_0^\star \stackrel{\text{def}}{=} S|_0$. ◊

Lemma 26 *Let S be a k-reducible, set of normalized clauses. If $i > k$, then $S|_i^\star \cup S|_\perp \models S|_i$.*

PROOF. Let $C \in S|_i$. Since S is k-reducible, there exists a derivation of C from a set of k-normalized clauses. Furthermore, every clause in this derivation is level-redundant w.r.t. S. We prove, by induction on the length of the derivation, than for every clause C' of level i occurring in the derivation, we have $S|_i^\star \cup S|_\perp \models C'$. The proof is obvious if C' is non ground. If C is ground, since $i > k$, C' must be deduced from two clauses D_1 and D_2 by resolution (or factorization). Furthermore, by Lemma 18, D_1 and D_2 are of level i or $\perp$. By the induction hypothesis we have $S|_i^\star \cup S|_\perp \models D_1, D_2 \models C'$. ∎

Proposition 27 *Let S be a set of clause. Let C be a clause such that $C \sqsubseteq_l S$. Let i be the level of C. If C is not ground then $C \sqsubseteq_l S|_\perp \cup S|_i^\star$.*

PROOF. By definition there exist n clauses $C_1, \ldots, C_n$ of level $\perp$ or i such that for every substitution σ there exist n substitutions $\sigma_1, \ldots, \sigma_n$ such that $C\sigma_1, \ldots, C\sigma_n \models C$ and $C_1\theta_1, \ldots, C_n\theta_n \preceq C\sigma$. Furthermore, since C is not ground, $C_1, \ldots, C_n$ are not ground. If C_i is of level $\perp$ then it occurs in $S|_\perp$ and if it is of level i, then, since it is not ground, it occurs in $S|_i^\star$. Thus $C_1, \ldots, C_n \in S|_\perp \cup S|_i^\star$ and $C \sqsubseteq_l S|_\perp \cup S|_i^\star$. ∎

Lemma 28 *Let S be a k-reducible set of normalized clauses and let i, j be natural numbers.*

If:

1. *$j \neq 0$ and $i > k$.*
2. *S is saturated up to level i.*
3. *$S|_\perp \cup S|_{i+j}^\star \models \mathit{shift}(S|_i^\star, j)$.*

Then: S is satisfiable iff $S \cup \{n < i + j\}$ is. $n < i + j$ is called a pruning clause *for S.*

PROOF. Let $S' = S|_{[\perp,i[} \cup \bigcup_{l \geq i} S|_l^\star$. By definition we have $S|_l = S'|_l$ if $l < i$, and $S'|_l = S|_l^\star$ if $l \geq i$. Furthermore, since S is k-reducible, S' is also k-reducible. Finally, by Lemma 26 we have $S \equiv S'$. Hence it suffices to show that $S' \cup \{n < i + j\}$ is satisfiable. We prove that S' satisfies the application conditions of Lemma 22.

1. **S' is saturated up to level i.** Let C be a clause deduced from clauses in $S'|_{[\perp,i[} = S|_{[\perp,i[}$. Since S is saturated up to level i, C is level-redundant with respect to S. Hence $C \sqsubseteq_l S$. If the level of C is strictly lower than i we deduce that $C \sqsubseteq_l S|_{[\perp,i[}$ hence $C \sqsubseteq_l S'$. Otherwise, by Lemma 18, C is non ground, thus $C \sqsubseteq_l S|_\perp \cup S|_i^\star$ by Proposition 27.
2. $S'|_i \cup S'|_\perp \models S'|_{[i,\infty[}$. This is a direct application of Lemma 24.

3. $S'|_i$ **contains no ground clause**. If $i = 0$, then this is trivial since by definition there exists no ground clause of depth 0. If $i > 0$ then the result stems from the definition of $S|^\star_i$.

4. $S'|_\perp \cup S'|_{i+j} \models \mathit{shift}(S'|_i, j)$. We have $S|_\perp \cup S|^\star_{i+j} \models \mathit{shift}(S|^\star_i, j)$ and $S|_\perp \cup S|^\star_{i+j} = S'|_\perp \cup S'|_{i+j}$. Thus $S'|_\perp \cup S'|_{i+j} \models \mathit{shift}(S|^\star_i, j)$. Furthermore $S|^\star_i \setminus S'|_i$ hence $\mathit{shift}(S|^\star_i, j) \setminus \mathit{shift}(S'|_i, j)$. ∎

In practice, Condition 3 may be difficult to check. A simple solution (that is sufficient for the termination result in the following section) is to check that $S|^\star_{i+j} = \mathit{shift}(S|^\star_i, j)$ (up to a renaming of variables).

Example 29 Consider the clause set $S = \{1, \ldots, 8\}$ of Example 7 (generated from the formula $p_n \wedge (\forall x\, p_{succ(x)} \Rightarrow p_x) \wedge \neg p_0$):

1	$n \not\approx x \vee p_x$	(level 1)
2	$p_y \vee \neg p_{succ(y)}$	(level $\perp$)
3	$\neg p_0$	(level $\perp$)
4	$n \not\approx succ(y) \vee p_y$	(level 2)
5	$n \not\approx 0$	(level 1)
6	$n \not\approx succ(0)$	(level 2)
7	$n \not\approx succ(succ(y)) \vee p_y$	(level 3)
8	$n \not\approx succ(succ(0))$	(level 3)

The initial clauses $1, 2, 3$ are of level $\perp$, 0 or 1 thus S is 1-reducible. Let $i = 2$, $j = 1$. We have $S|^\star_i = \{n \not\approx succ(y) \vee p_y\}$, $S|^\star_{i+j} = \{n \not\approx succ(succ(y))\}$ (the clauses 6 and 8 are dismissed since they are ground, according to Definition 25). Thus $\mathit{shift}(S|^\star_i, j) = \{n \not\approx succ(succ(y)) \vee p_y\} = S|^\star_{i+j}$, hence Condition 3 holds trivially. Furthermore, it is easy to check that S is saturated up to level i.

Hence the pruning rule applies and generates: $n \not\approx succ(succ(succ(x)))$, i.e. $n < 3$. Together with clauses 5, 6 and 8, we obtain a finite, purely equational, and unsatisfiable clause set. The unsatisfiability of this set can be tested by standard algorithms. Thus the initial clause set is unsatisfiable. ♣

Example 30 Consider the following schema:

$$p_0 \vee \bigwedge_{x=0}^{n} (p_x \Rightarrow q_x) \wedge \bigwedge_{x=0}^{n} (q_x \Leftrightarrow \neg q_{succ(x)}) \wedge \neg q_n \wedge \neg q_{succ(n)}.$$

This schema can be encoded by the following clause set (see Section 8 for details):

1	p_0	(level $\perp$)
2	$\neg p_x \vee q_x$	(level $\perp$)
3	$\neg q_x \vee \neg q_{succ(x)}$	(level $\perp$)
4	$q_x \vee q_{succ(x)}$	(level $\perp$)
5	$n \not\approx x \vee \neg q_x$	(level 1)
6	$n \not\approx x \vee \neg q_{succ(x)}$	(level 0)

We apply our calculus (we assume that $p \prec q$):

7	q_0	(resolution 1,2)	(level $\bot$)
8	$n \not\approx 0$	(resolution 7, 5)	(level 1)
9	$n \not\approx succ(x) \vee q_x$	(resolution 5,4)	(level 2)
10	$\neg p_{succ(x)} \vee \neg q_x$	(resolution 2, 3)	(level $\bot$)
11	$n \not\approx succ(succ(x)) \vee \neg q_x$	(resolution 9, 3)	(level 3)
12	$n \not\approx succ(succ(0))$	(resolution 11, 7)	(level 3)
13	$n \not\approx succ(succ(succ(x))) \vee q_x$	(resolution 11, 4)	(level 4)
14	$n \not\approx x \vee q_x$	(resolution 4, 6)	(level 1)
15	$n \not\approx succ(x) \vee \neg q_x$	(resolution 3, 14)	(level 2)
16	$n \not\approx succ(0)$	(resolution 7, 15)	(level 2)
17	$n \not\approx succ(succ(x)) \vee q_x$	(resolution 4, 15)	(level 3)
18	$n \not\approx succ(succ(succ(x))) \vee \neg q_x$	(resolution 17, 3)	(level 4)
19	$n \not\approx succ(succ(succ(0)))$	(resolution 18, 7)	(level 4)

Let $i = 2$, $j = 2$. Let $S' = \{1\text{-}18\}$. S' is saturated up to level 2. We have $S'|_2^\star = \{9, 15\}$, $S'|_4^\star = \{13, 18\}$. One can easily check that $shift(S|_2^\star, 2) = S|_4^\star$. Thus the looping rule applies and generates: $n \not\approx succ(succ(succ(succ(x))))$ i.e. $n < 4$. Together with the clauses 8, 10, 12 and 19, this yields a finite, purely equational, unsatisfiable, clause set. ♣

6 Termination

In this section we define a *pruning rule* based on Lemma 28 and we show that the addition of this rule makes the calculus terminating, provided that the rules are applied in a fair way.

Definition 31 A *derivation with pruning from* a clause set S is a (possibly infinite) sequence of clause sets $(S_i)_{i \in I}$, with $I = [0, n]$ or $I = \mathbb{N}$, such that $S_0 = S$ and for every $i \in I \setminus \{0\}$, one the following conditions holds:

- $S_i = S_{i-1} \cup \{C\}$ where $C \in \mathcal{R}(S_{i-1})$ (deduction step).
- $S_i = S_{i-1} \cup \{C\}$ where C is a pruning clause for S_{i-1} (pruning step).
- $S_i = S_{i-1} \setminus \{C\}$, where C is level-redundant in S_i (deletion step). ◊

In practice identifying all level-redundant clauses is unfeasible, thus we only delete the clauses that are valid or subsumed.

We write $S \vdash_p C$ if there exists a derivation with pruning $C_1, \ldots, C_n$ from S such that $C_n = C$.

Definition 32 A *derivation with pruning of* a clause set S is a derivation with pruning containing every clause in S (here S denotes the *conclusions* of the derivation). ◊

A derivation is *non redundant* iff the deletion steps are applied with the highest priority. It is *fair* iff for every C such that $S \vdash_p C$, there exists an $i \in \mathbb{N}$ such that C is redundant w.r.t. S_i.

Lemma 33 *Let $(S_i)_{i\in I}$ be a derivation with pruning from a k-normalized clause set. For every $i \in I$, S_i is k-reducible.*

Proof. By definition every clause C in S_i can be deduced from clauses in S. Thus there exists a derivation $C_1, \ldots, C_n$ from S such that $C_n = C$. By definition, for every clause C_j $(1 \leq j \leq n)$, either $C_j \in S_i$, or C_j has been deleted by a previous redundancy elimination step. In both cases C_j must be level-redundant with respect to S_i. ∎

Corollary 34 *Let $(S_i)_{i\in I}$ be a fair, non redundant derivation with pruning from a finite and normalized clause set. I is finite.*

Proof. Since S_0 is finite and normalized, it must be k-normalized for some $k \in \bot \cup \mathbb{N}$. Assume that I is infinite. By Proposition 12, the number of distinct clauses of a given level is finite. Thus for every l, there exists an index $\gamma(l)$ such that every clause that is generated after this step is of level strictly greater than l. Since I is infinite, γ is strictly increasing. Let l be a natural number strictly greater that the maximal number of distinct normalized clause sets of the same level.

For every $m \in \mathbb{N}$, let S'_m denote the set of clauses C such that $n \not\approx \mathit{succ}^{m-1+\epsilon}(x) \vee C \in S_{\gamma(l)}|_m$, with $x \in \mathcal{V}$. Obviously, S'_m is a set of normalized clauses. Then by the pigeonhole argument, there exist $i < l$ and $j \neq 0$ such that $S'_i = S'_{i+j}$. By definition, this implies that $S_{\gamma(l)}|^\star_{i+j} = \mathit{shift}(S_{\gamma(l)}|^\star_i, j)$ (thus in particular $S_{\gamma(l)}|^\star_{i+j} \models \mathit{shift}(S_{\gamma(l)}|^\star_i, j)$).

Furthermore, $S_{\gamma(l)}$ is saturated up to level i (since the derivation is fair and since no clause of level i or $i+1$ can be deduced after step $\gamma(l)$). Thus the pruning rule applies on $S_{\gamma(l)}$. Hence it also applies on $S_{\gamma(l)+1}, \ldots$. By fairness it must be applied at some point, thus the clause $n < i + j$ occurs in one of the S_k. But then the number of non redundant clauses is finite since any clause of level $\geq i + j$ is subsumed by $n < i + j$. ∎

7 Constructing the resolution proof

From the result of the previous sections, we know that a normalized clause set S is satisfiable iff a finite set of unsatisfiable purely equational clauses is derivable from S. In this section we show how to explicitly construct a resolution proof for S (which is needed for applying the cut elimination procedure in [2]). Since S is parameterized by a parameter n, so is its proof, which is actually a schema of (propositional) resolution proofs.

We introduce some notations. Let δ be a clause set or a derivation. If $k \in \mathbb{N}$, we denote by $\delta[k]$ the clause set or derivation obtained by:

- Deleting from δ every clause of the form $n \not\approx t \vee C$ such that k is not an instance of t.
- Replacing every clause $n \not\approx t \vee C$ such that $k = t\sigma$ by $C\sigma$.

Intuitively the clauses in $S[k]$ are exactly those obtained by fixing the value of the parameter n to k. Notice that the clauses in $\delta[k]$ are standard clauses (with no parameter). We have the following:

Proposition 35 *Let S be a set of clauses. Let I be an interpretation. $I \models S$ iff $I \models S[I(n)]$.*

PROOF. This follows immediately from the definition. ∎

Thus a set of clauses S is satisfiable iff there exists a natural number k such that $S[k]$ is satisfiable. The set $\{S[k] \mid k \in \mathbb{N}\}$ is the family of (standard) clause sets denoted by the schema S. Our goal is to construct, from a refutation of S, a family of refutation (without pruning) of every clause set $S[k]$.

Proposition 36 *For every set of non parameter-free and non ground clauses S and for every pair of natural number k, j such that $k \geq j$, we have $\mathit{shift}(S, j)[k] = S[k - j]$.*

PROOF. By definition $\mathit{shift}(S, j)$ is the set of clauses of the form $n \not\approx \mathit{succ}^{i+j}(x) \vee C$, where $x \in \mathcal{V}$ and $n \not\approx \mathit{succ}^{i}(x) \vee C \in S$. Notice that every clause in S must have the form $n \not\approx \mathit{succ}^{i}(x) \vee C$ where $i \in \mathbb{N}$ and $x \in \mathcal{V}$.

Thus $\mathit{shift}(S, j)[k]$ is the set of clauses of the form $C\sigma$, where $\mathit{succ}^{i+j}(x)\sigma = k$, i.e. the set of clauses of the form $C\{x \to k - j - i\}$ where $n \not\approx \mathit{succ}^{i}(x) \vee C \in S$ and $i \leq k - j$. This last set of clauses is exactly $S[k - j]$. ∎

Proposition 37 *Let S be a set of normalized clauses. If δ is a derivation from S, then $\delta[k]$ is a derivation (in the usual sense) from $S[k]$.*

PROOF. The proof is by a straightforward induction on the length of δ. ∎

The following lemma handles the (easy) case of the derivations without pruning.

Lemma 38 *Let S be a normalized clause set and let δ be a refutation of S. Let $(\Delta_k)_{k\in\mathbb{N}}$ be the family of derivations defined by $\Delta_k \stackrel{def}{=} \delta[k]$.*

If δ does not contain any application of the pruning rule then Δ_k is a refutation of $S[k]$.

PROOF. This is an immediate consequence of Proposition 37. ∎

Let δ be a derivation with pruning from a clause set S. We may assume that δ contains only one application of the pruning rule (this is sufficient to ensure termination, as shown in Section 6). We denote by i^δ and j^δ the natural number i, j corresponding to the application of the pruning rule in δ (as defined in Lemma 28). We denote by S^δ the set of clauses on which the pruning rule is applied and by T^δ the set: $T^\delta \stackrel{\text{def}}{=} S^\delta|_i^\star \cup S|_\perp$.

Lemma 39 *Let S be a set of normalized clauses and let δ be a refutation with pruning of S.*

Let δ' be the greatest subderivation of δ from T^δ.

Let $(\Delta_k)_{k\in\mathbb{N}}$ be the family of derivations defined as follows:

- $\Delta_k \stackrel{def}{=} \delta[k]$ *if* $k < i$.
- $\Delta_k \stackrel{def}{=} \delta[k].\Gamma_k$ *if* $k \geq i$ *where:*
  - $\Gamma_k \stackrel{def}{=} \delta'[k]$ *if* $k < i + j$
  - $\Gamma_k \stackrel{def}{=} \delta'[k].\Gamma_{k-j}$ *if* $k \geq i + j$

Δ_k is a refutation of $S[k]$.

Proof. If $k < i$, then the proof is an immediate consequence of Proposition 37. If $k \geq i$, we show (by induction on k) that Γ_k is a refutation of $T^\delta[k]$. This is sufficient to obtain the desired result since, by Lemma 24, each clause in T^δ occurs in δ (thus $\delta[k]$ is a derivation of $T^\delta[k]$).

Again, we distinguish two cases, according to the value of k.

- Assume firstly that $i \leq k < i + j$. By definition, $\delta'[k]$ is a derivation from $T^\delta[k]$. δ must contain a clause of the form $n \not\approx k$. This clause is of level greater than i, thus must be deducible form T^δ, as shown in the proof of Lemma 24. Hence δ' is a derivation of $n \not\approx k$ and $\delta'[k]$ is a refutation of $T^\delta[k]$.
- Now assume that $k \geq i + j$. Every clause in $S^\delta|^\star_{i+j}$ is deducible from T^δ (as shown by Lemma 24). Thus δ' is a derivation from T^δ of $S^\delta|^\star_{i+j}$. By definition of the pruning rule, we have $S^\delta|^\star_{i+j} = \mathit{shift}(S^\delta|^\star_i, j)$. Hence δ' is a derivation from T^δ of $\mathit{shift}(S^\delta|^\star_i, j)$, thus $\delta'[k]$ is a derivation of $\mathit{shift}(S^\delta|^\star_i, j)[k]$.

  We have $T^\delta = S^\delta|^\star_i \cup S^\delta|_\perp$. By Proposition 36 $S^\delta|^\star_i[k - j]$ is equal to $\mathit{shift}(S^\delta|^\star_i, j)[k]$. Hence $\delta'[k]$ is a derivation of $S^\delta|^\star_i[k - j]$. Thus $\delta'[k]$ is a derivation of $S^\delta|^\star_i[k - j] \cup S^\delta|_\perp$ hence of $T^\delta[k - j]$. But by the induction hypothesis Γ_{k-j} is a refutation of $T^\delta[k-j]$. Thus $\delta'[k].\Gamma_{k-j}$ is a refutation of $T^\delta[k]$. ∎

8 Expressive power

Although the class of normalized clause sets is strongly restricted from a syntactic point of view it is actually rather expressive. We show how to encode literals that are outside the class into normalized clause sets.

8.1 Encoding translation on indices

We first consider atoms of the form $p_{succ^k(x)}$ (where $k \in \mathbb{N}$). Such formulae are encoded by a new predicate symbol p^k_x, with the intended meaning that $p^k_x \Leftrightarrow p_{succ^k(x)}$. The following axioms ensure that p^k_x have the intended interpretation:

$$\mathcal{A}_k \stackrel{\text{def}}{=} \{p^l_x \Leftrightarrow p^{l-1}_{succ(x)}, p^0_x \Leftrightarrow p_x \mid k \geq l > 0\}.$$

Notice that $\mathcal{A}_k$ is finite and normalized. We have the following:

Proposition 40 *If $I \models \mathcal{A}_k$ then for every $l \in [0,k]$, $I \models (p^l_x \Leftrightarrow p_{succ^l(x)})$.*

Proof. By a straightforward induction on l. ∎

8.2 Encoding inequalities and equalities

The formula $x + l > \epsilon.n + k$ (with $l, k \in \mathbb{N}$, $\epsilon \in \{0,1\}$) is encoded by a predicate $q^{l,\epsilon,k}_x$, whose semantics are defined by the following axioms:

$\mathcal{A}'_{l',k'}$ is the set of clauses of the form:

$$\begin{gathered}
q^{l,\epsilon,k}_x \Leftrightarrow q^{0,\epsilon,k}_{succ^l(x)} \\
q^{0,\epsilon,k+1}_{succ(x)} \Leftrightarrow q^{0,\epsilon,k}_x \\
\neg q^{0,\epsilon,k}_0 \\
q^{0,0,0}_{succ(x)} \\
n \not\approx x \vee q^{0,1,0}_{succ(x)} \\
n \not\approx x \vee \neg q^{0,1,0}_x \\
q^{0,1,0}_{succ(x)} \vee \neg q^{0,1,0}_x
\end{gathered}$$

where $l \in [0,l']$, $k \in [0,k']$ and $\epsilon \in \{0,1\}$.

Again, $\mathcal{A}'_{l',k'}$ is finite. It is not normalized due to the index $succ^k(x)$ in the first axiom, but obviously one can apply the previous transformation to obtain an equivalent normalized formulation.

Proposition 41 *Let $I \models \mathcal{A}'_{l',k'}$. For every $(l,k,\epsilon) \in [0,l'] \times [0,k'] \times \{0,1\}$, we have $I \models q^{l,\epsilon,k}_x \Leftrightarrow x + l > \epsilon.n + k$.*

Proof. Assume that $l = k = 0$. If $\epsilon = 0$ then the interpretation of $q^{l,\epsilon,k}$ is fixed by the third and fourth axioms and we have indeed $q^{0,0,0}_{succ(x)}$ (i.e. $succ(x) > 0$) and $\neg q^{0,0,0}_0$ (i.e. $0 \not> 0$). If $\epsilon = 1$, then the interpretation of $q^{l,\epsilon,k}$ is fixed by the fifth and sixth axioms that state that we have $q^{0,1,0}_{succ(x)}$ if $x = n$ (i.e. $succ(n) > n$) and $\neg q^{0,1,0}_x$ if $x = n$ (i.e. $n \not> n$). Furthermore, the last axiom ensures that $q^{0,1,0}_x$ is true (resp. false) if $x > s(n)$ (resp. $x < n$).

By using the second and third axiom we can show by a straightforward induction on x that $q^{0,\epsilon,k}_x$ is equivalent to $q^{0,\epsilon,k-x}_0$ if $x < k$ and to `false` otherwise.

Then the first axiom states that $q^{l,\epsilon,k}_x$ is equivalent to $q^{0,\epsilon,k}_{x+l}$. ∎

The formulae $x + l \leq \epsilon.n + k$ can be easily encoded as $x + l + 1 \not> \epsilon.n + k$. Then $x + l \approx \epsilon.n + l$ can be written as $x + l + 1 \geq \epsilon.n + l \wedge x + l \leq \epsilon.n + l + 1$.

8.3 Encoding schemata with several parameters

The restriction to one parameter only is also non restrictive. Indeed, let S be a clause set with k parameters $n_1, \ldots, n_k$. We introduce a new parameter n (which encodes the max of the n_i's). The formula $x > n_l$ are defined by atoms r^l_x, defined by the following formulae $\mathcal{A}''$:

$$\neg r^l_x \vee r^l_{succ(x)}$$
$$\neg r^l_0$$
$$n \not\approx x \vee r^l_{succ(x)}$$

where $l \in [1, k]$.

Notice that the obtained formula is normalized.

Proposition 42 *If $I \models \mathcal{A}''$ then for every $l \in [1, k]$ there is exactly one natural number n_l such that $I \models r^l_x$ iff $x > n_l$. Furthermore, $n_l \leq n$.*

Proof. Let n_l be the lowest natural number such that $I \models r^l_{succ(n_l)}$. n_l exists since $r^l_{succ(n_l)}$ holds (by the third axiom). We have $I \models \neg r^0_l$, thus by minimality of n_l $r^l_{n_l}$ cannot hold. Then by using the first axiom we show that $I \models r^l_x$ for every $x > n_l$ and that $I \models \neg r^l_x$ for every $x \leq n_l$. ∎

Then the formula $x \approx n_l$ is written $x + 1 > n_l \wedge x \not> n_l$.

8.4 Inductive definitions

Now, consider an inductive definition: $\phi_0 \Leftrightarrow B$ and $\phi_{k+1} \Leftrightarrow I$ if $k \geq 0$ where B and I denotes formulae and where I contains atoms of the form p_k or $p_{succ(k)}$ or inductively defined formulae ψ_k (with possibly $\phi = \psi$). This definition can be easily expressed in our formalism by considering every such formula ϕ as a predicate symbol:

$$\phi_0 \Leftrightarrow B$$

$$\phi_{succ(x)} \Leftrightarrow I.$$

One gets a 1-normalized clause set by translation into clausal form. More complex inductive definition may be encoded in the same way. In particular, formulae of the form $\bigvee_{i=a}^{b} \phi_i$ are easily encoded by an atom $\psi_{b-a+1} = \bigvee_{x=1}^{b-a+1} \phi_{x-1+a}$ defined as follows:

$$\psi_0 \Leftrightarrow \texttt{false}$$

$$\psi_{x+1} \Leftrightarrow \phi_{x-1+a} \vee \psi_x.$$

After translation into clausal normal form and after removing the translation as explained in Section 8.1, one gets a normalized clause set.

8.5 Regular schemata

Using the previous transformations it is easy to prove that every regular schema can be encoded into a normalized clause set, i.e. that for every regular schema ϕ one can construct a normalized clause set S such that ϕ and S are equisatisfiable (every model of S is also a model of ϕ and every model of ϕ can be extended into a model of S). Furthermore, the size of S is linear w.r.t. the one of ϕ (if natural numbers are encoded as unary terms in the original schema[2] and if a structural-preserving transformation is used to compute clausal forms).

Furthermore, normalized clause sets allow one to express properties that cannot be stated as regular schemata, for instance the formula $\bigwedge_{i=0}^{\infty} p_i$ is easily expressed by the normalized clause set $\{p_i\}$ but it is not a regular schema (due to the unbounded conjunction).

References

[1] V. Aravantinos, R. Caferra, and N. Peltier. A schemata calculus for propositional logic. In *TABLEAUX 09 (International Conference on Automated Reasoning with Analytic Tableaux and Related Methods)*, volume 5607 of *LNCS*, pages 32–46. Springer, 2009.

[2] M. Baaz and A. Leitsch. Cut-elimination and Redundancy-elimination by Resolution. *Journal of Symbolic Computation*, 29(2):149–176, 2000.

[3] H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–475, 1989.

[4] D. Cooper. Theorem proving in arithmetic without multiplication. In B. Meltzer and D. Michie, editors, *Machine Intelligence 7*, chapter 5, pages 91–99. Edinburgh University Press, 1972.

[5] M. Horbach and C. Weidenbach. Superposition for fixed domains. *ACM Trans. Comput. Logic*, 11(4):1–35, 2010.

[6] M. Presburger. Über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchen die addition als einzige operation hervortritt. In *Comptes Rendus du I congrés de Mathématiciens des Pays Slaves*, pages 92–101, 1929.

[2] Otherwise a schema as simple as $p_1 \underbrace{0 \ldots 0}_{n}$ could not encoded in linear size).