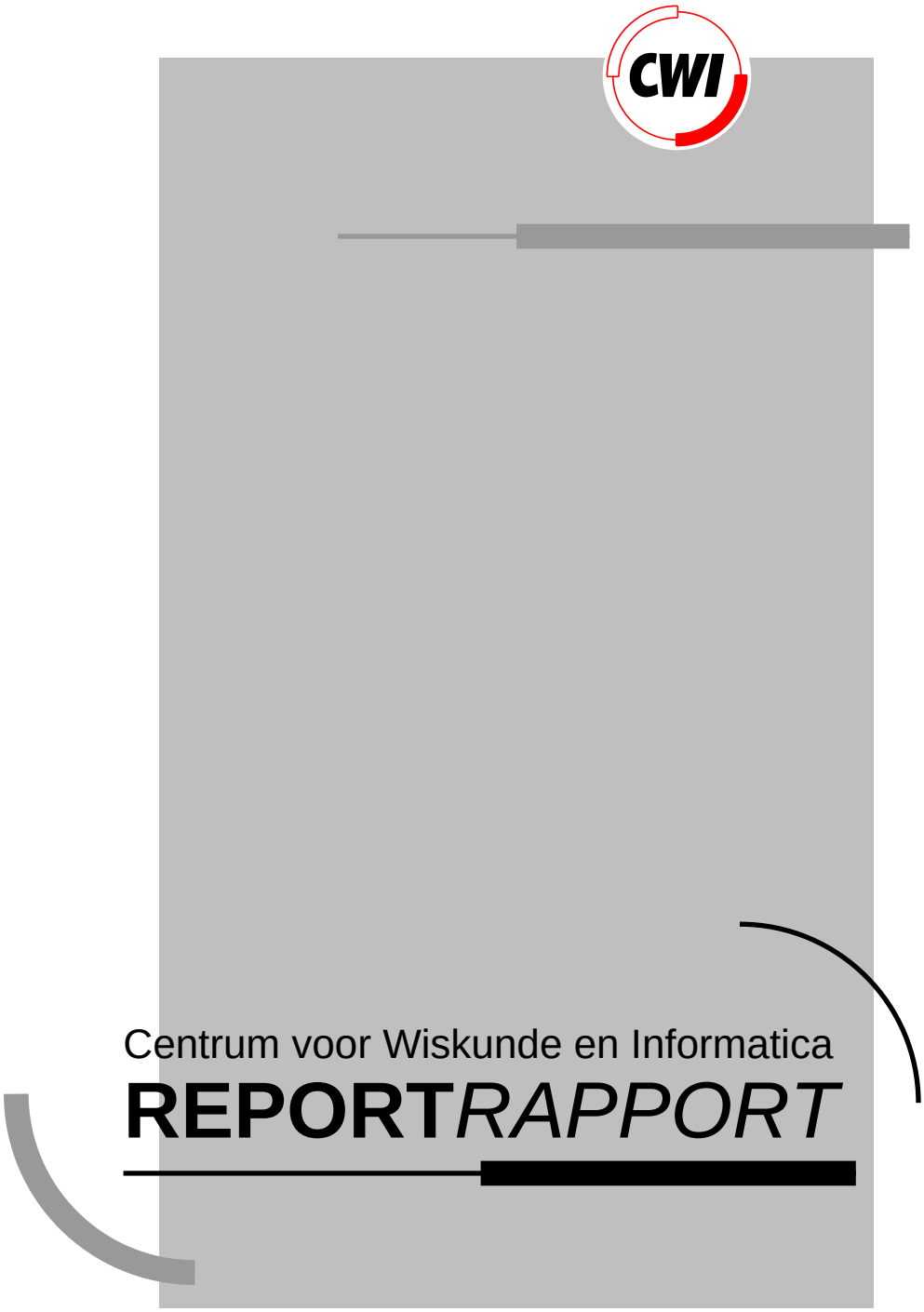

Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

Simultaneous Replacement in Normal Programs

A. Bossi, N. Cocco, S. Etalle

Computer Science/Department of Software Technology

CS-R9357 1993

Simultaneous Replacement in Normal Programs

Annalisa Bossi
Dipartimento di Matematica Pura ed Applicata,
Università di Padova,
Via Belzoni 7, 35131 Padova, Italy.

Nicoletta Cocco
IUAV,
Tolentini 191, S. Croce, Venezia.

Sandro Etalle
CWI,
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands.

Abstract

The simultaneous replacement transformation operation, is here defined and studied wrt normal programs. We give applicability conditions able to ensure the correctness of the operation wrt the set of logical consequences of the completed database. We consider separately the cases in which the underlying language is infinite and finite; in this latter case we also distinguish according to the kind of domain closure axioms adopted. As corollaries we obtain results for Fitting's and Kunen's semantics. We also show how simultaneous replacement can mimic other transformation operations such as thinning, fattening and folding, thus producing applicability conditions for them too.

1991 Mathematics Subject Classification: 68Q40, 68T15.
CR Categories: F.3.2., F.4.1, H.3.3, I.2.3.
Keywords and Phrases: Program's Transformation, Logic Programming, Semantics, Negation, Replacement.

Notes. Permanent address of the third author: Dipartimento di Matematica Pura ed Applicata, Università di Padova, Via Belzoni 7, 35131 Padova, Italy.

1 Introduction

1.1 The Language Problem

In this paper we consider as semantics for a normal logic program P the set of logical consequences of its completion $Comp(P)$; the consistency problem is avoided by using three valued logic instead of the classical two valued. Yet these choices still lead to possibly different approaches; the reason is that $Comp(P)$ depends strongly on the underlying language $\mathcal{L}$, and when $\mathcal{L}$ is finite (that is, when it contains only a finite number of functions symbols) the equality theory which is incorporated in $Comp(P)$ is not complete. This problem can be solved by adding to $Comp(P)$ some domain closure axioms (DCA), which are intended to restrict the quantification to the universe of $\mathcal{L}$-terms. Again in the literature we find two different kind of such axioms: the strong (DCA) and the weak (WDCA) ones. It follows that we have three different approaches are possible, namely we may:

a) Consider an infinite language, with no domain closure axioms. This is the approach followed by Kunen [11].

b) Consider a finite language and adopt the strong domain closure axioms (DCA). This was studied by Fitting in the case that $\mathcal{L}$ coincides with the language of the program $\mathcal{L}(P)$; this semantics is commonly

known as Fitting's Model semantics. His results can also be applied in the case in which $\mathcal{L}$ is larger than $\mathcal{L}(P)$.

c) Consider a finite language and adopt the weak domain closure axioms (WDCA). This has been studied by Shepherdson [16], and the results are similar to the ones found for the case of an infinite language (case (a) above).

What these three approaches have in common is that the semantics can always be characterized via the Kleene sequence of the operator Φ_P, the three-valued counterpart of the usual immediate consequence operator T_P. In this paper we consider the three cases separately; we also characterize the equivalence of two programs P and P' by referring solely to the Kleene sequence of the operators Φ_P and $\Phi_{P'}$.

1.2 The Replacement Operation

The *replacement* operation has been introduced for transforming definite programs by Tamaki and Sato in [17] and after that it has been rather neglected by people working on program transformations apart from Sato himself [15], Maher [13] and Gardner and Shepherdson [9]. Replacement consists in substituting a conjunction of literals, in the body of a clause, with another conjunction. It is a very general transformation able to mimic many other operations, such as thinning, fattening [3] and folding, which can be seen as particular instances of replacement.

A basic requirement for the applicability of replacement is that the replaced and replacing parts are equivalent with respect to the considered semantics. But this is not sufficient, it is also necessary that this equivalence still holds in the transformed program; in fact replacement could introduce infinite loops through the modified clause. In case of normal programs, the problem is further complicated by the presence of negation.

A few proposals have been given. For definite programs, Tamaki and Sato in [17] give an applicability condition which compares the smallest proof trees of the two considered conjunctions. Gardner and Shepherdson, in [9], give conditions for preserving both the procedural (SLDNF) semantics and the declarative one. Such conditions are based on Clark's (two valued) completion of the program [6]. Sato, in [15], considers replacement of tautologically equivalent formulas in first order programs. Maher [13, 14] restricts the applicability of replacement to the case in which the replaced literals are independent from the clause where the replacement is applied.

Here we study *simultaneous replacement* which consists in performing many replacements all at the same time, and define *applicability conditions* able to guarantee the correct application of the operation in normal programs with respect to the three semantics mentioned above. This will allow us to draw conclusions for Fitting's and for Kunen's semantics as well.

In some previous papers also the Well-Founded Model semantics for normal programs (in [7]) and the S-semantics for definite programs (in [4]) have been considered and similar results have been obtained.

Our approach is based on two concepts: The *semantic delay between two conjunctions of literals* and the *dependency degree of a conjunction of literals wrt a clause*. The first corresponds to compare a measure of complexity of predicates, namely the number of applications of the fixpoint operator which are necessary for determining their truth or falsity. The second corresponds to the length of the shortest path reaching a clause in the derivation tree of a conjunction of literals. Our applicability conditions for replacement compare the semantic delay between the two conjunctions of literals and the dependency degree of the replaced part with the clause to be transformed. In this way it is possible to characterize when "there is no space to introduce a loop". Such applicability conditions, are undecidable in general, but other decidable, syntactic conditions can be derived for special cases. In [5] we consider two such cases when replacement simulates folding.

1.3 Structure of the Paper

In Section 2 the main definitions related to the semantics we use are briefly recalled, and the definition of simultaneous replacement is given. We also define equivalence among programs and correctness of a transformation operation wrt a general first order theory. In Section 3 we characterize these definitions via the three valued operator Φ_P for the case in which the theory corresponds to the program's completion together with the DCA closure axioms; finally we state and prove the results on the correctness of the

replacement operation wrt to the semantics just mentioned. Section 4 contains the same results of Section 3 for the case of the WDCA closure axioms. In Section 5 we also take into account the case of an infinite language, with no closure axioms. In Section 6 some examples are provided and it is shown also how thinning and fattening can be seen as special cases of replacement, thus yielding, as a consequence, conditions for a safe application of these operations to normal programs. Reversible folding is also considered and its safeness is proved by assimilating it to the replacement operation. A short conclusion follows. Part of the proofs is given in the Appendices.

2 Preliminaries

We assume that the reader is familiar with the basic concepts of logic programming; throughout the paper we use the standard terminology of [12] and [1]. We consider *normal programs*, that is finite collections of *normal rules*, $A \leftarrow L_1, \ldots, L_m.$ where A is an atom and $L_1, \ldots, L_m$ are literals. Symbols with a $\sim$ on top denote tuples of objects, for instance $\tilde{x}$ denotes a tuple of variables $x_1, \ldots, x_n$, and $\tilde{x} = \tilde{y}$ stands for $x_1 = y_1 \wedge \ldots \wedge x_n = y_n$. We also adopt the usual logic programming notation that uses "," instead of $\wedge$, hence a conjunction of literals $L_1 \wedge \ldots \wedge L_n$ will be denoted by $L_1, \ldots, L_n$ or by $\widetilde{L}$.

In this paper we work with Kleene's three valued logic [10] where the truth values are *true*, *false* and *undefined*. The usual logical connectives have value *true* (or *false*) when they have that value in ordinary two valued logic for all possible replacements of *undefined* by *true* or *false*, otherwise they have the value *undefined*.

Three valued logic allows us to define connectives that do not exist in two valued logic. For example, in the sequel we use the $\Leftrightarrow$, Lukasiewicz's operator of "having the same truth value"; $a \Leftrightarrow b$ is *true* if a and b are both *true*, both *false* or both *undefined*; in any other case $a \Leftrightarrow b$ is *false*. By contrast, the usual $\leftrightarrow$ is *undefined* when one or both its arguments are *undefined*.

In some cases we restrict our attention to formulas which we consider "well-behaving" in the three valued semantics. Next definition is intended for characterizing such formulas.

Definition 2.1

- A logic connective $\diamond$ is *allowed* if the following property holds: when $a \diamond b$ is *true* or *false* then its truth value does not change if one of its argument is changed from *undefined* to *true* or *false*.
- A first order formula is *allowed* if it contains only allowed connectives. □

Note that any formula containing the connective $\Leftrightarrow$ is not allowed, while formulas built with the usual logic connectives are allowed.

Allowed formulas can be seen as monotonic functions over the lattice on the set $\{undefined, true, false\}$ which has *undefined* as bottom element and *true* and *false* are not comparable.

2.1 Completion for Normal Programs

A *language* $\mathcal{L}$ is determined by a set of function and predicate symbols of fixed arities. Constants are treated as 0-ary function symbols. We say that a language is *infinite* if it contains infinitely many function symbols (including those of arity 0), otherwise we say that it is *finite*. If P is a program then $\mathcal{L}(P)$ denotes the finite language of the functions and predicate symbols actually occurring in the program.

The usual Clark's completion definition, $Comp(P)$, [6] is extended to three valued logic by replacing $\leftrightarrow$, in the completed definitions of the predicates, with $\Leftrightarrow$. This saves $Comp(P)$ from the inconsistencies that it can have in two valued logic. For example the program $P = \{p \leftarrow \neg p.\}$ has $Comp(P) = \{p \Leftrightarrow \neg p\}$ which has a model with p *undefined*.

Definition 2.2 Let P be a program and $p(\tilde{t}_1) \leftarrow \widetilde{B}_1, \ldots, p(\tilde{t}_r) \leftarrow \widetilde{B}_r$ be all the clauses which define predicate symbol p in P. The *completed definition* of p is

$$p(\tilde{x}) \Leftrightarrow \bigvee_{i=1}^{r} \exists \tilde{y}_i \ (\tilde{x} = \tilde{t}_i) \wedge \widetilde{B}_i.$$

where $\tilde{x}$ are new variables and $\tilde{y}_i$ are the variables in $p(\tilde{t}_i) \leftarrow \widetilde{B}_i$.

If P contains no clause defining p, then the completed definition of p is

$$p(\tilde{x}) \Leftrightarrow \mathit{false}.$$

□

The completed definition of a predicate is a first order formula that contains the equality symbol; hence, in order to interpret "=" correctly, we also need an equality theory.

Definition 2.3 $\mathrm{CET}_{\mathcal{L}}$, *Clark's Equality Theory for the language* $\mathcal{L}$, consists of the axioms:

- $f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m)$ for all distinct f, g in $\mathcal{L}$;
- $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \rightarrow (x_1 = y_1) \wedge \ldots \wedge (x_n = y_n)$ for all f in $\mathcal{L}$;
- $x \neq t(x)$ for all terms $t(x)$ distinct from x in which x occurs;

together with the usual *equality axioms*, that are needed in order to interpret correctly "=" , which are *reflexivity, symmetry, transitivity,* and $(\tilde{x} = \tilde{y}) \rightarrow (f(\tilde{x}) = f(\tilde{y}))$ for all functions and predicate symbols f in $\mathcal{L}$. □

Note that "=" is always interpreted as two valued.

Definition 2.4 The *Clark's completion of* P wrt the language $\mathcal{L}$, $Comp_{\mathcal{L}}(P)$ consists in the conjunction of the completed definition of all the predicates in P together with $\mathrm{CET}_{\mathcal{L}}$. □

2.2 Domain Closure Axioms

Consider the following example, which is borrowed from [16].

$$P = \{ \quad p \leftarrow \neg q(X). \\ \qquad q(a). \qquad \}$$

The completed definition is

$$p \Leftrightarrow \exists X \; \neg q(X) \quad \wedge \quad q(X) \Leftrightarrow X = a.$$

That is, $Comp_{\mathcal{L}}(P) \models p \Leftrightarrow \exists X \; X \neq a$. If $\mathcal{L} = \{a\}$ then neither p nor $\neg p$ is a logical consequence of $Comp_{\mathcal{L}}(P)$. The problem in this case is due to the fact that $\mathcal{L}$ is a finite language and for this reason $\mathrm{CET}_{\mathcal{L}}$ is not a complete theory.

The two main approaches used in Logic Programming in order to obtain a complete theory out of $\mathrm{CET}_{\mathcal{L}}$ are the following:

- adopting an infinite language (that is a language with infinitely many functions symbols);
- adopting a finite language together with some domain closure axioms.

For a extended study of the subject, we refer to [16].

Definition 2.5 Let $\mathcal{L}$ be a finite language.

The *Domain Closure Axiom*, $\mathrm{DCA}_{\mathcal{L}}$, is

$$x = t_1 \vee x = t_2 \vee \ldots$$

where $t_1, t_2, \ldots$ is the sequence of all the ground $\mathcal{L}$-terms.

The *Weak Domain Closure Axiom*, $\mathrm{WDCA}_{\mathcal{L}}$, is

$$\exists \tilde{y}_1 \; (x = f_1(\tilde{y}_1)) \vee \ldots \vee \exists (\tilde{y}_r \; x = f_r(\tilde{y}_r)).$$

where $f_1, \ldots, f_r$ are all the function symbols in $\mathcal{L}$ and $\tilde{y}_i$ are tuples of variables of the appropriate arity. □

Assuming $\mathrm{DCA}_{\mathcal{L}}$ is equivalent to a restriction to models and interpretations over the Herbrand Universe of $\mathcal{L}$. Note that when $\mathcal{L}$ contains functions of arity greater than zero, then $\mathrm{DCA}_{\mathcal{L}}$ is an infinite disjunction and hence it is not a first-order formula. As opposed to $\mathrm{DCA}_{\mathcal{L}}$, being $\mathcal{L}$ finite, $\mathrm{WDCA}_{\mathcal{L}}$ is a first-order formula.

Example 2.6 Let P be the following program:

$$P = \{ \begin{array}{ll} n(0). & \\ n(s(X)) & \leftarrow n(X). \\ q & \leftarrow \neg n(X). \end{array} \}$$

And let $\mathcal{L} = \mathcal{L}(P)$
The completion of P is

$$n(x) \Leftrightarrow (x = 0) \vee (\exists y\ (x = s(y)) \wedge n(y)) \quad \wedge \quad q \Leftrightarrow \exists y\ \neg n(y)$$

together with $\mathrm{CET}_{\mathcal{L}}$.

On one hand, when we use $\mathrm{DCA}_{\mathcal{L}}$ we have that

$$Comp_{\mathcal{L}}(P) \cup \mathrm{DCA}_{\mathcal{L}} \models \forall x\ n(x).$$

In fact assuming $\mathrm{DCA}_{\mathcal{L}}$ is equivalent to restric ourselves to $\mathcal{L}$-Herbrand models, and the formula $\forall x\ n(x)$ is *true* in the only Herbrand model of P. It follows that:

$$Comp_{\mathcal{L}}(P) \cup \mathrm{DCA}_{\mathcal{L}} \models \neg q.$$

On the other hand, if we use $\mathrm{WDCA}_{\mathcal{L}}$ we have that

$$Comp_{\mathcal{L}}(P) \cup \mathrm{WDCA}_{\mathcal{L}} \not\models \forall x\ n(x).$$

In fact $\mathrm{WDCA}_{\mathcal{L}}$ allows a model which contains, besides the natural numbers, also infinite terms t_i such that for each i, $t_i = s(t_{i+1})$. In such a model each $n(t_i)$ can be *false*. It follows that:

$$Comp_{\mathcal{L}}(P) \cup \mathrm{WDCA}_{\mathcal{L}} \not\models \neg q.$$

□

Assuming $\mathrm{WDCA}_{\mathcal{L}}$ we obtain a semantics which is stronger than the one that adopts $\mathrm{DCA}_{\mathcal{L}}$. In fact $\mathrm{DCA}_{\mathcal{L}} \models \mathrm{WDCA}_{\mathcal{L}}$, and hence if $Comp_{\mathcal{L}}(P) \cup \mathrm{WDCA}_{\mathcal{L}} \models \phi$, then also $Comp_{\mathcal{L}}(P) \cup \mathrm{DCA}_{\mathcal{L}} \models \phi$.

$Comp_{\mathcal{L}}(P) \cup \mathrm{DCA}_{\mathcal{L}}$ and $Comp_{\mathcal{L}}(P) \cup \mathrm{WDCA}_{\mathcal{L}}$ are two different theories that can express the "intended meaning" of the program P. Since we are going to refer to them quite often in the sequel, to simplify the notation we will use the following shorthand:

- $\mathcal{T}_1^{\mathcal{L}}(P) \equiv Comp_{\mathcal{L}}(P) \cup \mathrm{DCA}_{\mathcal{L}}$;
- $\mathcal{T}_2^{\mathcal{L}}(P) \equiv Comp_{\mathcal{L}}(P) \cup \mathrm{WDCA}_{\mathcal{L}}$.

2.3 Three valued semantics for normal programs

Definition 2.7 Let $\mathcal{L}$ be a language. *A three valued (or partial) $\mathcal{L}$-interpretation*, I, is a mapping from the ground atoms of $\mathcal{L}$ into the set $\{$*true*, *false*, *undefined*$\}$. □

A partial interpretation I is represented by an ordered couple, (T, F), of disjoint sets of ground atoms. The atoms in T (resp. F) are considered to be *true* (resp. *false*) in I. T is the positive part of I and is denoted by I^+; equivalently F is denoted by I^-. Atoms which do not appear in either set are considered to be *undefined*.

If I and J are two partial $\mathcal{L}$-interpretations, then $I \cap J$ is the three valued $\mathcal{L}$-interpretation given by $(I^+ \cap J^+, I^- \cap J^-)$, $I \cup J$ is the three valued $\mathcal{L}$-interpretation given by $(I^+ \cup J^+, I^- \cup J^-)$ and we say that $I \subseteq J$ iff $I^+ \subseteq J^+$ and $I^- \subseteq J^-$.

The underlying universe of an $\mathcal{L}$-interpretation is the universe of $\mathcal{L}$-terms. Accordingly when we say that a first order formula ϕ is $true_{\mathcal{L}}$ in I, $I \models_{\mathcal{L}} \phi$, we mean that the quantifiers of ϕ are ranging over the Herbrand Universe of $\mathcal{L}$.

We now give a definition of Fitting's operator [8]. In the sequel of the paper we write $\exists y B\theta$ as a shorthand for $(\exists y B)\theta$, that is, unless explicitly stated, the quantification applies always before the substitution. We denote by $Var(E)$ the set of all the variables in an expression E.

Definition 2.8 Let P be a normal program, $\mathcal{L}$ a language that contains $\mathcal{L}(P)$, and I a three valued $\mathcal{L}$-interpretation. $\Phi_P(I)$ is the three valued $\mathcal{L}$-interpretation defined as follows:

- A ground atom A is *true* in $\Phi_P(I)$, $(A \in \Phi_P(I)^+)$
  iff there exists a clause $c: \; B \leftarrow \widetilde{L}.$ in P whose head unifies with A, $\theta = mgu(A, B)$, and
  $\exists W \widetilde{L}\theta$ is $true_{\mathcal{L}}$ in I
  where W is the set of local variables of c, $W = Var(\widetilde{L}) \backslash Var(B)$.
- A ground atom A is *false* in $\Phi_P(I)$, $(A \in \Phi_P(I)^-)$
  iff for all clauses $c: \; B \leftarrow \widetilde{L}$ in P for which there exists $\theta = mgu(A, B)$ we have that
  $\exists W \widetilde{L}\theta$ is $false_{\mathcal{L}}$ in I
  where W is the set of local variables of c, $W = Var(\widetilde{L}) \backslash Var(B)$. □

Note that Φ_P depends on the language $\mathcal{L}$. It would actually be more appropriate to write $\Phi_P^{\mathcal{L}}$ instead of Φ_P, but then the notation would become more cumbersome.
We adopt the standard notation:

- $\Phi_P^0(I) = I$;
- $\Phi_P^{\alpha+1}(I) = \Phi_P(\Phi_P^{\alpha}(I))$;
- $\Phi_P^{\alpha}(I) = \cup_{\delta<\alpha}\Phi_P^{\delta}(I)$, when α is a limit ordinal.

When the argument is omitted, we assume it to be the empty interpretation $(\emptyset, \emptyset)$: $\Phi_P^{\alpha} = \Phi_P^{\alpha}(\emptyset, \emptyset)$.

Φ_P is a monotonic operator, that is $I \subseteq J$ implies $\Phi_P^I \subseteq \Phi_P^J$; it follows that the Kleene's sequence Φ_P^0, Φ_P^1, ..., Φ_P^k, ..., Φ_P^{ω}, ... is monotonically increasing and it converges to the least fixpoint of Φ_P. Hence there always exists an ordinal α such that $lfp(\Phi_P) = \Phi_P^{\alpha}$. Since Φ_P is monotone but not continuous, α could be greater than ω.

The Φ_P operator characterizes the three valued semantics of $Comp_{\mathcal{L}}$ as stated in the following theorem.

Theorem 2.9 Let P be a normal program, $\mathcal{L}$ a finite language, ϕ any allowed first order formula. Then

(a) $\mathcal{T}_1^{\mathcal{L}}(P) \models \phi$ iff $lfp(\Phi_P) \models_{\mathcal{L}} \phi$;

(b) $\mathcal{T}_2^{\mathcal{L}}(P) \models \phi$ iff for some integer n, $\Phi_P^n \models_{\mathcal{L}} \phi$.

Proof. The first statement follows from theorem 5b in [16]. The second from theorem 6.6 in [11]. □

Note that statement (a) could be restated as follows: $\mathcal{T}_1^{\mathcal{L}}(P) \models \phi$ iff for some ordinal β, $\Phi_P^{\beta} \models_{\mathcal{L}} \phi$.

Example 2.10 Let us refer to the program in example 2.6. If $\mathcal{L} = \mathcal{L}(P)$, we have that

$$
\begin{array}{rcll}
 & & \Phi_P^0 & = (\emptyset, \emptyset). \\
 & & \Phi_P^1 & = (\{n(0)\}, \emptyset). \\
 & & \Phi_P^2 & = (\{n(0), n(s(0))\}, \emptyset). \\
 & & \ldots & \\
 & & \Phi_P^{\omega} & = (\{n(0), \ldots, n(s^k(0)), \ldots\}, \emptyset). \\
lfp(\Phi_P) & = & \Phi_P^{\omega+1} & = (\{n(0), \ldots, n(s^k(0)), \ldots\}, \{q\}).
\end{array}
$$

Hence q is false in $lfp(\Phi_P)$ but not in any Φ_P^n; this coincides with the fact that $\mathcal{T}_1^{\mathcal{L}}(P) \models \neg q$ while $\mathcal{T}_2^{\mathcal{L}}(P) \not\models \neg q$. □

2.4 The Simultaneous Replacement Operation

The replacement operation has been introduced by Tamaki and Sato in [17] for definite programs. Syntactically it consists in substituting a conjunction, $\widetilde{C}$, of literals with another one, $\widetilde{D}$, in the body of a clause. Similarly, *simultaneous* replacement consists in substituting a set of conjunctions of literals $\{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, occurring in the bodies of clauses $\{cl_1, \ldots, cl_p\}$, with another corresponding set of conjunctions $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$. Note that the order of literals is irrelevant for the semantics we are interested in.

Definition 2.11 (simultaneous replacement) Let P be a normal program and $\{cl_1, \ldots, cl_p\}$ a set of clauses of P such that for each i, $cl_i \;=\; A_i \leftarrow \widetilde{C}_{i_1}, \ldots, \widetilde{C}_{i_{r(i)}}, \widetilde{E}_i$. where $\widetilde{C}_{i_1}, \ldots, \widetilde{C}_{i_{r(i)}}$ are conjunctions of literals we want to replace with $\widetilde{D}_{i_1}, \ldots, \widetilde{D}_{i_{r(i)}}$. Now let $\{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$ be the set of conjunctions to be replaced in all the clauses, and $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ be the corresponding set of replacing conjunctions.

- The *simultaneous replacement of* $\{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$ *with* $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ in $\{cl_1, \ldots, cl_p\}$ produces the program $P' = P \backslash \{cl_1, \ldots, cl_p\} \cup \{cl'_1, \ldots, cl'_p\}$, where for each i,
  $cl'_i \;=\; A_i \leftarrow \widetilde{D}_{i_1}, \ldots, \widetilde{D}_{i_{r(i)}}, \widetilde{E}_i$.

replace $(P, \{cl_1, \ldots, cl_p\}, \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}, \{\widetilde{D}_1, \ldots, \widetilde{D}_n\}) \stackrel{\text{def}}{=} P \backslash \{cl_1, \ldots, cl_p\} \cup \{cl'_1, \ldots, cl'_p\}$. □

Note that each $\widetilde{C}_i$ may occur in only one of the clauses $\{cl_1, \ldots, cl_p\}$, this is not restrictive since even if $i \neq j$, $\widetilde{C}_i$ and $\widetilde{C}_j$ may actually represent identical literals.

Some *applicability conditions* are necessary in order to ensure the preservation of the semantics through the transformation. Such conditions depend on the semantics we associate to the program. In the literature some applicability conditions for ordinary replacement are given. In [17] definite programs are considered; the applicability condition requires the replaced atom C and the replacing atom D to be logically equivalent in P and that the size of the smallest proof tree for C is greater or equal to the size of the smallest proof tree for D. Gardner and Shepherdson, in [9], give different conditions for preserving procedural (SLDNF) semantics and the declarative one. Such conditions are based on Clark's (two valued) completion of the program. Also Maher, in [13, 14], studies replacement wrt Success set, Finite Failure Set, Ground Finite Failure Set and Perfect Model semantics. Sato, in [15], considers also replacement of tautologically equivalent formulas in first order programs. Bossi et al. have studied the correctness of this operation wrt the S-semantics for definite programs [4], Fitting's semantics [5] and the Well-Founded semantics for normal programs [7]. In this paper we consider the replacement operation for normal programs and state some applicability conditions for both the three valued semantics mentioned in theorem 2.9. Later on we also consider the case in which the language $\mathcal{L}$ is infinite.

2.5 Equivalences

We give the definition of equivalence of formulas wrt an arbitrary theory $\mathcal{T}$. With $FV(\chi)$ we denote the free variables in a formula χ. We say that a substitution $\theta = (\tilde{t}/\tilde{x})$ is *ground* if all the terms in the tuple $\tilde{t}$ are ground. Given the formulas ζ, χ and ϕ, we denote by $\zeta[\phi/\chi]$ is the formula obtained from ζ by substituting all occurrences of χ as a subformula with ϕ.

Definition 2.12 (equivalence of formulas) Let χ, ϕ be first order formulas and $\mathcal{T}$ be a theory. We say that

- χ *is less defined or equal to* ϕ *wrt* $\mathcal{T}$, $\chi \preceq_{\mathcal{T}} \phi$, iff
  for each closed allowed formula ζ and for each ground substitution σ,

$$\mathcal{T} \models (\neg)\zeta \quad \text{implies} \quad \mathcal{T} \models (\neg)\zeta[\phi\sigma/\chi\sigma];$$

- χ *is equivalent to* ϕ *wrt* $\mathcal{T}$, $\chi \cong_{\mathcal{T}} \phi$, iff $\chi \preceq_{\mathcal{T}} \phi$ and $\phi \preceq_{\mathcal{T}} \chi$; □

Note that, in the above definition, since the domain of σ could be smaller than $FV(\chi)$, $\chi\sigma$ is not necessarily a closed formula.

As far as we are concerned in this paper, a semantics is a theory $\mathcal{T}(P)$ that we associate to the normal program P. Hence two programs P and P' are considered semantically equivalent iff the set of logical consequences of $\mathcal{T}(P)$ and $\mathcal{T}(P')$ coincide.

Definition 2.13 Let $\mathcal{T}(P)$ and $\mathcal{T}(P')$, be the semantic theories associated with the normal programs P and P'. We say that P and P' are *equivalent* wrt $\mathcal{T}$ iff for each allowed formula ϕ

- $\mathcal{T}(P) \models \phi$ iff $\mathcal{T}(P') \models \phi$. □

In the case that P' was obtained by transforming P, the above definition is used to define the *correctness* of a transformation operation.

Definition 2.14 Let P, P' be normal programs and $\mathcal{T}(P)$, $\mathcal{T}(P')$ the associated semantic theories. Suppose that P' is obtained by applying a transformation operation to P. We say that the transformation is

- *$\mathcal{T}$-Partially Correct* if for each allowed formula ϕ, when $\mathcal{T}(P') \models \phi$ then also $\mathcal{T}(P) \models \phi$.
- *$\mathcal{T}$-Complete* if for each allowed formula ϕ, when $\mathcal{T}(P) \models \phi$ then also $\mathcal{T}(P') \models \phi$.
- *$\mathcal{T}$-Totally Correct* or *Safe* if it is both partially correct and complete. This is the case in which P and P' are *equivalent* wrt $\mathcal{T}$. □

Here and in the sequel we assume that the language $\mathcal{L}$ of the theories $\mathcal{T}(P)$and $\mathcal{T}(P')$ contains both $\mathcal{L}(P)$ and $\mathcal{L}(P')$. This is obviously a necessary condition for the correctness of the transformation operation. Note that the transformation is partially correct if all the information contained in (the semantics of) P' was already present in (the semantics of) P, that is if no new knowledge was added to the program during the transformation. On the other hand the transformation is complete if no information is lost during the transformation.

3 Correctness Results wrt $Comp_{\mathcal{L}}(P) \cup \mathbf{DCA}_{\mathcal{L}}$ ($\mathcal{L}$ finite)

In this Section we refer to the semantics given by $Comp(P)_{\mathcal{L}} \cup \mathrm{DCA}_{\mathcal{L}}$, where we assume $\mathcal{L}$ to be finite. Adopting $\mathrm{DCA}_{\mathcal{L}}$ is equivalent to restricting our attention to Herbrand models (on the language $\mathcal{L}$) and in this particular case we have that:

a) there always exists a minimal Herbrand model (wrt $\subseteq$);

b) an allowed formula is *true* in all Herbrand models iff it is *true* in the minimal one.

Moreover the minimal model coincides with the interpretation given by $lfp(\Phi_P)$. So to check if an allowed formula is a logical consequence of $\mathcal{T}_1^{\mathcal{L}}$ it is sufficient to check if it is true in $lfp(\Phi_P)$. These properties are proved in [8], and [11] and are summarized in statement (a) of theorem 2.9. In the case that $\mathcal{L} = \mathcal{L}(P)$, this semantics is called Fitting's model semantics [8].

By Theorem 2.9 and Definition 2.14, we can easily characterize the correctness of the transformation by referring to the least fixed point of the Φ_P operator.

Lemma 3.1 Let P, P' be normal programs and $\mathcal{L}$ be a finite language. Suppose that P' is obtained by applying a transformation operation to P. Then the operation is

- $\mathcal{T}_1^{\mathcal{L}}$-Partially Correct iff $lfp(\Phi_P) \supseteq lfp(\Phi_{P'})$;
- $\mathcal{T}_1^{\mathcal{L}}$-Complete iff $lfp(\Phi_P) \subseteq lfp(\Phi_{P'})$;
- $\mathcal{T}_1^{\mathcal{L}}$-Totally Correct iff $lfp(\Phi_P) = lfp(\Phi_{P'})$. □

3.1 Partial Correctness

When we replace the conjunction $\widetilde{C}$ with $\widetilde{D}$, the "first requirement" we ask for is the equivalence of $\widetilde{C}$ and $\widetilde{D}$ wrt $\mathcal{T}_1^{\mathcal{L}}(P)$. After all it would make no sense to replace $\widetilde{C}$ with something which has a different meaning.

By Theorem 2.9 and Definition 2.12 we can characterize the equivalence of formulas in $\mathcal{T}_1^{\mathcal{L}}(P)$ by referring to the least fixed point of the operator Φ_P. First we introduce the three valued operator $\Rightarrow$, which is "one side" of $\Leftrightarrow$ and is defined as follows: $\phi \Rightarrow \chi$ is *true* iff χ is more defined than ϕ, that is if ϕ and χ are both *true* (or both *false*) or if ϕ is *undefined*. In any other case $\phi \Rightarrow \chi$ is *false*.

Proposition 3.2 Let χ, ϕ be first order allowed formulas and P be a normal program. The following statements are equivalent:

(a) $\chi \preceq_{\mathcal{T}_1^{\mathcal{L}}(P)} \phi$;

(b) $lfp(\Phi_P) \models_{\mathcal{L}} \chi \Rightarrow \phi$.

Proof.
(a) implies (b).
By the definition of the operator $\Rightarrow$, (b) is equivalent to
for each tuple of $\mathcal{L}$-terms $\tilde{t}$, $lfp(\Phi_P) \models_{\mathcal{L}} (\neg)\chi(\tilde{t}/\tilde{x})$ implies $lfp(\Phi_{P'}) \models_{\mathcal{L}} (\neg)\phi(\tilde{t}/\tilde{x})$.
By Theorem 2.9 this is equivalent to
for each tuple of $\mathcal{L}$-terms $\tilde{t}$, $\mathcal{T}_1^{\mathcal{L}}(P) \models (\neg)\chi(\tilde{t}/\tilde{x})$ implies $\mathcal{T}_1^{\mathcal{L}}(P) \models (\neg)\phi(\tilde{t}/\tilde{x})$.
This is immediate by Definition 2.12.

(b) implies (a).
Let ζ be any allowed formula such that $\mathcal{T}_1^{\mathcal{L}} \models \zeta$, σ be any ground substitution; we have to prove that $\mathcal{T}_1^{\mathcal{L}} \models \zeta[\phi\sigma/\chi\sigma]$.
If ζ does not contain $\chi\sigma$ as a subformula then the result holds trivially, so let us suppose that ζ contains $\chi\sigma$ as a subformula. The proof proceeds by induction on the structure of ζ.

Base step: $\zeta \equiv \chi\sigma$. By Theorem 2.9, $\mathcal{T}_1^{\mathcal{L}} \models \chi\sigma$ implies that $lfp(\Phi_P) \models_{\mathcal{L}} \chi\sigma$.
By (b) this implies that $lfp(\Phi_P) \models_{\mathcal{L}} \phi\sigma$, and, by Theorem 2.9, that $\mathcal{T}_1^{\mathcal{L}} \models \phi\sigma$.
Since $\phi\sigma \equiv \zeta[\phi\sigma/\chi\sigma]$, this implies the thesis.

Induction step: we have to consider four cases:

1) $\zeta \equiv \triangle\ \zeta_1$, where $\triangle$ is any allowed unary connective. The result holds trivially, since by the inductive hypothesis, $\mathcal{T}_1^{\mathcal{L}} \models (\neg)\zeta_1$ implies $\mathcal{T}_1^{\mathcal{L}} \models (\neg)\zeta_1[\phi\sigma/\chi\sigma]$.

2) $\zeta \equiv \zeta_1 \diamond \zeta_2$, where $\diamond$ is any allowed binary connective. For $i \in \{1,2\}$, either ζ_i does not contain an instance of χ as a subformula, in which case the result holds trivially, or the inductive hypothesis applies to ζ_i.

3) $\zeta \equiv \forall w\ \zeta_1(w)$.
Suppose that $\mathcal{T}_1^{\mathcal{L}} \models \forall w\ \zeta_1(w)$.
This is equivalent to: for any $\mathcal{L}$-term t, $\mathcal{T}_1^{\mathcal{L}} \models \zeta_1(t)$.
For each $\mathcal{L}$-term t, let γ_t be the substitution (t/w), by the inductive hypothesis, we have that for any $\mathcal{L}$-term t, $\mathcal{T}_1^{\mathcal{L}} \models \zeta_1(t)[\phi\sigma\gamma_t/\chi\sigma\gamma_t]$.
Since $\mathrm{DCA}_{\mathcal{L}}$ forces the quantification to be over $\mathcal{L}$-terms, and $\mathrm{DCA}_{\mathcal{L}}$ is included in $\mathcal{T}_1^{\mathcal{L}}$, this implies that $\mathcal{T}_1^{\mathcal{L}} \models \forall w\ \zeta_1(w)[\phi\sigma/\chi\sigma]$.
On the other hand, for the case when $\mathcal{T}_1^{\mathcal{L}} \models \neg\forall w\ \zeta_1(w)$, a similar reasoning applies.

4) $\zeta \equiv \exists w\ \zeta_1(w)$
This falls into the previous case, since $\exists w\ \zeta_1(w) \equiv \neg\forall w\ \neg\zeta_1(w)$. □

Example 3.3 Let us consider the following program:

$$
\begin{array}{ll}
 & m1(El,[El \mid Tail],s(0)). \\
 & m1(El,[X \mid Tail],s(N)) \quad \leftarrow m1(El,Tail,N). \\
 & m2(El,[El \mid Tail]). \\
 & m2(El,[X \mid Tail]) \quad \leftarrow m2(El,Tail). \\
d: & intersect(L1,L2) \quad \leftarrow m1(El,L1,N1), m1(El,L2,N2).
\end{array}
$$

Predicates $m1$ and $m2$ behave like "member" predicates. The only difference between the two is that $m1$ "reports", as third argument, the location where element El has been found. As far as the definition of *intersect* goes, this is totally unnecessary, and we can replace the conjunction $m1(El, L1, N1), m1(El, L2, N2)$ with the new conjunction $m2(El, L1), m2(El, L2)$ in the body of d, without affecting the semantics of the program. In practice we want to replace clause d with

$d' : intersect(L1, L2) \leftarrow m2(El, L1), m2(El, L2)$.

Now observe that the completed definition of *intersect* before the transformation is

$$intersect(L1, L2) \Leftrightarrow \exists N,\ M.\ m1(El, L1, N), m1(El, L2, M), \tag{1}$$

while after the transformation it is

$$intersect(L1, L2) \Leftrightarrow m2(El, L1), m2(El, L2). \tag{2}$$

When applying a replacement we want the replacing conjunction to be semantically equivalent to the replaced one. In this particular case, by Proposition 3.2 we can formalize this statement by requiring the equivalence of the two "bodies", (1) and (2), of the completed definition of *intersect*, that is, we require that

$$\exists N,\ M.\ m1(El, L1, N), m1(El, L2, M) \cong_{\mathcal{T}_1^{\mathcal{L}}(P)} m2(El, L1), m2(El, L2).$$

In this example we have specified two existentially quantified variables: N and M. In the sequel, when replacing, say, $\widetilde{C}$ with $\widetilde{D}$, we will always specify a set X of "local" variables, namely variables which can appear in either $\widetilde{C}$ or $\widetilde{D}$ (or both) but cannot occur in the rest of the clause where $\widetilde{C}$ is found. Consequently, our first requirement will be the equivalence of $\exists X\widetilde{C}$ and $\exists X\widetilde{D}$. Such an equivalence is weaker than the equivalence between $\widetilde{C}$ and $\widetilde{D}$, while being still sufficient for our purposes. □

We now formalize this concept of local variables for simultaneous replacement.

Definition 3.4 (Locality Property) Refer to the notation of Definition 2.11: $\{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$ is the set of conjunctions to be replaced with $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ in the clauses $\{cl_1, \ldots, cl_p\}$. Let $i \in [1, n]$, and let cl_j be the clause in which $\widetilde{C}_i$ occurs. A set of variables X_i satisfies the *Locality Property* with respect to $\widetilde{C}_i$ and $\widetilde{D}_i$ if the following holds:

- $X_i \subseteq Var(\widetilde{C}_i) \cup Var(\widetilde{D}_i)$ and the variables in X_i do not occur anywhere else neither in the clause cl_j, where $\widetilde{C}_i$ is found, nor, after replacement, in cl'_j, where $\widetilde{D}_i$ is found. □

Note that the locality property is trivially satisfied when X_i is empty. Note also that the locality property implies, that if $\widetilde{C}_h$ and $\widetilde{C}_k$ occur in the same clause then the corresponding X_h and X_k are disjoint.

Next we give the theorem on partial correctness of the replacement operation we were aiming at. It shows that the equivalence between the replacing and the replaced literals is sufficient to ensure the partial correctness of the replacement operation.

Theorem 3.5 (partial correctness) Let $\mathcal{L}$ be a finite language. In the hypothesis of Definition 2.11, if for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, there exists a (possibly empty) set of variables X_i satisfying the locality property wrt $\widetilde{C}_i$ and $\widetilde{D}_i$ such that

$$\exists X_i \widetilde{D}_i \preceq_{\mathcal{T}_1^{\mathcal{L}}(P)} \exists X_i \widetilde{C}_i,$$

then $lfp(\Phi_P) \supseteq lfp(\Phi_{P'})$.

Proof. Let us first recall the notation adopted.

P is the original program,

$\{cl_1, \ldots, cl_p\}$ is the set of clauses of P which will be affected by the simultaneous replacement operation, each cl_i has the form

$cl_i \ = \ A_i \leftarrow \widetilde{C}_{i_1}, \ldots, \widetilde{C}_{i_{r(i)}}, \widetilde{E}_i$.

P' is the transformed program, obtained by replacing each $\widetilde{C}_i$ by $\widetilde{D}_i$. $P' = P \backslash \{cl_1, \ldots, cl_p\} \cup \{cl'_1, \ldots, cl'_p\}$ where each cl'_i has the form

$cl'_i \;=\; A_i \leftarrow \widetilde{D}_{i_1}, \ldots, \widetilde{D}_{i_{r(i)}}, \widetilde{E}_i.$

The proof is by contradiction. Let us suppose $lfp(\Phi_P) \not\supseteq lfp(\Phi_{P'})$. Since the sequence $\Phi^0_{P'}, \Phi^1_{P'}, \ldots$ is monotonically increasing and $\Phi^0_{P'} = (\emptyset, \emptyset) \subseteq lfp(\Phi_P)$, there has to be an ordinal α such that

$$lfp(\Phi_P) \supseteq \Phi^\alpha_{P'} \quad \text{and} \quad lfp(\Phi_P) \not\supseteq \Phi^{\alpha+1}_{P'} = \Phi_{P'}(\Phi^\alpha_{P'}).$$

Hence $lfp(\Phi_P) \not\supseteq \Phi_{P'}(lfp(\Phi_P))$, and since Φ is monotone, from the first inclusion it follows that

$$\Phi_{P'}(lfp(\Phi_P)) \supseteq \Phi_{P'}(\Phi^\alpha_{P'}).$$

Since $\Phi_P(lfp(\Phi_P)) = lfp(\Phi_P)$ we have that

$$\Phi_P(lfp(\Phi_P)) \not\supseteq \Phi_{P'}(lfp(\Phi_P)). \tag{3}$$

Let X_i be any set of variables which satisfies the locality property. Note that with the exception of clauses $\{cl_1, \ldots, cl_p\}$, P is just like P'. Hence if for each i, $\exists X_i \widetilde{C}_i$ and $\exists X_i \widetilde{D}_i$ have the same meaning in a given interpretation I, that is if $I \models_{\mathcal{L}} \exists X_i \widetilde{C}_i \Leftrightarrow \exists X_i \widetilde{D}_i$, then $\Phi_P(I) = \Phi_{P'}(I)$. It follows that whenever $\Phi_P(I) \neq \Phi_{P'}(I)$, then there exists an integer j such that $\exists X_j \widetilde{C}_j$ and $\exists X_j \widetilde{D}_j$ have different meanings in I.

This idea is formalized and extended in the following Claim, whose proof is given in the Appendix A.

Claim 1 Let I, I' be two partial interpretations. If $I' \subseteq I$ but $\Phi_{P'}(I') \not\subseteq \Phi_P(I)$, then there exist a conjunction $\widetilde{C}_j \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$ and a ground substitution θ such that:

- either $I' \models_{\mathcal{L}} \exists X_j \widetilde{D}_j \theta$ while $I \not\models_{\mathcal{L}} \exists X_j \widetilde{C}_j \theta$;
- or $I' \models_{\mathcal{L}} \neg\exists X_j \widetilde{D}_j \theta$ while $I \not\models_{\mathcal{L}} \neg\exists X_j \widetilde{C}_j \theta$.

From this Claim and (3) it follows that there exists an integer j and a ground substitution θ such that $\exists X_j \widetilde{D}_j \theta$ is $true_{\mathcal{L}}$ (or $false_{\mathcal{L}}$) in $lfp(\Phi_P)$, while $\exists X_j \widetilde{C}_j \theta$ is not. This, by Proposition 3.2 (c), contradicts hypothesis (ii). □

3.2 Semantic Delay and Dependency Degree

As we proved in the previous Section, if X is a set which satisfies the locality property, the equivalence of $\exists X\widetilde{C}$ and $\exists X\widetilde{D}$ is sufficient to guarantee the partial correctness of the replacement of $\widetilde{C}$ with $\widetilde{D}$. Unfortunately this may not be enough to obtain total correctness. For that we need the equivalence to hold also after the transformation and the equivalence can be destroyed when $\widetilde{D}$ depends on the modified clause. This is shown by the next example.

Example 3.6 Let P be the following definite program:

$$\begin{array}{lll} P = \{ & & p \leftarrow q. \\ & cl: & q \leftarrow r. \\ & & r. \qquad\qquad \} \end{array}$$

Let also $\mathcal{L} = \mathcal{L}(P)$. In this case $lfp(\Phi_P) = (\{p, q, r\}, \emptyset)$. p, q and r are all *equivalent* wrt $\mathcal{T}_1^{\mathcal{L}}(P)$, but if we replace r with p in the body of cl we obtain

$$\begin{array}{lll} P' = \{ & & p \leftarrow q. \\ & cl': & q \leftarrow p. \\ & & r. \qquad\qquad \} \end{array}$$

which is by no means equivalent to the previous program. In fact $lfp(\Phi_{P'}) = (\{r\}, \emptyset)$. We have introduced a loop and p and q are no more *true*. □

In order to obtain the desired completeness results we introduce two more concepts: the *semantic delay* and the *dependency degree*. They are meant to express relations between first order formulas, such as conjunctions of literals, in terms of their semantic properties.

Consider the following definite program:

$$P = \{ \quad \begin{array}{lll} m(X) & \leftarrow & n(s(X)). \\ n(0). & & \\ n(s(X)) & \leftarrow & n(X). \end{array} \quad \}$$

The predicates m and n have exactly the same meaning, but in order to refute the goal $\leftarrow m(s(0))$. we need four resolution steps, while for refuting $\leftarrow n(s(0))$. two steps are sufficient. Each time $\leftarrow n(t)$. has a refutation (or finitely fails) with j resolution steps, $\leftarrow m(t)$. has a refutation (or fails) with k resolution steps, where $k \leq j+2$. By transposing this idea into the three valued semantics we are adopting, we have that each time $n(t)$ is *true* (or *false*) in Φ_P^j, $m(t)$ is *true* (resp. *false*) in Φ_P^{j+2}. We can formalize this intuitive idea by saying that *the semantic delay of m wrt n is 2.*

Definition 3.7 (semantic delay in $lfp(\Phi_P)$) Let P be a normal program, χ and ϕ be first order formulas, and $\tilde{x} = \{x_1, \ldots, x_k\} = FV(\chi) \cup FV(\phi)$. Suppose that $\phi \preceq_{\mathcal{T}_1^{\mathcal{L}}(P)} \chi$.

- *The semantic delay of χ wrt ϕ in $lfp(\Phi_P)$* is the least integer k such that, for each ordinal α and each k-uple of $\mathcal{L}$-terms $\tilde{t}$: if $\Phi_P^\alpha \models_{\mathcal{L}} (\neg)\phi(\tilde{t}/\tilde{x})$, then $\Phi_P^{\alpha+k} \models_{\mathcal{L}} (\neg)\chi(\tilde{t}/\tilde{x})$. □

Since we are assuming that $\phi \preceq_{\mathcal{T}_1^{\mathcal{L}}(P)} \chi$, if $\phi(\tilde{t}/\tilde{x})$ is *true* in some Φ_P^α, then there exists an ordinal β such that $\chi(\tilde{t}/\tilde{x})$ is *true* in Φ_P^β.
Intuitively, $\phi(\tilde{t}/\tilde{x})$ is $true_{\mathcal{L}}$ in Φ_P^α iff its truth has been proved from scratch in at most α steps. The semantic delay of χ wrt ϕ shows how many steps later than $\phi(\tilde{t}/\tilde{x})$, we determine the truth value of $\chi(\tilde{t}/\tilde{x})$ (at worse).

Example 3.8 Let P be the following program:

$$P = \{ \quad \begin{array}{lll} p(0). & & q(0). \\ p(s(0)). & & q(s(X)) \leftarrow q(X). \\ p(s(s(X))) \leftarrow p(X). & & \end{array} \quad \}$$

Let $\mathcal{L} = \mathcal{L}(P)$. p and q both compute natural numbers, $p(X) \cong_{\mathcal{T}_1^{\mathcal{L}}(P)} q(X)$. but while $q(s^k(0))$ is *true* starting from Φ_P^{k+1}, $p(s^k(0))$ is *true* starting from $\Phi_P^{(k/2)+1}$. The delay of $p(X)$ wrt $q(X)$ in $lfp(\Phi_P)$ is zero, in fact if for some ground term t and ordinal α, $q(t)$ is *true* (resp. *false*) in Φ_P^α, then $p(t)$ is also *true* (resp. *false*) in Φ_P^α. Vice versa, the delay of $q(X)$ wrt $p(X)$ is not definable in fact there exists no integer $m < \omega$ such that if, for some ground term t and ordinal α, $p(t)$ is *true* (resp. *false*) in Φ_P^α, then $q(t)$ is *true* (resp. *false*) in $\Phi_P^{\alpha+m}$. □

A simple property of semantic delay which will be used in the sequel is the following.

Lemma 3.9 If $d: \; A \leftarrow \widetilde{L}.$ is the only clause in a program P whose head unifies with an atom A, and W is the set of variables local to the body of d, $W = Var(\widetilde{L}) \backslash Var(A)$, then

- $lfp(\Phi_P) \models_{\mathcal{L}} (A \Leftrightarrow \exists W \widetilde{L})$, that is, $A \cong_{\mathcal{T}_1^{\mathcal{L}}(P)} \exists W \widetilde{L}$;
- the delay of A wrt $\exists W \widetilde{L}$ in $lfp(\Phi_P)$ is one.

Proof. It is a straightforward application of the definition of Fitting's operator, since, by Definition 2.8, for all integers r and substitution θ, $(\exists W \widetilde{L})\theta$ is $true_{\mathcal{L}}$ ($false_{\mathcal{L}}$) in Φ_P^r iff $A\theta$ is *true* (*false*) in Φ_P^{r+1}. □

Now we want to introduce one further concept: the *dependency degree*. Let us consider the following normal program:

$$P = \{ \begin{array}{llll} c1: & p & \leftarrow & \neg q, s. \\ c2: & q & \leftarrow & r. \\ c3: & r. & & \\ c4: & s & \leftarrow & q. \end{array} \}$$

The definitions of the atoms p, q, s and r, all depend from clause $c3$. Informally we could say that *the dependency degree of the predicate p over clause $c3$ is two*, as the shortest derivation path from a clause having head p to $c3$ contains two arcs: the first from $c1$ to $c2$, through the negative literal $\neg q$; the second from $c2$, to $c3$, through the atom r. Similarly, the dependency degree of q and s on $c3$ are respectively one and two and the dependency degree of r on $c3$ is zero. The next definition formalizes this intuitive notion. The atom A and the clause cl are assumed to be standardized apart.

Definition 3.10 (dependency degree) Let P be a program, cl a clause of P and A an atom. *The dependency degree of A (and $\neg A$) on cl*, $depen_P(A, cl)$, is

0 if A unifies with the head of cl;

n+1 if A does not unify with the head of cl and n is the least integer such that there exists a clause $C \leftarrow C_1, \ldots, C_k.$ in P, whose head unifies with A via mgu, say, θ, and, for some i, $depen_P(C_i\theta, cl) = n$;

ω when there exists no such n. In this case we say that A *is independent from cl.*

Now let $\widetilde{L} = L_1, \ldots, L_n$ be a conjunction of literals. *The dependency degree of $\widetilde{L}$ on cl* is equal to the least dependency degree of one of its elements on cl, $depen_P(\widetilde{L}, cl) = inf\{depen_P(L_i, cl),$ where $1 \leq i \leq n\}$. Similarly, $\widetilde{L}$ *is independent from cl* iff all its components are independent from cl. □

Example 3.11 Consider the following normal program:

$$P = \{ \begin{array}{llll} d: & p(X) & \leftarrow & \neg q(X). \\ cl: & r & \leftarrow & \ldots, \neg q(t), \ldots \\ & & \ldots & \end{array} \}$$

where d is the only clause defining the predicate symbol p. Let also $\mathcal{L} = \mathcal{L}(P)$. By Lemma 3.9 $p(X) \cong_{\mathcal{T}_1^{\mathcal{L}}(P)} \neg q(X)$. Now, if we replace $\neg q(t)$ with $p(t)$ in cl, we obtain the following program:

$$P' = \{ \begin{array}{llll} d: & p(X) & \leftarrow & \neg q(X). \\ cl: & r & \leftarrow & \ldots, p(t), \ldots \\ & & \ldots & \end{array} \}$$

which has the same semantics of the previous one, that is $lfp(\Phi_P) = lfp(\Phi_{P'})$. This holds even if the definition of p is not independent from cl; that is, even if we are exposed to the risk of introducing a loop, losing completeness. But in this case we can show that "there is no room for introducing a loop". in fact replacing $\neg q(t)$ by $p(t)$ in cl preserves the semantics of the initial program if

- either p *does not depend on cl* (in this case no loop can be introduced) or
- *the dependency level of p on cl* (this is how big the loop would be) *is greater or equal to the semantic delay of $p(X)$ wrt $\neg q(X)$* (this is the space where the loop would be introduced).

By lemma 3.9 the *delay* of $p(X)$ wrt $\neg q(X)$ in $lfp(\Phi_P)$ is one; moreover, since d is the only clause defining predicate p and $d \neq cl$, $depen_P(p(X), cl) > 0$, thus satisfying the above conditions. □

3.3 Completeness

We want a completeness result which formalizes the idea outlined in the previous example and that matches with Theorem 3.5.

Let us first establish the notation and state a few simple remarks:

Notation.
P is the original program,
$\{cl_1, \ldots, cl_p\}$ is the set of clauses of P which will be affected by the simultaneous replacement operation, each cl_i has the form

$cl_i \;=\; A_i \leftarrow \widetilde{C}_{i_1}, \ldots, \widetilde{C}_{i_{r(i)}}, \widetilde{E}_i;$

P' is the transformed program, obtained by replacing each $\widetilde{C}_i$ by $\widetilde{D}_i$: $P' = P \backslash \{cl_1, \ldots, cl_p\} \cup \{cl'_1, \ldots, cl'_p\}$ where each cl'_i has the form

$cl'_i \;=\; A_i \leftarrow \widetilde{D}_{i_1}, \ldots, \widetilde{D}_{i_{r(i)}}, \widetilde{E}_i.$ □

The first remark states that when a conjunction of literals $\widetilde{L}$ is *independent* from clauses $\{cl_1, \ldots, cl_p\}$ then its meaning does not change when replacing $\{cl_1, \ldots, cl_p\}$ with $\{cl'_1, \ldots, cl'_p\}$.

Remark 3.12 Let $\widetilde{L}$ be a conjunction of literals independent from the clauses $\{cl_1, \ldots, cl_p\}$ in P. Let $W = Var(\widetilde{L})$, Then, for each ordinal α,

- $\Phi_P^\alpha \models_{\mathcal{L}} (\neg)\exists W\widetilde{L}$ iff $\Phi_{P'}^\alpha \models_{\mathcal{L}} (\neg)\exists W\widetilde{L}$.

Consequently

- $lfp(\Phi_P) \models_{\mathcal{L}} (\neg)\exists W\widetilde{L}$ iff $lfp(\Phi_{P'}) \models_{\mathcal{L}} (\neg)\exists W\widetilde{L}$. □

The following lemma represents an important step in the proof of the completeness result.

Let I be an $\mathcal{L}$-interpretation and B a ground atom that can be proved *true* (or *false*), starting from I, in m steps, that is, B is *true* in $\Phi_P^m(I)$. The lemma states that if the dependency level of B on $\{cl_1, \ldots, cl_p\}$ is greater or equal to m, then the clauses $\{cl_1, \ldots, cl_p\}$ cannot have been used in the proof of B, hence B is *true* in $\Phi_{P'}^m(I)$ too.

Lemma 3.13 Let B be a ground atom, m a natural number such that $depen_P(B, \{cl_1, \ldots, cl_p\}) \geq m$; then

- B is *true* (resp. *false*) in $\Phi_P^m(I)$ iff B is *true* (resp. *false*) in $\Phi_{P'}^m(I)$.

Proof. The proof is by induction on m.

The base of the induction ($m = 0$) is trivial, since $\Phi_{P'}^0(I) = \Phi_P^0(I) = I$.

Induction step: $m > 0$. We will now proceed as follows: in a) we show that if B is *true* (resp. not *false*) in $\Phi_P^m(I)$, then it is also *true* (resp. not *false*) in $\Phi_{P'}^m(I)$. That is, we show that if B is *true* in $\Phi_P^m(I)$, then it is also *true* in $\Phi_{P'}^m(I)$; and, by contradiction, that if B is *false* in $\Phi_{P'}^m(I)$, then it is also *false* in $\Phi_P^m(I)$. In b) we consider the converse implications. This will be sufficient to prove the thesis.

a) Let us assume B *true* (resp. not *false*) in $\Phi_P^m(I)$. There has to be a clause $c \in P$ and a ground substitution γ such that $head(c)\gamma = B$ and $body(c)\gamma$ is *true* (resp. not *false*) in $\Phi_P^{m-1}(I)$. It follows that, for each literal L belonging to $body(c)\gamma$:

- L is *true* (resp. not *false*) in $\Phi_P^{m-1}(I)$;
- $depen_P(L, \{cl_1, \ldots, cl_p\}) \geq m - 1$.

Then, from the inductive hypothesis each L is *true* (resp. not *false*) in $\Phi_{P'}^{m-1}(I)$.
Since $depen_P(B, \{cl_1, \ldots, cl_p\}) \geq m > 0$, B does not unify with the head of any clause in $\{cl_1, \ldots, cl_p\}$, that is $c \notin \{cl_1, \ldots, cl_p\}$. Hence $c \in P'$ and B is *true* (not *false*) in $\Phi_{P'}^m(I)$.

b) Now we have to prove that if B is *true* (not *false*) in $\Phi_{P'}^m(I)$, then it is also *true* (not *false*) in $\Phi_P^m(I)$. This part is omitted as it is perfectly symmetrical to the previous one. □

The previous lemma leads to the following generalization.

Lemma 3.14 Let $\widetilde{L}$ be a conjunction of literals, $W = Var(\widetilde{L})$ and I be an $\mathcal{L}$-interpretation. Suppose that, for some integer m, $depen_P(\widetilde{L}, \{cl_1, \ldots, cl_p\}) \geq m$, then,

- $\Phi_P^m(I) \models_{\mathcal{L}} (\neg)\exists W\widetilde{L}$ iff $\Phi_{P'}^m(I) \models_{\mathcal{L}} (\neg)\exists W\widetilde{L}$.

Proof. Let $\widetilde{L} = L_1, \ldots, L_j$. Observe that $depen_P(\widetilde{L}, \{cl_1, \ldots, cl_p\}) \geq m$ implies that for $i \in [1, j]$, $depen_P(L_i, \{cl_1, \ldots, cl_p\}) \geq m$.

Suppose first that $\exists W \widetilde{L}$ is $true_{\mathcal{L}}$ in $\Phi_P^m(I)$. Then for some ground substitution θ, $Dom(\theta) = W$, $\widetilde{L}\theta$ is *true* in $\Phi_P^m(I)$. Then for $i \in [1, j]$, $L_i\theta$ is *true* in $\Phi_P^m(I)$, and by Lemma 3.13, it is true also in $\Phi_{P'}^m(I)$. Hence the conjunction $\widetilde{L}\theta$ is *true* in $\Phi_{P'}^m(I)$. It follows that $\exists W \widetilde{L}$ is $true_{\mathcal{L}}$ in $\Phi_{P'}^m(I)$.

Now suppose that $\exists W \widetilde{L}$ is $false_{\mathcal{L}}$ in $\Phi_P^m(I)$. Then for each ground substitution θ, $Dom(\theta) = W$, $\widetilde{L}\theta$ is *false* in $\Phi_P^m(I)$. That is, for each of the above θ, there exists an $i \in [1, j]$ such that $L_i\theta$ is *false* in $\Phi_P^m(I)$. By Lemma 3.13 $L_i\theta$ is also *false* in $\Phi_{P'}^m(I)$. Hence $\widetilde{L}\theta$ is *false* in $\Phi_{P'}^m(I)$. It follows that $\exists W \widetilde{L}$ is $false_{\mathcal{L}}$ in $\Phi_{P'}^m(I)$. □

Now we can prove the result we were looking for.

Theorem 3.15 (completeness) In the hypothesis of Definition 2.11 for simultaneous replacement, if for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, there exists a (possibly empty) set of variables X_i satisfying the locality property wrt $\widetilde{C}_i$ and $\widetilde{D}_i$ such that

$$\exists X_i \widetilde{C}_i \preceq_{\mathcal{T}_1^{\mathcal{L}}(P)} \exists X_j \widetilde{D}_j ,$$

and if one of the following two conditions holds:

(a) $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ are all independent from the clauses $\{cl_1, \ldots, cl_p\}$; or

(b) there exists an integer m such that, for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, and each $cl_j \in \{cl_1, \ldots, cl_p\}$:
- the delay of $\exists X_i \widetilde{D}_i$ wrt $\exists X_i \widetilde{C}_i$ in $lfp(\Phi_P)$ is less or equal to m, and
- $depen_P(\widetilde{D}_i, cl_j) \geq m$;

then $lfp(\Phi_P) \subseteq lfp(\Phi_{P'})$.

Proof. First we need to establish a Claim similar to the one in the proof of Theorem 3.5.

Claim 2 Let I, I' be two partial interpretations. If $I \subseteq I'$ but $\Phi_P(I) \not\subseteq \Phi_{P'}(I')$, then there exist a conjunction $\widetilde{C}_j \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$ and a ground substitution θ such that:

- either $I \models_{\mathcal{L}} \exists X_j \widetilde{C}_j\theta$ while $I' \not\models_{\mathcal{L}} \exists X_j \widetilde{D}_j\theta$;
- or $I \models_{\mathcal{L}} \neg\exists X_j \widetilde{C}_j\theta$ while $I' \not\models_{\mathcal{L}} \neg\exists X_j \widetilde{D}_j\theta$;

Proof. The proof is identical to the one given in the Appendix A for Claim 1 in Theorem 3.5, and it is omitted here. □

The proof of the Theorem, is by contradiction.
Let us suppose $lfp(\Phi_P) \not\subseteq lfp(\Phi_{P'})$. By the same argument used in the proof of 3.5, it follows that there exists an ordinal α such that:

$$lfp(\Phi_{P'}) \supseteq \Phi_P^{\alpha} \quad \text{and} \quad lfp(\Phi_{P'}) \not\supseteq \Phi_P^{\alpha+1}.$$

Since $\Phi_{P'}(lfp(\Phi_{P'})) = lfp(\Phi_{P'})$, it follows that $\Phi_{P'}(lfp(\Phi_{P'})) \supseteq \Phi_P(\Phi_P^{\alpha})$.
From Claim 2 there exists an integer j and a ground substitution θ such that:

$$\exists X_j \widetilde{C}_j\theta \text{ is } true_{\mathcal{L}} \text{ (or } false_{\mathcal{L}}) \text{ in } \Phi_P^{\alpha}, \quad \text{while} \quad \exists X_j \widetilde{D}_j\theta \text{ is not } true_{\mathcal{L}} \text{ (resp. not } false_{\mathcal{L}}) \text{ in } lfp(\Phi_{P'}). \quad (4)$$

Let us distinguish two cases.

1) Condition (a) of the hypothesis applies, and $\widetilde{D}_j$ is independent from $\{cl_1, \ldots, cl_p\}$.
Since $\Phi_P^{\alpha} \subseteq lfp(\Phi_P)$, from the left hand side of (4) it follows that $\exists X_j \widetilde{C}_j\theta$ is also $true_{\mathcal{L}}$ (resp. $false_{\mathcal{L}}$) in $lfp(\Phi_P)$.
Hence, by the hypothesis and Proposition 3.2, also $\exists X_j \widetilde{D}_j\theta$ is $true_{\mathcal{L}}$ (resp. $false_{\mathcal{L}}$) in $lfp(\Phi_P)$. Because of

condition (a) and Remark 3.12 it follows that $\exists X_j \widetilde{D}_j\theta$ is $true_{\mathcal{L}}$ (resp. $false_{\mathcal{L}}$) in $lfp(\Phi_{P'})$. This contradicts the left hand side of (4).

2) Condition (b) of the hypothesis applies. The delay of $\exists X_j \widetilde{D}_j$ wrt $\exists X_j \widetilde{C}_j$ is not greater that m, hence from the left hand side of (4) it follows that $\exists X_j \widetilde{D}_j\theta$ is $true_{\mathcal{L}}$ (or $false_{\mathcal{L}}$) in $\Phi_P^{\alpha+m}$,
that is, $\exists X_j \widetilde{D}_j\theta$ is $true_{\mathcal{L}}$ (resp. $false_{\mathcal{L}}$) in $\Phi_P^m(\Phi_P^\alpha)$.
Since by (b), $depen_P(\widetilde{D}_j\theta, \{cl_1, \ldots, cl_p\}) \geq m$, from lemma 3.14 it follows that

$$\exists X_j \widetilde{D}_j\theta \text{ is } true_{\mathcal{L}} \text{ (resp. } false_{\mathcal{L}}) \text{ in } \Phi_{P'}^m(\Phi_P^\alpha).$$

Now $\Phi_P^\alpha \subseteq lfp(\Phi_{P'})$ and $\Phi_{P'}$ is monotone, then

$$\exists X_j \widetilde{D}_j\theta \text{ is } true_{\mathcal{L}} \text{ (resp. } false_{\mathcal{L}}) \text{ in } \Phi_{P'}^m(lfp(\Phi_{P'}))$$

But since $\Phi_{P'}^m(lfp(\Phi_{P'})) = lfp(\Phi_{P'})$, this contradicts the right hand side of (4) . □

From Theorems 3.5 and 3.15 we obtain the following.

Corollary 3.16 (applicability conditions wrt $Comp_{\mathcal{L}} \cup \mathbf{DCA}_{\mathcal{L}}$ with $\mathcal{L}$ finite) Let $\mathcal{L}$ be a finite language. In the hypothesis of Definition 2.11 for simultaneous replacement, if for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, there exists a (possibly empty) set of variables X_i satisfying the locality property wrt $\widetilde{C}_i$ and $\widetilde{D}_i$ such that

$$\exists X_i \widetilde{D}_i \cong_{\mathcal{T}_1^{\mathcal{L}}(P)} \exists X_i \widetilde{C}_i,$$

and one of the following two conditions holds:

1. $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ are all independent from the clauses in $\{cl_1, \ldots, cl_p\}$; or
2. there exists an integer m such that, for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, and each $cl_j \in \{cl_1, \ldots, cl_p\}$:
   - the delay of $\exists X_i \widetilde{D}_i$ wrt $\exists X_i \widetilde{C}_i$ in $lfp(\Phi_P)$ is less or equal to m, and
   - $depen_P(D_i, cl_j) \geq m$;

then P is equivalent to P' wrt $\mathcal{T}_1^{\mathcal{L}}$. □

4 Correctness Results wrt $Comp_{\mathcal{L}}(P) \cup \mathbf{WDCA}_{\mathcal{L}}$ ($\mathcal{L}$ finite)

The aim of this Section is to reformulate the results on the correctness of the replacement operation given for $\mathcal{T}_1^{\mathcal{L}}(P)$ in order to adapt them to $\mathcal{T}_2^{\mathcal{L}}(P)$. We always assume $\mathcal{L}$ to be a finite language.

We are just replacing the $\text{DCA}_{\mathcal{L}}$ closure axioms with $\text{WDCA}_{\mathcal{L}}$. The next example shows how program's equivalence may be affected from such a change.

Example 4.1 Consider the three programs:

$P_1 = \{ \quad n(0).$
$\qquad\quad n(s(X)) \qquad \leftarrow n(X). \ \}$

$P_2 = \{ \quad n(0).$
$\qquad\quad n(s(X)). \ \}$

$P_3 = \{ \quad n(X). \ \}$

Let $\mathcal{L} = \mathcal{L}(P_1)$.

If we assume $\text{DCA}_{\mathcal{L}}$, for all three the programs

$$\mathcal{T}_1^{\mathcal{L}}(P) \models \forall x \ n(x), \quad P \in \{P_1, P_2, P_3\}.$$

Then, all the programs are pairwise *equivalent* wrt this semantics.

If we assume $\mathrm{WDCA}_{\mathcal{L}}$, for P_1

$$\mathcal{T}_2^{\mathcal{L}}(P_1) \not\models \forall x\ n(x),$$

while for $P \in \{P_2, P_3\}$

$$\mathcal{T}_2^{\mathcal{L}}(P) \models \forall x\ n(x), \tag{5}$$

and P_2 and P_3 are *equivalent* wrt this semantics.

Finally if we assume that $\mathcal{L}$ strictly contains $\mathcal{L}(P_1)$, then P_3 is the only program for which (5) holds. In this case no program is equivalent to any of the other ones. □

This example shows that two programs may be equivalent wrt $\mathcal{T}_1^{\mathcal{L}}$ and not equivalent wrt $\mathcal{T}_2^{\mathcal{L}}$. But there are also cases in which the converse of this statement is true. So even though the semantics obtained by assuming $\mathrm{WDCA}_{\mathcal{L}}$ is stronger than the one obtained by assuming $\mathrm{DCA}_{\mathcal{L}}$, no program's equivalence is stronger than the other one.

As before, we characterize equivalence of programs by the Φ_P operator. But in the previous Section this task was quite straightforward through Lemma 3.1. Here, since we are assuming $\mathrm{WDCA}_{\mathcal{L}}$, things are slightly more complicated. We need a lemma first.

Lemma 4.2 Let P be a normal program, $\mathcal{L}$ an arbitrary language, and χ an allowed formula with free variables $\tilde{x}$. For each integer n, there exists two formulas in the language of equality T_χ^n and F_χ^n, with free variables $\tilde{x}$ such that, for any tuple $\tilde{t}$ of ground terms,

- $T_\chi^n(\tilde{t}/\tilde{x})$ is $true_{\mathcal{L}}$ in Φ_P^n iff $\chi(\tilde{t}/\tilde{x})$ is;
  in any other case $T_\chi^n(\tilde{t}/\tilde{x})$ is $false_{\mathcal{L}}$ in Φ_P^n.
- $F_\chi^n(\tilde{t}/\tilde{x})$ is $true_{\mathcal{L}}$ in Φ_P^n iff $\chi(\tilde{t}/\tilde{x})$ is $false_{\mathcal{L}}$ in Φ_P^n.
  in any other case $F_\chi^n(\tilde{t}/\tilde{x})$ is $false_{\mathcal{L}}$ in Φ_P^n.

Proof. From lemma 4.1 in [16] it follows that $T_\chi^n(\tilde{t}/\tilde{x})$ is $true_{\mathcal{L}}$ in Φ_P^n iff $\chi(\tilde{t}/\tilde{x})$ is, and that $F_\chi^n(\tilde{t}/\tilde{x})$ is $true_{\mathcal{L}}$ in Φ_P^n iff $\chi(\tilde{t}/\tilde{x})$ is $false_{\mathcal{L}}$ in Φ_P^n. From the completeness of $\mathrm{CET}_{\mathcal{L}}$ in the case that the underlying universe is the Herbrand Universe, we have that when $T_\chi^n(\tilde{t}/\tilde{x})$ (resp. $F_\chi^n(\tilde{t}/\tilde{x})$) is not $true_{\mathcal{L}}$ in Φ_P^n, it has to be $false_{\mathcal{L}}$ in Φ_P^n. □

To give the intuitive idea of how such formulas are built, let us consider the simple case in which $\chi = n(x)$, and P is the program

$$\begin{array}{lll} P = \{ & n(0). & \\ & n(s(x)) & \leftarrow n(x) \\ & \ldots & \}. \end{array}$$

We have that
$T_n^1(x) \equiv x = 0$,
$T_n^2(x) \equiv x = 0 \vee x = 1$,
...
On the other hand,
$F_n^1(x) \equiv x \neq 0 \wedge \neg\exists y\ x = s(y)$,
$F_n^2(x) \equiv (x \neq 0 \wedge \neg\exists y\ x = s(y)) \vee (\exists y\ x = s(y) \vee (y \neq 0 \wedge \neg\exists z\ y = s(z)))$,
...

Theorem 4.3 Let $\mathcal{L}$ be a finite language, P_1 and P_2 be two normal programs. The following statements are equivalent:

(a) for all ϕ, $\mathcal{T}_2^{\mathcal{L}}(P_1) \models \phi$ implies $\mathcal{T}_2^{\mathcal{L}}(P_2) \models \phi$;

(b) $\forall n\ \exists m\ \Phi_{P_1}^n \subseteq \Phi_{P_2}^m$.

where ϕ ranges over the set of allowed formulas and n and m are quantified over natural numbers.

Proof.
(a) implies (b)
This part is proved by contradiction.
Assume (a) holds and that there exists a fixed n such that

$$\text{for all } m, \quad \Phi^n_{P_1} \not\subseteq \Phi^m_{P_2} \tag{6}$$

For each predicate symbol p let $T^n_{p(\tilde{x})}$ and $F^n_{p(\tilde{x})}$ be the equality formulas described in Lemma 4.2. Hence $T^n_{p(\tilde{x})}(\tilde{t}/\tilde{x})$ is $true_{\mathcal{L}}$ in Φ^n_P iff $p(\tilde{t}/\tilde{x})$ is, and that $F^n_{p(\tilde{x})}(\tilde{t}/\tilde{x})$ is $true_{\mathcal{L}}$ in Φ^n_P iff $p(\tilde{t}/\tilde{x})$ is $false_{\mathcal{L}}$ in Φ^n_P. Let also

$$\chi \equiv \bigwedge_{p \in pred(P_1)} \forall \tilde{x} \ (T^n_{p(\tilde{x})} \rightarrow p(\tilde{x}) \wedge F^n_{p(\tilde{x})} \rightarrow \neg p(\tilde{x}))$$

where p ranges over the finite set of predicate symbols occurring in P_1. From lemma 4.2 it follows that $\Phi^n_{P_1} \models_{\mathcal{L}} \chi$, and, by theorem 2.9

$$\mathcal{T}^{\mathcal{L}}_2(P_1) \models \chi.$$

By (a) we have that $\mathcal{T}^{\mathcal{L}}_2(P_2) \models \chi$, and, by theorem 2.9 there exists an integer r such that

$$\Phi^r_{P_2} \models_{\mathcal{L}} \chi.$$

By (6) $\Phi^n_{P_1} \not\subseteq \Phi^r_{P_2}$, hence there exists a ground atom $q(\tilde{t})$ such that

$$\text{either } \Phi^n_{P_1} \models_{\mathcal{L}} q(\tilde{t}) \text{ and } \Phi^r_{P_2} \not\models_{\mathcal{L}} q(\tilde{t}) \quad \text{or} \quad \Phi^n_{P_1} \models_{\mathcal{L}} \neg q(\tilde{t}) \text{ and } \Phi^r_{P_2} \not\models_{\mathcal{L}} \neg q(\tilde{t}).$$

We consider only the first possibility, the other case is perfectly symmetrical. So we assume that

$$\Phi^n_{P_1} \models_{\mathcal{L}} q(\tilde{t}) \quad \text{and} \quad \Phi^r_{P_2} \not\models_{\mathcal{L}} q(\tilde{t}) \tag{7}$$

By the left hand side of this and the definition of $T^n_{q(\tilde{x})}$ in Lemma 4.2,

$$\Phi^n_{P_1} \models_{\mathcal{L}} T^n_{q(\tilde{x})}(\tilde{t}/\tilde{x})$$

$T^n_{q(\tilde{x})}(\tilde{t}/\tilde{x})$ is a formula of the equality language and contains no predicate symbols other than "=", so if it is $true_{\mathcal{L}}$ in $\Phi^n_{P_1}$ it must be $true_{\mathcal{L}}$ also in $\Phi^0_{P_1}$, i.e. $\Phi^0_{P_1} \models_{\mathcal{L}} T^n_{q(\tilde{x})}(\tilde{t}/\tilde{x})$. But $\Phi^0_{P_1} = (\emptyset, \emptyset) \subseteq \Phi^r_{P_2}$, hence

$$\Phi^r_{P_2} \models_{\mathcal{L}} \ T^n_{q(\tilde{x})}(\tilde{t}/\tilde{x}).$$

Since, $\Phi^r_{P_2} \models_{\mathcal{L}} \chi$, from the definition of χ, follows that also $\Phi^r_{P_2} \models_{\mathcal{L}} \forall \tilde{x} \ (T^n_{q(\tilde{x})}(\tilde{x}) \rightarrow q(\tilde{x}))$, hence that $\Phi^r_{P_2} \models_{\mathcal{L}} \ T^n_{q(\tilde{x})}(\tilde{t}/\tilde{x}) \rightarrow q(\tilde{t})$; and, from the above statement,

$$\Phi^r_{P_2} \models_{\mathcal{L}} \ q(\tilde{t})$$

which contradicts the right hand side of (7).

(b) implies (a)
Let us assume (b), and let ϕ be any allowed formula such that $\mathcal{T}^{\mathcal{L}}_2(P_1) \models \phi$. By theorem 2.9, there exists an integer n such that $\Phi^n_{P_1} \models_{\mathcal{L}} \phi$; by the hypothesis there exists an m such that $\Phi^n_{P_1} \subseteq \Phi^m_{P_2}$, hence $\Phi^m_{P_2} \models_{\mathcal{L}} \phi$.
Again, by theorem 2.9, this implies that $\mathcal{T}^{\mathcal{L}}_2(P_2) \models \phi$. □

4.1 Partial Correctness

As in Proposition 3.2, we can characterize the equivalence of formulas wrt $\mathcal{T}_2^{\mathcal{L}}(P)$ by referring to the Kleene sequence of the operator Φ_P.

Proposition 4.4 Let $\mathcal{L}$ be a finite language, P be a normal program and χ, ϕ be first order allowed formulas. The following statements are equivalent

(a) $\chi \preceq_{\mathcal{T}_2^{\mathcal{L}}(P)} \phi$;

(b) $\forall n\ \exists m\ \forall \tilde{t} \quad \Phi_P^n \models_{\mathcal{L}} (\neg)\chi(\tilde{t}/\tilde{x}) \quad implies \quad \Phi_P^m \models_{\mathcal{L}} (\neg)\phi(\tilde{t}/\tilde{x})$;

where n, m are quantified over natural numbers, $\tilde{x} = \{x_1, \ldots, x_k\} = FV(\chi) \cup FV(\phi)$, and $\tilde{t}$ is quantified over k-tuples of $\mathcal{L}$-terms.

Proof.

(a) implies (b)

This part is by contradiction. Let us assume there exists a *fixed* n, such that for each integer m there exists a k-uple of $\mathcal{L}$-terms $\tilde{t}_m$ for which the following hold

(i) $\Phi_P^n \models_{\mathcal{L}} (\neg)\chi(\tilde{t}_m/\tilde{x})$;

(ii) $\Phi_P^m \not\models_{\mathcal{L}} (\neg)\phi(\tilde{t}_m/\tilde{x})$.

By Lemma 4.2 there exist two formulas T_χ^n and F_χ^n in the language of equality, such that $FV(T_\chi^n) = FV(F_\chi^n) = FV(\chi)$ and

$$\Phi_P^n \models_{\mathcal{L}} \forall\tilde{x}\ (T_\chi^n \rightarrow \chi\ \wedge\ F_\chi^n \rightarrow \neg\chi).$$

By Theorem 2.9

$$\mathcal{T}_2^{\mathcal{L}}(P)\ \models \forall\tilde{x}\ (T_\chi^n \rightarrow \chi\ \wedge\ F_\chi^n \rightarrow \neg\chi).$$

By (a),

$$\mathcal{T}_2^{\mathcal{L}}(P)\ \models \forall\tilde{x}\ (T_\chi^n \rightarrow \phi\ \wedge\ F_\chi^n \rightarrow \neg\phi).$$

This is an allowed formula, then by Theorem 2.9 there exists an r such that

$$\Phi_P^r \models_{\mathcal{L}} \forall\tilde{x}\ (T_\chi^n \rightarrow \phi\ \wedge\ F_\chi^n \rightarrow \neg\phi). \tag{8}$$

But by (i) $\chi(\tilde{t}_r/\tilde{x})$ is either $true_{\mathcal{L}}$ or $false_{\mathcal{L}}$ in Φ_P^n, let us now consider just the first possibility, that is

$$\Phi_P^n \models_{\mathcal{L}} \chi(\tilde{t}_r/\tilde{x}).$$

The other case is perfectly symmetrical and omitted here.

From this and the definition of T_χ^n in Lemma 4.2, we have that $\Phi_P^n \models_{\mathcal{L}} T_\chi^n(\tilde{t}_r/\tilde{x})$, and since $T_\chi^n(\tilde{t}_r)$ is a formula in the language of equality, if it is $true_{\mathcal{L}}$ in Φ_P^n it must be $true_{\mathcal{L}}$ already at stage 0, that is $\Phi_P^0 \models_{\mathcal{L}} T_\chi^n(\tilde{t}_r/\tilde{x})$, but $\Phi_P^0 \subseteq \Phi_P^r$, hence

$$\Phi_P^r \models_{\mathcal{L}} T_\chi^n(\tilde{t}_r/\tilde{x}).$$

But then, by (8), $\Phi_P^r \models_{\mathcal{L}} \phi(\tilde{t}_r/\tilde{x})$, contradicting (ii).

(b) implies (a)

We prove that for each n there exists an m such that for any closed allowed formula ζ, and for any ground substitution σ,

$$\Phi_P^n \models_{\mathcal{L}} \zeta \quad \text{implies} \quad \Phi_P^m \models_{\mathcal{L}} \zeta[\phi\sigma/\chi\sigma]. \tag{9}$$

By Theorem 2.9 this implies (a).

Let m be an integer that satisfies hypothesis (b) for some n. It is not restrictive to assume that $m \geq n$. Let ζ be a closed allowed formula such that

$$\Phi_P^n \models_{\mathcal{L}} \zeta.$$

If ζ does not contain any instance of $\chi\sigma$ as a subformula then (9) follows immediately from the assumption that $m \geq n$. In the case that ζ contains $\chi\sigma$ as a subformula we proceed by induction on the structure of ζ.

Base step: $\zeta = \chi\sigma$, then (9) follows immediately from (b).

Induction step: we consider three cases:

1) If $\zeta = \triangle\ \zeta_1$, where $\triangle$ is any allowed unary connective, or $\zeta = \zeta_1 \diamond \zeta_2$, where $\diamond$ is any allowed binary connective, then we have that either ζ_i does not contain $\chi\sigma$ as a subformula (and the result holds trivially) or the inductive hypothesis applies.

2) If $\zeta = \forall w\ \zeta_1(w)$.

Since $\Phi_P^n \models_{\mathcal{L}} \zeta$, we have that

$$\text{for each } \mathcal{L}\text{-term } t, \Phi_P^n \models_{\mathcal{L}} \zeta_1(t).$$

For each $\mathcal{L}$-term t, let γ_t be the substitution (t/w), by the inductive hypothesis there exists an m such that

$$\text{for each } \mathcal{L}\text{-term } t, \Phi_P^m \models_{\mathcal{L}} \zeta_1(t)[\phi\sigma\gamma_t/\chi\sigma\gamma_t].$$

Since the underlying universe of Φ_P^m is the Herbrand universe on $\mathcal{L}$, this implies that

$$\Phi_P^m \models_{\mathcal{L}} \forall w\ \zeta_1(w)[\phi\sigma/\chi\sigma].$$

3) Finally, the case $\zeta = \exists w\ \zeta_1(w)$, is treated as $\neg\forall w\ \neg\zeta_1(w)$. □

In the above Proposition, statement (b) differs from the corresponding one of Proposition 3.2. Let us consider the two statements:

(a) $\chi \cong_{\mathcal{T}_2^{\mathcal{L}}(P)} \phi$.

(b) for each tuple of $\mathcal{L}$-terms $\tilde{t}$, $\mathcal{T}_2^{\mathcal{L}}(P_1) \models \chi(\tilde{t}/\tilde{x})$ iff $\mathcal{T}_2^{\mathcal{L}}(P_1) \models \phi(\tilde{t}/\tilde{x})$.

(a) implies (b), but not vice-versa. On the other hand, if we use $\mathcal{T}_1^{\mathcal{L}}$ instead of $\mathcal{T}_2^{\mathcal{L}}$, we have that the two statements are equivalent, and this is just a reformulation of Proposition 3.2. When we are assuming $\text{WDCA}_{\mathcal{L}}$, the universe of a model of $\mathcal{T}_2^{\mathcal{L}}(P_1)$ may contain non-standard elements, that is, elements which are not $\mathcal{L}$-terms. Hence the equivalence between all the closed instances of χ and ϕ is no longer sufficient to ensure the equivalence between χ and ϕ.

For example, if we consider the following program:

$$\begin{array}{lll} P = \{ & n(0). & \\ & n(s(X)) \leftarrow n(X). & \\ & m(X). & \} \end{array}$$

and we fix $\mathcal{L} = \mathcal{L}(P)$, we have that for each $\mathcal{L}$-term t, both $n(t)$ and $m(t)$ are *true* in all models of $\mathcal{T}_2^{\mathcal{L}}(P)$, but $n(X) \cong_{\mathcal{T}_2^{\mathcal{L}}(P)} m(X)$. In fact, let $\zeta \equiv \forall x\ m(x)$, then $\mathcal{T}_2^{\mathcal{L}}(P) \models \zeta$, while $\mathcal{T}_2^{\mathcal{L}}(P) \not\models \zeta[n(x)/m(x)]$ (see Example 4.1).

The equivalence defined as follows: "χ is equivalent to ϕ iff (b) holds" is too weak for our purposes. In fact if we consider the following extension to program P:

$$\begin{array}{lll} P_1 = P \cup \{ & q_1 \leftarrow & \neg n(X). \\ & q_2 \leftarrow & \neg m(X). \quad \} \end{array}$$

and $\mathcal{L} = \mathcal{L}(P_1)$, $n(X)$ is equivalent to $m(X)$ while q_1 is not equivalent to q_2 and it would be impossible to obtain an applicability condition similar to 3.16.

Now, given a characterization of equivalent formulas, we can state the result on partial correctness of the replacement operation wrt the $\mathcal{T}_2^{\mathcal{L}}(P)$ semantics.

Theorem 4.5 (partial correctness) Let $\mathcal{L}$ be a finite language. In the hypothesis of Definition 2.11 for simultaneous replacement, if for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, there exists a (possibly empty) set of variables X_i satisfying the locality property wrt $\widetilde{C}_i$ and $\widetilde{D}_i$ such that

$$\exists X_i \widetilde{D}_i \preceq_{\mathcal{T}_2^{\mathcal{L}}(P)} \exists X_i \widetilde{C}_i,$$

then $\forall n \; \exists m \quad \Phi_P^m \supseteq \Phi_{P'}^n$.

Proof. The proof is by contradiction. Let us suppose there exist two integers i and j such that:

$$\Phi_P^i \supseteq \Phi_{P'}^j \quad \text{and} \quad \text{for all integers } l, \; \Phi_P^l \not\supseteq \Phi_{P'}^{j+1}.$$

Clearly it also follows that

$$\text{for all integers } l, \; \Phi_P^{l+i+1} \not\supseteq \Phi_{P'}^{j+1}.$$

Since $\Phi_{P'}^{j+1} = \Phi_{P'}(\Phi_{P'}^j)$, $\Phi_P^i \supseteq \Phi_{P'}^j$ and $\Phi_{P'}$ is monotone, we have that $\Phi_{P'}(\Phi_P^i) \supseteq \Phi_{P'}^{j+1}$, hence

$$\text{for all integers } l, \; \Phi_P(\Phi_P^{l+i}) \not\supseteq \Phi_{P'}(\Phi_P^i).$$

Since $\Phi_P^{l+i} \supseteq \Phi_P^i$, from Claim 1 in the proof of Theorem 3.5, it follows that for each integer l there exist an integer $j(l) \in \{1, \ldots, n\}$ and a ground substitution θ_l such that:

$$\exists X_{j(l)} \widetilde{D}_{j(l)} \theta_l \text{ is } true_{\mathcal{L}} \text{ (or } false_{\mathcal{L}}) \text{ in } \Phi_P^i, \text{ while } \exists X_{j(l)} \widetilde{C}_{j(l)} \theta_l \text{ it is not } true_{\mathcal{L}} \text{ (resp. } false_{\mathcal{L}}) \text{ in } \Phi_P^{l+i}. \quad (10)$$

By hypothesis $\exists X_i \widetilde{D}_i \preceq_{\mathcal{T}_2^{\mathcal{L}}(P)} \exists X_i \widetilde{C}_i$, then we can apply Proposition 4.4 to the left hand side of (10) to obtain that for each l, there has to be an integer r such that

$$\exists X_{j(l)} \widetilde{C}_{j(l)} \theta_l \text{ is } true_{\mathcal{L}} \text{ (resp } false_{\mathcal{L}}) \text{ in } \Phi_P^r;$$

but when l satisfies $l + i > r$, we have that $\Phi_P^{l+i} \supseteq \Phi_P^r$ and hence

$$\text{for each } l \text{ such that } l + i > r, \quad \exists X_{j(l)} \widetilde{C}_{j(l)} \theta_l \text{ is } true_{\mathcal{L}} \text{ (resp } false_{\mathcal{L}}) \text{ in } \Phi_P^{l+i}.$$

This contradicts (10). □

4.2 Completeness

In order to state the completeness result, we can use a definition of semantic delay slightly weaker than the one given for $\mathcal{T}_1^{\mathcal{L}}(P)$.

Definition 4.6 (semantic delay in Φ_P^ω) Let P be a normal program, χ and ϕ be first order formulas, and $\tilde{x} = \{x_1, \ldots, x_k\} = FV(\chi) \cup FV(\phi)$. Suppose that $\phi \preceq_{\mathcal{T}_2^{\mathcal{L}}(P)} \chi$.

- *The semantic delay of χ wrt ϕ in Φ_P^ω* is the least integer k such that, for each integer n and each k-uple of $\mathcal{L}$-terms $\tilde{t}$: if $\Phi_P^n \models_{\mathcal{L}} (\neg)\phi(\tilde{t}/\tilde{x})$, then $\Phi_P^{n+k} \models_{\mathcal{L}} (\neg)\chi(\tilde{t}/\tilde{x})$. □

Theorem 4.7 (completeness) In the hypothesis of Definition 2.11 for simultaneous replacement, if for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, there exists a (possibly empty) set of variables X_i satisfying the locality property wrt $\widetilde{C}_i$ and $\widetilde{D}_i$ such that

$$\exists X_i \widetilde{C}_i \preceq_{\mathcal{T}_2^{\mathcal{L}}(P)} \exists X_j \widetilde{D}_j,$$

and if one of the following two conditions holds:

(a) $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ are all independent from the clauses $\{cl_1, \ldots, cl_p\}$; or

(b) there exists an integer m such that, for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, and each $cl_j \in \{cl_1, \ldots, cl_p\}$:
- the delay of $\exists X_i \widetilde{D}_i$ wrt $\exists X_i \widetilde{C}_i$ in Φ_P^ω is less or equal to m, and
- $depen_P(\widetilde{D}_i, cl_j) \geq m$;

then $\forall n\ \exists\ m\ \ \Phi_P^n \subseteq \Phi_{P'}^m$.

Proof. Again the proof is by contradiction. Let us suppose that there exist two integers i and j such that:

$$\Phi_{P'}^i \supseteq \Phi_P^j \ \text{ and } \ \text{for all integers } l, \Phi_{P'}^{i+l+1} \not\supseteq \Phi_P^{j+1}.$$

Since Φ_P^{j+1}=$\Phi_P(\Phi_P^j)$, from Claim 2 in the proof of Theorem 3.15 we have that:
for each integer l there exists an integer $j(l) \in \{1, \ldots, n\}$ and a ground substitution θ_l such that:

$$\exists X_{j(l)}\widetilde{C}_{j(l)}\theta_l \text{ is } true_{\mathcal{L}} \text{ (or } false_{\mathcal{L}}) \text{ in } \Phi_P^j, \text{ while } \exists X_{j(l)}\widetilde{D}_{j(l)}\theta_l \text{ is not } true_{\mathcal{L}} \text{ (resp. not } false_{\mathcal{L}}) \text{ in } \Phi_{P'}^{i+l}. \tag{11}$$

Let us distinguish two cases.

1) Hypothesis (a) is satisfied and each conjunction in $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ is independent from $\{cl_1, \ldots, cl_p\}$. From the left hand side of (11), the hypothesis and Proposition 4.4 it follows that for each l there has to be an integer r such that

$$\exists X_{j(l)}\widetilde{D}_{j(l)}\theta_l \text{ is } true_{\mathcal{L}} \text{ (resp. } false_{\mathcal{L}}) \text{ in } \Phi_P^r.$$

From remark 3.12, it follows that for each integer l, $\exists X_{j(l)}\widetilde{D}_{j(l)}\theta_l$ is $true_{\mathcal{L}}$ (resp. $false_{\mathcal{L}}$) in $\Phi_{P'}^r$.
This contradicts (11); in fact, when $i + l > r$, by the monotonicity of $\Phi_{P'}$, we have that $\Phi_{P'}^r \subseteq \Phi_{P'}^{i+l}$ and since $\exists X_{j(l)}\widetilde{D}_{j(l)}\theta_l$ is $true_{\mathcal{L}}$ (resp. $false_{\mathcal{L}}$) in $\Phi_{P'}^r$, it must be $true_{\mathcal{L}}$ (resp. $false_{\mathcal{L}}$) in $\Phi_{P'}^{i+l}$.

2) Hypothesis (b) is satisfied. We know that for each integer l, the delay of $\exists X_{j(l)}\widetilde{D}_{j(l)}$ wrt $\exists X_{j(l)}\widetilde{C}_{j(l)}$ is not greater than m, hence from the left hand side of (11) it follows that,

$$\text{for each } l, \ \ \exists X_{j(l)}\widetilde{D}_{j(l)}\theta_l \text{ is } true_{\mathcal{L}} \text{ or } false_{\mathcal{L}} \text{ in } \Phi_P^{j+m}.$$

Since $\Phi_P^{j+m} = \Phi_P^m(\Phi_P^j)$, it follows that,

$$\text{for each } l, \ \ \exists X_{j(l)}\widetilde{D}_{j(l)}\theta_l \text{ is } true_{\mathcal{L}} \text{ (resp. } false_{\mathcal{L}}) \text{ in } \Phi_P^m(\Phi_P^j).$$

$depen_P(\widetilde{D}_{j(l)}\theta_l, \{cl_1, \ldots, cl_p\}) \geq m$, then, from Lemma 3.14 it follows that,

$$\text{for each } l, \ \ \exists X_{j(l)}\widetilde{D}_{j(l)}\theta_l \text{ is } true_{\mathcal{L}} \text{ (resp. } false_{\mathcal{L}}) \text{ in } \Phi_{P'}^m(\Phi_P^j).$$

Now $\Phi_P^j \subseteq \Phi_{P'}^i$ and $\Phi_{P'}$ is monotone, then,

$$\text{for each } l, \ \ \exists X_{j(l)}\widetilde{D}_{j(l)}\theta_l \text{ is } true_{\mathcal{L}} \text{ (resp. } false_{\mathcal{L}}) \text{ in } \Phi_{P'}^m(\Phi_{P'}^i) = \Phi_{P'}^{m+i},$$

this contradicts the right hand side of (11). □

From Theorems 4.5 and 4.7 we obtain the following.

Corollary 4.8 (applicability conditions wrt $Comp_{\mathcal{L}} \cup \mathbf{WDCA}_{\mathcal{L}}$ with $\mathcal{L}$ finite) Let $\mathcal{L}$ be a finite language. In the hypothesis of Definition 2.11 for simultaneous replacement, if for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, there exists a (possibly empty) set of variables X_i satisfying the locality property wrt $\widetilde{C}_i$ and $\widetilde{D}_i$ such that

$$\exists X_i\widetilde{D}_i \ \cong_{\mathcal{T}_2^{\mathcal{L}}(P)} \ \exists X_i\widetilde{C}_i,$$

and one of the following two conditions holds:

1. $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ are all independent from the clauses in $\{cl_1, \ldots, cl_p\}$; or
2. there exists an integer m such that, for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, and each $cl_j \in \{cl_1, \ldots, cl_p\}$:
   - the delay of $\exists X_i\widetilde{D}_i$ wrt $\exists X_i\widetilde{C}_i$ in $lfp(\Phi_P)$ is less or equal to m, and
   - $depen_P(D_i, cl_j) \geq m$;

then P is equivalent to P' wrt $\mathcal{T}_2^{\mathcal{L}}$. □

5 Correctness Results when $\mathcal{L}$ is Infinite

5.1 Correctness Results when $\mathcal{L}$ is Infinite

When the language is infinite, that is when it contains infinitely many function symbols, the domain closure axioms are no longer needed since in this case $\text{CET}_{\mathcal{L}}$ is already a complete theory.

Three valued program's completion semantics in the case of an infinite language has been studied by Kunen [11] and successively by Shepherdson [16]. As far as we are concerned the $Comp_{\mathcal{L}}(P)$ semantics when $\mathcal{L}$ is infinite behaves exactly as the $Comp_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$ when $\mathcal{L}$ is finite. This fact is due to the following result:

Theorem 5.1 Let P be a normal program, $\mathcal{L}$ an infinite language and ϕ an allowed formula.

- $Comp_{\mathcal{L}}(P) \models \phi$ iff for some integer n, $\Phi_P^n \models_{\mathcal{L}} \phi$

Proof. This is Theorem 5b in [16]. □

Observe that this is identical to Theorem 2.9 (b), which was the only result on the semantics that we used in Section 4. Consequently, the results that we can prove on program's and formula's equivalence and on the replacement operation are identical to the ones given in the previous Section. Theorem 4.3 holds also for $Comp(P)$ with $\mathcal{L}$ infinite; the proof is identical as well, as Lemma 4.2 holds for an arbitrary language. The same reasoning applies to Proposition 4.4 on the equivalence of formulas. Finally, the results on the replacement operation, that is Theorems 4.5, 4.7 and Corollary 4.8 hold also for this semantics.

Corollary 5.2 (applicability conditions wrt $Comp_{\mathcal{L}}$ with $\mathcal{L}$ infinite) Let $\mathcal{L}$ be an infinite language. In the hypothesis of Definition 2.11 for simultaneous replacement, if for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, there exists a (possibly empty) set of variables X_i satisfying the locality property wrt $\widetilde{C}_i$ and $\widetilde{D}_i$ such that

$$\exists X_i \widetilde{D}_i \text{ is equivalent to } \exists X_i \widetilde{C}_i \text{ wrt } Comp_{\mathcal{L}}(P),$$

and one of the following two conditions holds:

1. $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ are all independent from the clauses in $\{cl_1, \ldots, cl_p\}$; or
2. there exists an integer m such that, for each $\widetilde{C}_i \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$, and each $cl_j \in \{cl_1, \ldots, cl_p\}$:
   - the delay of $\exists X_i \widetilde{D}_i$ wrt $\exists X_i \widetilde{C}_i$ in $lfp(\Phi_P)$ is less or equal to m, and
   - $depen_P(D_i, cl_j) \geq m$;

then P is equivalent to P' wrt the three valued completion semantics, $Comp_{\mathcal{L}}$. □

6 Replacement vs. Other Operations.

We now consider some other operations which are normally used in program's transformation and show how they can be seen as particular cases of replacement. This will also give us the opportunity of providing some other examples.

6.1 Reversible Folding

The *fold* operation consists in substituting an atom for an equivalent conjunction of literals, in the body of a clause. This operation is generally used in all the transformation systems in order to pack back unfolded clauses and to detect implicit recursive definitions. In the literature we find different definitions for this operation. This is due to the fact that it is not generally safe even for definite programs and declarative semantics and its application must be restricted by some conditions which depend on the semantics we choose. The *reversible* folding corresponds to the kind of folding considered in [13, 9].

Definition 6.1 (reversible fold) Let $cl : A \leftarrow \widetilde{L}, \widetilde{H}'.$ and $d : B \leftarrow \widetilde{H}.$ be distinct clauses in a program P; let also W be the set of local variables of $\widetilde{H}$ in d, $W = Var(\widetilde{H}) \backslash Var(B)$.

If there exists a substitution θ, $dom(\theta) = Var(\widetilde{H}) \backslash W$, such that $\widetilde{H}' = \widetilde{H}\theta$ and d is the only clause of P whose head unifies with $B\theta$. Then

- *Folding* $\widetilde{H}'$ *in* cl by using d as folding clause consists of substituting cl' for cl, where $cl' : A \leftarrow \widetilde{L}, B\theta.$

$fold(P, \widetilde{H}, cl, d) = P \backslash \{cl\} \cup \{cl' : A \leftarrow \widetilde{L}, B\theta.\}$ □

Example 6.2 Let us consider the following program:

$$P = \{ \begin{array}{lll} cl: & p(X) & \leftarrow q(X,b), \neg s(X), r(a,X). \\ d: & r(Z,Y) & \leftarrow q(Y,Z), \neg s(Y). \\ & r(a,Y) & \leftarrow p(Y). \\ & q(X,a). & \\ & q(X,b). & \end{array} \}$$

With $\theta = \{b/Z, X/Y\}$, we have $body(d)\theta = (q(X,b), \neg s(X))$ and that d is the only clause of P whose head unifies with $r(Z,Y)\theta$. Hence we can fold clause cl, thus obtaining the program:

$$P = \{ \begin{array}{lll} cl: & p(X) & \leftarrow r(b,X), r(a,X). \\ d: & r(Z,Y) & \leftarrow q(Y,Z), \neg s(Y). \\ & r(a,Y) & \leftarrow p(Y). \\ & q(X,a). & \\ & q(X,b). & \end{array} \}$$

□

This operation is always safe. Indeed we now show that it can be seen as a special case of replacement in which the conditions of Corollaries 3.16 and 4.8 are always satisfied.

Theorem 6.3 (correctness of reversible folding) The reversible folding operation is safe wrt both $\mathcal{T}_1^{\mathcal{L}}$ and $\mathcal{T}_2^{\mathcal{L}}$.

Proof. Consider the notation as in Definition 6.1. Recall that d is the only clause that unifies with $B\theta$. By the definition of Fitting's operator we have that, for all substitutions γ, $B\theta\gamma$ is *true* (resp. *false*) at step Φ_P^α iff α is an ordinal greater than 1 and $\exists W \widetilde{H}\theta\gamma$ is $true_{\mathcal{L}}$ (resp. $false_{\mathcal{L}}$) at step $\Phi_P^{\alpha-1}$. This implies that whichever semantics we consider:
- $B\theta$ is *equivalent* to $\exists W \widetilde{H}\theta$, and
- the delay of $B\theta$ wrt $\exists W \widetilde{H}\theta$ is one.

Since $d \neq cl$, we also have that $depen_P(B\theta, cl) > 0$. Hence, by Corollary 3.16 (resp. 4.8), the operation is $\mathcal{T}_1^{\mathcal{L}}$-safe (resp. $\mathcal{T}_2^{\mathcal{L}}$-safe). □

Now we need to define the unfold operation which is widely used in transformations. We suppose that all the clauses are disjoint, that is, they have no variable in common.

Definition 6.4 (unfold) Let $cl : A \leftarrow \widetilde{L}, H.$ be a clause of a normal program P, where H is an atom. Let $\{H_1 \leftarrow \widetilde{B}_1, \ldots, H_n \leftarrow \widetilde{B}_n\}$ be the set of clauses of P whose heads unify with H, by mgu's $\{\theta_1, \ldots, \theta_n\}$.

- *Unfolding an atom* H *in* cl consists of substituting cl with $\{cl'_1, \ldots, cl'_n\}$, where, for each i, $cl'_i = (A \leftarrow \widetilde{L}, \widetilde{B}_i)\theta_i$.

$unfold\ (P, cl, H) \stackrel{\text{def}}{=} P \backslash \{cl\} \cup \{cl'_1, \ldots, cl'_n\}$. □

This operation is safe wrt all the semantics we consider in this paper[1].

[1] The proof of safeness will appear in a technical report and is now reported in the Appendix B.

Example 6.5 (sorting by permutation and check, part I) The following program is borrowed from [17]. The transformation process is intentionally redundant in order to be more explanatory. For the sake of simplicity, here we consider the semantics given by $\mathcal{T}_1^{\mathcal{L}}$. The results hold also in the case we adopt $\mathcal{T}_2^{\mathcal{L}}$ although, as we will point out, the proofs are more complicated.

Let P_0 be the following program:

$$
\begin{array}{lll}
P_0 = \{ & c1: \ perm([\,],[\,]). & \\
& c2: \ perm([A \mid Xs], Ys) & \leftarrow \quad perm(Xs, Zs), ins(A, Zs, Ys). \\
& c3: \ ins(A, Xs, [A \mid Xs]). & \\
& c4: \ ins(A, [B \mid Xs], [B \mid Ys]) & \leftarrow \quad ins(A, Xs, Ys). \\
& c5: \ ord([\,]). & \\
& c6: \ ord([A]). & \\
& c7: \ ord([A, B \mid Xs]) & \leftarrow \quad A \leq B, ord([B \mid Xs]). \\
& c8: \ sort(Xs, Ys) & \leftarrow \quad perm(Xs, Ys), ord(Ys). \\
& \ldots & \qquad\qquad \}
\end{array}
$$

(Step 1) If we unfold $perm(Xs, Ys)$ in the body of $c8$; the resulting program is:
$P_1 = \{c1, \ldots, c7\} \cup$

$$
\begin{array}{lll}
\{ & c9: \ sort([\,],[\,]) & \leftarrow \quad ord([\,]). \\
& c10: \ sort([A \mid Xs], Ys) & \leftarrow \quad perm(Xs, Zs), ins(A, Zs, Ys), ord(Ys).\}
\end{array}
$$

(Step 2) By unfolding $ord([\,])$ in $c9$, we eliminate $ord([\,])$ from the body of that clause.
$P_2 = \{c1, \ldots, c7\} \cup \{c10\} \cup \{\ c11: \ sort([\,],[\,]).\}$
By the safeness of the unfold operation (Corollary 9.2 in the Appendix B) P_0, P_1 and P_2 are equivalent programs both wrt $\mathcal{T}_1^{\mathcal{L}}$ and $\mathcal{T}_2^{\mathcal{L}}$. □

6.2 Thining and Fattening

The *fatten* operation consists in introducing redundant literals in the body of a clause. It is generally used in order to make possible some other transformations such as folding.

Definition 6.6 (fatten) Let $cl : A \leftarrow \widetilde{L}.$ be a clause in a program P and $\widetilde{H}$ a conjunction of literals.

- *Fattening cl with* $\widetilde{H}$ *in* P consists of substituting cl' for cl, where $cl' : A \leftarrow \widetilde{L}, \widetilde{H}.$

fatten $(P, c, \widetilde{H}) \stackrel{\text{def}}{=} P \backslash \{cl\} \cup \{cl'\}$. □

The fatten operation is a special case of replacement, and then its applicability conditions can be drawn directly from Corollaries 3.16 and 4.8.

The next Lemma shows that for fattening, we actually need to check only part of the applicability conditions.

Lemma 6.7 Let $cl = A \leftarrow \widetilde{E}, \widetilde{G}.$ be a clause in the normal program P, X be a set of variables not occurring in $(A, \widetilde{E})$ and $\widetilde{H}$ be another conjunction of literals. Then

(a) If for each θ, $lfp(\Phi_P) \models_{\mathcal{L}} \exists X\widetilde{G}\theta$ implies $lfp(\Phi_P) \models_{\mathcal{L}} (\exists X\widetilde{G}, \widetilde{H})\theta$,
then $\exists X\widetilde{G} \preceq_{\mathcal{T}_1^{\mathcal{L}}(P)} \exists X\widetilde{G}, \widetilde{H}$.

(b) If for each θ, $lfp(\Phi_P) \models_{\mathcal{L}} \neg(\exists X\widetilde{G}, \widetilde{H})\theta$ implies $lfp(\Phi_P) \models_{\mathcal{L}} \neg\exists X\widetilde{G}\theta$
then $\exists X\widetilde{G}, \widetilde{H} \preceq_{\mathcal{T}_1^{\mathcal{L}}(P)} \exists X\widetilde{G}$.

(c) If m is an integer such that, for each α and θ, $\Phi_P^{\alpha} \models_{\mathcal{L}} \exists X\widetilde{G}\theta$ implies $\Phi_P^{\alpha+m} \models_{\mathcal{L}} (\exists X\widetilde{G}, \widetilde{H})\theta$, then
- $\exists X\widetilde{G} \preceq_{\mathcal{T}_1^{\mathcal{L}}(P)} \exists X\widetilde{G}, \widetilde{H}$,

- the delay of $\exists X\widetilde{G}, \widetilde{H}$ wrt $\exists X\widetilde{G}$ in $lfp(\Phi_P)$ is less or equal to m.

If m is the least of such integers, then the delay of $\exists X\widetilde{G}, \widetilde{H}$ wrt $\exists X\widetilde{G}$ in $lfp(\Phi_P)$ is exactly m.

Proof. It is a straightforward application of Theorem 2.9 together with the fact that, for any interprtation I, $I \models_{\mathcal{L}} \neg\widetilde{G}\theta$ implies that $I \models_{\mathcal{L}} \neg(\widetilde{G}, \widetilde{H})\theta$. □

This Lemma applies as well to the semantics given by $\mathcal{T}_2^{\mathcal{L}}$, as it is shown by Lemma 8.2 in the Appendix A.

Example 6.5 (sorting by permutation and check, part II)

(Step 3) Now we can fatten clause $c10$ by adding $ord(Zs)$ to its body.
Let P_3 be the resulting program:
$P_3 = \{c1, \ldots, c7\} \cup$

$$\{ \quad c11: \quad sort([\,],[\,]). \\ \quad\; c12: \quad sort([A \mid Xs], Ys) \quad \leftarrow \quad perm(Xs, Zs), ord(Zs), ins(A, Zs, Ys), ord(Ys).\}$$

This operation corresponds to a replacement of $ins(A, Zs, Ys), ord(Ys)$ with $ord(Zs)$, $ins(A, Zs, Ys)$, $ord(Ys)$.

We now use Theorem 3.15 to prove that $lfp(\Phi_{P_2}) \subseteq lfp(\Phi_{P_3})$.
Observe that if $(ins(A, Zs, Ys), ord(Ys))\theta$ is $true$ in $lfp(\Phi_{P_2})$ then $Ys\theta$ is an ordered list and $Zs\theta$ is a sublist of $Ys\theta$; hence also $Zs\theta$ is ordered and $(ord(Zs), ins(A, Zs, Ys), ord(Ys))\theta$ is also $true$ in $lfp(\Phi_{P_2})$. By Lemma 6.7, this is sufficient to state that:

$$ins(A, Zs, Ys), ord(Ys) \preceq_{\mathcal{T}_1^{\mathcal{L}}(P_2)} ord(Zs), ins(A, Zs, Ys), ord(Ys)^2.$$

Notice also that the conjunction $ord(Zs), ins(A, Zs, Ys), ord(Ys)$ is independent from clause $c10$. Hence by Theorem 3.15 $lfp(\Phi_{P_2}) \subseteq lfp(\Phi_{P_3})$, by Lemma 3.1 this means that the operation is $\mathcal{T}_1^{\mathcal{L}}$-complete.

To show that the operation is also safe, that is, that $lfp(\Phi_{P_2}) = lfp(\Phi_{P_3})$, we could use Corollary 3.16, but it is easier to observe that $lfp(\Phi_{P_2})$ is already a *total* model [3], namely nothing is undefined, hence $lfp(\Phi_{P_2}) \subseteq lfp(\Phi_{P_3})$ implies that $lfp(\Phi_{P_2}) = lfp(\Phi_{P_3})$, and by (2.9) the operation is also safe.

(Step 4) We can now fatten $c12$ with $sort(Xs, Zs)$. The resulting program is:
$P_4 = \{c1, \ldots, c7\} \cup$

$$\{ \quad c11: \quad sort([\,],[\,]). \\ \quad\; c13: \quad sort([A \mid Xs], Ys) \quad \leftarrow \quad sort(Xs, Zs), perm(Xs, Zs), ord(Zs), ins(A, Zs, Ys), ord(Ys).\}$$

This operation corresponds to a replacement of $perm(Xs, Zs), ord(Zs)$ with $sort(Xs, Zs)$, $perm(Xs, Zs)$, $ord(Zs)$. Using Corollary 3.16 we can prove that $lfp(\Phi_{P_3}) = lfp(\Phi_{P_4})$; in order to apply the Corollary we have to show that:

(a) $sort(Xs, Zs), perm(Xs, Zs), ord(Zs) \cong_{\mathcal{T}_1^{\mathcal{L}}(P_3)} perm(Xs, Zs), ord(Zs)$;

(b) the delay of $sort(Xs, Zs), perm(Xs, Zs), ord(Zs)$ wrt $perm(Xs, Zs), ord(Zs)$ in $lfp(\Phi_{P_3})$ is zero.

To prove (a) we proceed as follows: since $sort(Xs, Zs) \leftarrow perm(Xs, Zs), ord(Zs)$, is a clause of P_0, by Lemma 3.9, we have that $sort(Xs, Zs) \cong_{\mathcal{T}_1^{\mathcal{L}}(P_0)} perm(Xs, Zs), ord(Zs)$, this clearly implies that

[2] When using WDCA instead of DCA, in order to establish the equivalence, computations are in general more complicated. In this example it is sufficient to observe that $(ins(A, Zs, Ys), ord(Ys))\theta$ is $true_{\mathcal{L}}$ in $\Phi^n_{P_2}$ then also $ord(Zs)\theta$ is $true_{\mathcal{L}}$ in $\Phi^n_{P_2}$.

[3] This also follows from a result due to Apt and Bezem [2], that states that the Fitting's Model of an acyclic program is always a total model.

$sort(Xs,Zs), perm(Xs,Zs), ord(Zs) \cong_{\mathcal{T}_1^{\mathcal{L}}(P_0)} perm(Xs,Zs), ord(Zs)$. By Proposition 3.2 this implies that $lfp(P_0) \models sort(Xs,Zs), perm(Xs,Zs), ord(Zs) \Leftrightarrow perm(Xs,Zs), ord(Zs)$. From the correctness of the previous transformation's steps we have that $lfp(P_3) = lfp(P_0)$; hence

$lfp(P_3) \models sort(Xs,Zs),\ perm(Xs,Zs),\ ord(Zs) \Leftrightarrow perm(Xs,Zs), ord(Zs)$,

and (a) follows from Proposition 3.2.

We now prove (b). Note that it is sufficient to show that the delay of $sort(Xs,Zs)$ wrt $perm(Xs,Zs)$, $ord(Zs)$ in $lfp(\Phi_{P_3})$ is zero. By Lemma 6.7, it is sufficient to prove that:

for all θ, k, if $\Phi^k_{P_3} \models_{\mathcal{L}} (perm(Xs,Zs), ord(Zs))\theta$ then also $\Phi^k_{P_3} \models_{\mathcal{L}} sort(Xs,Zs)\theta$.

First we need to prove a few properties. In the following we denote $|l|$ the length of a list l.

(i) $ins(A,Zs,Ys)\theta$ becomes *true* at step $\Phi^n_{P_3}$, where $n \leq |Ys\theta|$, in fact n is precisely the place where A ends up in Ys.
For example: $ins(a,[t,s,\ldots],[a,t,s,\ldots])$ is *true* in $\Phi^1_{P_3}$.
$ins(a,[t,s,\ldots],[t,a,s,\ldots])$ is *true* in $\Phi^2_{P_3}$.
$ins(a,[t,s,\ldots],[t,s,a,\ldots])$ is *true* in $\Phi^3_{P_3}\ldots$.
Moreover, when $ins(A,Zs,Ys)\theta$ is *true* in $lfp(\Phi_{P_3})$, we have that

$$|Ys\theta| = |Zs\theta| + 1. \tag{12}$$

(ii) $perm(Xs,Zs)\theta$ becomes *true* in $\Phi^{|Zs\theta|+1}_{P_3}$.
This can be proven by induction on the length of $|Zs\theta|$.
$perm([],[])$ is *true* in $\Phi^1_{P_3}$;
if $|Zs\theta| > 0$ then $perm(Xs,Zs)\theta$ is *true* in Φ^α_P iff there exists an instance of $c2$,
$(perm([A'|Xs'],Ys') \leftarrow perm(Xs',Zs'), ins(A',Zs',Ys').)\theta'$,
such that
- $perm([A'|Xs'],Ys')\theta' = perm(Xs,Zs)\theta$ and
- $(perm(Xs',Zs'), ins(A',Zs',Ys'))\theta'$ is *true* in $\Phi^{\alpha-1}_P$.

Now we can apply the inductive hypothesis and the previous results in order to determine $\alpha - 1$:
- $perm(Xs',Zs')\theta'$ is, by the inductive hypothesis, *true* in $\Phi^{|Zs'\theta'|+1}_{P_3}$.
- $ins(A',Zs',Ys')\theta'$ becomes *true* at step $\Phi^n_{P_3}$, where $n \leq |Ys'\theta'|$.

By (12), $|Ys'\theta'| = |Zs'\theta'| + 1$, hence the conjunction $(perm(Xs',Zs'), ins(A',Zs',Ys'))\theta'$ becomes *true* exactly at step $\Phi^{|Ys'\theta'|}_{P_3}$. But $|Ys'\theta'| = |Zs\theta|$, hence $perm(Xs,Zs)\theta$ becomes *true* at step $\Phi^{|Zs\theta|+1}_{P_3}$.

(iii) $ord(Zs)\theta$ becomes *true* at step $\Phi^{max(1,|Zs\theta|)}_{P_3}$.
This can be proven by induction on $|Zs\theta|$.

(iv) $sort(Xs,Zs)\theta$ becomes *true* at step $\Phi^{|Zs\theta|+1}_{P_3}$.
Again, this can be proven by induction on $|Zs\theta|$.
$sort([],[])$ is $true_{\mathcal{L}}$ in $\Phi^1_{P_3}$. When $|Zs\theta| > 0$, $sort(Xs,Zs)\theta$ is in Φ^α_P iff there exists an instance of $c12$: $(sort([A \mid Xs'],Ys') \leftarrow perm(Xs',Zs'), ord(Zs'), ins(A,Zs',Ys'), ord(Ys').)\theta'$ such that
- $sort([A \mid Xs'],Ys')\theta' = sort(Xs,Zs)\theta$ and
- $(perm(Xs',Zs'), ord(Zs'), ins(A,Zs',Ys'), ord(Ys').)\theta'$ $true_{\mathcal{L}}$ in $\Phi^{\alpha-1}_P$.

Now to determine the value of $\alpha - 1$, we can use (i), (ii) and (iii):
- $perm(Xs',Zs')\theta'$ is *true* in $\Phi^{|Zs'\theta'|+1}_{P_3}$.
- $ord(Zs')\theta'$ is *true* in $\Phi^{max(1,|Zs'\theta'|)}_{P_3}$.
- $ins(A,Zs',Ys')\theta'$ is *true* in $\Phi^n_{P_3}$, where $n \leq |Ys'\theta'|$.
- $ord(Ys')\theta'$ is *true* in $\Phi^{max(1,|Ys'\theta'|)}_{P_3}$.

Since $|Zs'\theta'|+1 = |Ys'\theta'| = |Zs\theta|$, $(perm(Xs',Zs'), ord(Zs'), ins(A,Zs',Ys'), ord(Ys'))\theta'$ becomes *true* exactly at step $\Phi^{|Ys'\theta|}_{P_3}$ and $sort(Xs,Zs)$ becomes *true* at step $\Phi^{|Zs\theta|+1}_{P_3}$.

We can finally prove (b). By (iv), whenever $sort(Xs,Zs)\theta$ is *true* in $lfp(\Phi_{P_3})$, it is *true* in $\Phi^{|Zs\theta|+1}_{P_3}$; but by (ii) and (iii), whenever $(perm(Xs,Zs), ord(Zs))\theta$ is *true* in $lfp(\Phi_{P_3})$, it is *true* in $\Phi^{|Zs\theta|+1}_{P_3}$. By

Lemma 6.7 the delay of $sort(Xs, Zs)$ wrt $perm(Xs, Zs), ord(Zs)$ is zero. It follows that also the delay of $sort(Xs, Zs), perm(Xs, Zs), ord(Zs)$ wrt $perm(Xs, Zs), ord(Zs)$ is zero. □

The *thinning* operation is the converse of fattening, and allows one to eliminate superfluous literals from the body of a clause.

Definition 6.8 (thin) Let $cl : A \leftarrow \widetilde{L}, \widetilde{H}.$ be a clause in a program P.

- *Thinning cl of the literals* $\widetilde{H}$ *in* P consists of substituting cl' for cl, where $cl' : A \leftarrow \widetilde{L}$.

$thin(P, cl, \widetilde{H}) \stackrel{\text{def}}{=} P \backslash \{cl\} \cup \{cl'\}$. □

As for fattening, thinning can be interpreted as a replacement and then its applicability conditions can be inferred from Corollaries 3.16 and 4.8. Moreover Lemma 6.7 applies in a natural way also to this operation; only statement (c) requires a symmetric formulation. We restate only this last point.

Lemma 6.9 Let $cl = A \leftarrow \widetilde{E}, \widetilde{G}, \widetilde{H}.$ be a clause in P and X be a set of variables not occurring in $(A, \widetilde{E})$. The following property holds:

- If m is an integer such that, for each α and θ, $\Phi_P^{\alpha} \models_{\mathcal{L}} \neg(\exists X\widetilde{G}, \widetilde{H})\theta$ implies $\Phi_P^{\alpha+m} \models_{\mathcal{L}} \neg\exists X\widetilde{G}\theta$, then
  - $\exists X\widetilde{G}, \widetilde{H} \preceq_{\mathcal{T}_1^{\mathcal{L}}(P)} \exists X\widetilde{G}$,
  - the delay of $\exists X\widetilde{G}$ wrt $\exists X\widetilde{G}, \widetilde{H}$ in $lfp(\Phi_P)$ is smaller or equal to m.

  If m is the least of such integers, then the delay of $\exists X\widetilde{G}, \widetilde{H}$ wrt $\exists X\widetilde{G}$ in $lfp(\Phi_P)$ is exactly m.

Proof. It is a straightforward application of the fact that for any interpretation I, if $I \models_{\mathcal{L}} (\widetilde{G}, \widetilde{H})\theta$ then also $I \models_{\mathcal{L}} \widetilde{G}\theta$. □

In the Appendix A (Lemma 8.3) we state a corresponding Lemma for the case in which we adopt $\mathcal{T}_2^{\mathcal{L}}$ instead of $\mathcal{T}_1^{\mathcal{L}}$.

Example 6.5 (sorting by permutation and check, part III)

(Step (5)) We can eliminate $ord(Zs)$ from the body of $c13$ by thinning it. The resulting program is: $P_5 = \{c1, \ldots, c7\} \cup$

$$\begin{array}{lll} \{ \quad c11: & sort([\,],[\,]). & \\ \quad c14: & sort([A|Xs], Ys) \quad \leftarrow & sort(Xs, Zs), perm(Xs, Zs), ins(A, Zs, Ys), ord(Ys).\} \end{array}$$

This corresponds to replacing $ord(Zs), ins(A, Zs, Ys), ord(Ys)$ with $ins(A, Zs, Ys), ord(Ys)$. In order to prove that the operation is $\mathcal{T}_1^{\mathcal{L}}$-complete we apply Theorem 3.15.

First we have to prove that

$$\text{if } ord(Zs)\theta \text{ is } false \text{ in } lfp(\Phi_{P_4}) \text{ then } (ins(A, Zs, Ys), ord(Ys))\theta \text{ is } false \text{ in } lfp(\Phi_{P_4}) \text{ [4]}. \qquad (13)$$

[4] When adopting WDCA instead of DCA, calculations are truly more complicated, in fact in order to ensure the equivalence we have to show that for each j there is a k such that if $ord(Zs)\theta$ is $false$ in $\Phi_{P_4}^j$ then $(ins(A, Zs, Ys), ord(Ys))\theta$ is $false$ in $\Phi_{P_4}^k$.

This can be proved by the following schema: suppose that $ord(Zs)\theta$ is $false$ in $lfp(\Phi_{P_4})$ and let $Ws\theta$ be the maximal ordered prefix of $Zs\theta$, then $ord(Zs)\theta$ becomes $false$ at step $\Phi_{P_4}^{|Ws\theta|}$. We have to distinguish two cases:
- if there is no $Xs\theta$ such that $Xs\theta$ is a prefix of Ys and $ins(A, Ws, Xs)\theta$ is $true$ in some $\Phi_{P_4}^n$, then $ins(A, Zs, Ys)\theta$ becomes $false$ no later than $ord(Zs)\theta$ does, and we have the desired result.
- otherwise, either X_s is not ordered or it is the maximal ordered prefix of $Ys\theta$; in either cases, $ord(Ys)\theta$ becomes $false$ no later than step $\Phi_{P_4}^{|Xs|}$.

In any case if $ord(Zs)\theta$ is $false$ in $\Phi_{P_4}^j$ then $(ins(A, Zs, Ys), ord(Ys))\theta$ is $false$ in $\Phi_{P_4}^{j+1}$.

This is easy to prove: if $ins(A, Zs, Ys)\theta$ is *false* in $lfp(\Phi_{P_4})$ then we have the thesis. Otherwise, since $lfp(\Phi_{P_4})$ is a total interpretation, $ins(A, Zs, Ys)\theta$ cannot be *undefined* in it, and $ins(A, Zs, Ys)\theta$ is *true* in $lfp(\Phi_{P_4})$, but in this case it is easy to see that $Zs\theta$ has one element less than $Ys\theta$, and hence if $ord(Zs)\theta$ is *false* in $lfp(\Phi_{P_4})$, so is $ord(Ys)\theta$; and (2) follows.

Now (13) implies that whenever $(ord(Zs), ins(A, Zs, Ys), ord(Ys))\theta$ is *false* in $lfp(\Phi_{P_4})$ then also $(ins(A, Zs, Ys), ord(Ys))\theta$ is *false* in $lfp(\Phi_{P_4})$, and, by Lemma 6.7, that

$$ord(Zs), ins(A, Zs, Ys), ord(Ys) \preceq_{\mathcal{T}_1^{\mathcal{L}}(P_4)} ins(A, Zs, Ys), ord(Ys).$$

Since we also have that $ins(A, Zs, Ys), ord(Ys)$ is independent from $c13$, from Theorem 3.15 it follows that $lfp(\Phi_{P_4}) \subseteq lfp(\Phi_{P_5})$, that is, that the operation is $\mathcal{T}_1^{\mathcal{L}}$-complete. As in (Step 3), since $lfp(\Phi_{P_4})$ is a total interpretation, this implies that $lfp(\Phi_{P_4}) = lfp(\Phi_{P_5})$ and that the operation is also $\mathcal{T}_1^{\mathcal{L}}$-safe.

(Step 6) Finally we can eliminate $perm(Xs, Zs)$ from the body of $c14$ by a further thinning, obtaining: $P_6 = \{c1, \ldots, c7\} \cup$

$$\begin{array}{lll} \{ \quad c11: & sort([\,],[\,]). & \\ \quad c15: & sort([A|Xs], Ys) \quad \leftarrow & sort(Xs, Zs), ins(A, Zs, Ys), ord(Ys).\} \end{array}$$

This is an $O(n^3)$ sorting program, while P_0 runs in $O(n!)$. To prove the $\mathcal{T}_1^{\mathcal{L}}$-completeness of this last step, we use Theorem 3.15. Let us distinguish two cases.

- If $Xs\theta = [\,]$, then $perm(Xs, Zs)\theta$ is *false* in $\Phi^1_{P_5}$ iff $Zs\theta \neq [\,]$, but in this case also $sort(Xs, Zs)\theta$ is *false* in $\Phi^1_{P_5}$;
- otherwise observe that the body of $c2$, which defines $perm$, is contained in the body of $c14$, defining $sort$.

This implies that if some instance of $body(c2)$ is *false* in some interpretation I, then the corresponding instance of $body(c14)$ is *false* in I. Hence, if $perm([A|Xs], Zs)\theta$ is *false* in $\Phi^{j+1}_{P_5}$ then $sort([A|Xs], Zs)\theta$ is *false* in $\Phi^{j+1}_{P_5}$. It follows that

$$\text{if } (sort(Xs, Zs), perm(Xs, Zs))\theta \text{ is } false \text{ in } \Phi^j_{P_5} \text{ then } sort(Xs, Zs)\theta \text{ is } false \text{ in } \Phi^j_{P_5}.$$

By Lemma 6.9, this is sufficient to show that $sort(Xs, Zs), perm(Xs, Zs) \preceq_{\mathcal{T}_1^{\mathcal{L}}(P_5)} sort(Xs, Zs)$ and that the semantic delay of $sort(Xs, Zs), perm(Xs, Zs)$ wrt $sort(Xs, Zs)$ is zero, and hence, by Theorem 3.15, $lfp(\Phi_{P_5}) \subseteq lfp(\Phi_{P_6})$. Again, since $lfp(\Phi_{P_5})$ is already a total interpretation, this implies that $lfp(\Phi_{P_5}) = lfp(\Phi_{P_6})$, and hence, by Theorem 2.9 that the operation is $\mathcal{T}_1^{\mathcal{L}}$-safe. □

7 Conclusions

In this paper we study the simultaneous replacement operation wrt normal programs. Simultaneous replacement is a transformation operation which consists in substituting a set of conjunctions of literals $\{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$ in the bodies of some clauses, with a set of equivalent conjunctions $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$. The set of logical consequences of the program's completion is taken as the semantics of the normal program. In this way we obtain three different semantics which depend on the domain closure axioms and on the finiteness properties of the language we choose. More precisely, the semantics we consider are:

- $Comp_{\mathcal{L}}(P) \cup \mathrm{DCA}_{\mathcal{L}}$,
  where $\mathcal{L}$ is a finite language, namely it has a finite number of function symbols and DCA is the set of Domain Closure Axioms.
- $Comp_{\mathcal{L}}(P) \cup \mathrm{WDCA}_{\mathcal{L}}$,
  where $\mathcal{L}$ is a finite language, namely it has a finite number of function symbols and WDCA is the set of Weak Domain Closure Axioms.

- $Comp_{\mathcal{L}}(P)$,
  where $\mathcal{L}$ is an infinite language, this corresponds to Kunen's semantics.

All these semantics can be characterized by means of the Kleene sequence of the three valued immediate consequence operator Φ_P.

For each of these semantics we define formulas equivalence, programs equivalence and safeness of program transformations, namely their correctness and completeness, and express them also in terms of the Φ_P operator.

Furthermore, we propose applicability conditions for simultaneous replacement which guarantee safeness, that is the preservation of each semantics during the transformation. The equivalence between $\widetilde{C}_i$ and $\widetilde{D}_i$ is obviously necessary but it is generally not sufficient. In fact, we also need the equivalence to hold after the transformation. Such equivalence can be destroyed when a $\widetilde{D}_i$ depends on one of the clauses on which the replacement is performed. Hence we establish a relation between the level of dependency of $\{\widetilde{D}_1, \ldots, \widetilde{D}_n\}$ over the modified clauses and the difference in "semantic complexity" between each $\widetilde{C}_i$ and $\widetilde{D}_i$. Such semantic complexity is measured by counting the number of the applications of the immediate consequence operator which are necessary in order to determine the truth or falsity of a predicate.

By considering replacement as a generalization of other transformation operations such as thinning, fattening and reversible folding, we show how applicability conditions can be used also for them.

Acknowledgements

This work has been partially supported by "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" of CNR under grant n. 89.00026.69.

References

[1] K. Apt. Introduction to logic programming. In *Handbook of Theoretical Computer Science*, pages 493–574. Elsevier Science Publishers B.V., 1990.

[2] K. Apt and M. Bezem. Acyclic Programs. *New Generation Computing*, 9:335–363, 1991.

[3] A. Bossi and N. Cocco. Basic Transformation Operations which preserve Computed Answer Substitutions of Logic Programs. *Journal of Logic Programming*, 16:47–87, 1993.

[4] A. Bossi, N. Cocco, and S. Etalle. On Safe Folding. In M. Bruynooghe and M. Wirsing, editors, *Programming Language Implementation and Logic Programming - Proceedings PLILP'92*, volume 631 of *Lecture Notes in Computer Science*, pages 172–186. Springer-Verlag, 1992.

[5] A. Bossi, N. Cocco, and S. Etalle. Transforming normal program by replacement. In *Third Workshop on Metaprogramming in Logic, META92: Uppsala, Sweden*, June 1992.

[6] K. L. Clark. Negation as failure rule. In H. Gallaire and G. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

[7] S. Etalle. Transformazione dei programmi logici con negazione, Tesi di Laurea, Dip. Matematica Pura e Applicata, Università di Padova, Padova, Italy, July 1991.

[8] M. Fitting. A kripke-kleene semantics for logic programs. *Journal of Logic Programming*, (4), 1985.

[9] P. Gardner and J. Shepherdson. Unfold/fold transformations of logic programs. In J.-L. Lassez and e. G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*. 1991.

[10] S. Kleene. *Introduction to Metamathematics*. D. van Nostrand, Princeton, New Jersey, 1952.

[11] K. Kunen. Negation in Logic Programming. *Journal of Logic Programming*, 4:289–308, 1987.

[12] J. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.

[13] M. Maher. Correctness of a logic program transformation system. IBM Research Report RC13496, T.J. Watson Research Center, 1987.

[14] M. Maher. A transformation system for deductive databases with perfect model semantics. *Theoretical Computer Science*, to appear.

[15] T. Sato. An equivalence preserving first order unfold/fold transformation system. In *Second Int. Conference on Algebraic and Logic Programming, Nancy, France, October 1990, (Lecture Notes in Computer Science, Vol. 463)*, pages 175–188. Springer-Verlag, 1990.

[16] J. C. Shepherdson. Language and equality theory in logic programming. Technical Report PM-88-08, University Walk, Bristol, England, 1988.

[17] H. Tamaki and T. Sato. Unfold/fold transformation ol logic programs. In S. Tarnlund, editor, *2nd International Logic Programming Conference, Uppsala, Sweden, July 1984*, pages 127–138, 1984.

8 Appendix A.

Now we provide the proof of Claim 1 in Theorem 3.5. Let us first state a simple property of existentially quantified formulas.

Remark 8.1 Let $\mathcal{L}$ be any language, W and Z be sets of variables, $\widetilde{L}$ be a conjunction of literals, I a three valued $\mathcal{L}$-interpretation and θ any ground substitution. Suppose that $W \supseteq Z \cap Var(\widetilde{L})$. The following properties hold:

- If $\exists Z \widetilde{L}\theta$ is $true_{\mathcal{L}}$ in I then $\exists W \widetilde{L}\theta$ is $true_{\mathcal{L}}$ in I.
- If $\exists Z \widetilde{L}\theta$ is not $false_{\mathcal{L}}$ in I then $\exists W \widetilde{L}\theta$ is not $false_{\mathcal{L}}$ in I.

This is true in particular when Z is empty and $\exists Z \widetilde{L}\theta = \widetilde{L}\theta$. □

Claim A.1 (Claim 1 in Theorem 3.5.) Notation as in 3.5. Let I, I' be two partial interpretations. If $I' \subseteq I$ but $\Phi_{P'}(I') \not\subseteq \Phi_P(I)$, then there exist a conjunction $\widetilde{C}_j \in \{\widetilde{C}_1, \ldots, \widetilde{C}_n\}$ and a ground substitution θ such that:

- either $I' \models_{\mathcal{L}} \exists X_j \widetilde{D}_j\theta$ while $I \not\models_{\mathcal{L}} \exists X_j \widetilde{C}_j\theta$;
- or $I' \models_{\mathcal{L}} \neg\exists X_j \widetilde{D}_j\theta$ while $I \not\models_{\mathcal{L}} \neg\exists X_j \widetilde{C}_j\theta$.

Proof. Recall that $\Phi_{P'}(I') \not\subseteq \Phi_P(I)$ iff either $\Phi_{P'}(I')^+ \not\subseteq \Phi_P(I)^+$ or $\Phi_{P'}(I')^- \not\subseteq \Phi_P(I)^-$ (or both). We have to distinguish the two cases.

Case 1) Let us suppose that $\Phi_{P'}(I')^+ \not\subseteq \Phi_P(I)^+$ and let us take an atom $B \in \Phi_{P'}(I')^+ \backslash \Phi_P(I)^+$. There has to be a clause $c \in P' \backslash P$, a ground substitution θ' such that: $head(c)\theta' = B$ and $body(c)\theta'$ is $true$ in I'. $P' \backslash P = \{cl'_1, \ldots, cl'_p\}$, then there is an integer j such that: $c = cl'_j$ and $body(cl'_j)\theta' = (\widetilde{D}_{j_1}, \ldots, \widetilde{D}_{j_{r(j)}}, \widetilde{E}_j)\theta'$. is $true$ in I'.
Hence the conjunctions $\widetilde{D}_{j_1}\theta', \ldots, \widetilde{D}_{j_{r(j)}}\theta'$ are all $true$ in I'. From remark 8.1 it follows that the formulas:

$$\exists X_{j_1} \widetilde{D}_{j_1}\theta', \ldots, \exists X_{j_{r(j)}} \widetilde{D}_{j_{r(j)}}\theta' \text{ are } true_{\mathcal{L}} \text{ in } I', \tag{14}$$

where the X_i are set of variables that satisfy the locality property wrt to $\widetilde{C}_i$ and $\widetilde{D}_i$.

We know that $B = head(cl'_j)\theta' = head(cl_j)\theta'$, but since $B \notin \Phi_P(I)^+$, by definition 2.8 we have that $(\exists W\ body(cl_j))\theta'$ is not $true_{\mathcal{L}}$ in I, where $W = Var(body(cl_j)) \backslash Var(head(cl_j))$, that is,

$$(\exists W\ \widetilde{C}_{j_1}, \ldots, \widetilde{C}_{j_{r(j)}}, \widetilde{E}_j)\theta' \text{ is not } true_{\mathcal{L}} in I.$$

For each k, $W \supseteq X_{j_k} \cap Var(body(cl_j))$, now let $Y = W \backslash X_{j_1} \cup \ldots \cup X_{j_{r(j)}}$ and θ be a ground extension of θ' whose domain contains Y. Then from Remark 8.1 it follows that

$$(\exists X_{j_1}, \ldots, X_{j_{r(j)}}\ \ \widetilde{C}_{j_1}, \ldots, \widetilde{C}_{j_{r(j)}}, \widetilde{E}_j)\theta \text{ is not } true_{\mathcal{L}} \text{ in } I.$$

Since $\widetilde{E}_j\theta$ is *true* in I' and $I' \subseteq I$, then $\widetilde{E}_j\theta$ is *true* in I, by the locality property, the sets X_{j_k} are pairwise disjoint, hence one of the formulas in $\exists X_{j_1}\widetilde{C}_{j_1}\theta, \ldots, \exists X_{j_{r(j)}}\widetilde{C}_{j_{r(j)}}\theta$ is not $true_{\mathcal{L}}$ in I.
Since (14) holds also for θ, the thesis follows.

Case 2) It is perfectly symmetrical to case 1) except for the fact that it is proven by contradiction. Let us suppose that $\Phi_{P'}(I')^- \not\subseteq \Phi_P(I)^-$, and let us take an atom $B \in \Phi_{P'}(I')^- \backslash \Phi_P(I)^-$. There has to be a clause $c \in P \backslash P'$, a ground substitution θ' such that $head(c)\theta' = B$ and $body(c)\theta'$ is not *false* in I. $P \backslash P' = \{cl_1, \ldots, cl_p\}$, then there is an integer j such that: $c = cl_j$, and then the conjunction $(\widetilde{C}_{j_1}, \ldots, \widetilde{C}_{j_{r(j)}}, \widetilde{E}_j)\theta'$ is not *false* in I.
Hence the conjunctions $\widetilde{C}_{j_1}\theta', \ldots, \widetilde{C}_{j_{r(j)}}\theta'$ are all not *false* in I. From remark 8.1 it follows that:

$$\exists X_{j_1}\widetilde{C}_{j_1}\theta' \cdots \exists X_{j_{r(j)}}\widetilde{C}_{j_{r(j)}}\theta' \text{ are not } false_{\mathcal{L}} \text{ in } I. \tag{15}$$

We know that $B = head(cl_j)\theta' = head(cl'_j)\theta'$, but since $B \in \Phi_{P'}(I')^-$, by definition 2.8 we have that $(\exists W\ body(cl'_j))\theta'$ is $false_{\mathcal{L}}$ in I', with $W = Var(body(cl'_j)) \backslash Var(head(cl'_j))$, that is,

$$(\exists W\ \widetilde{D}_{j_1}, \ldots, \widetilde{D}_{j_{r(j)}}, \widetilde{E}_j)\theta' \text{ is } false_{\mathcal{L}} in I'.$$

For each k, $W \supseteq X_{j_k} \cap Var(body(cl_j))$, now let $Y = W \backslash X_{j_1} \cup \ldots \cup X_{j_{r(j)}}$ and θ be a ground extension of θ' whose domain contains Y. From remark 8.1 it follows that

$$(\exists X_{j_1}, \ldots, X_{j_{r(j)}}\ \ \widetilde{D}_{j_1}, \ldots, \widetilde{D}_{j_{r(j)}}, \widetilde{E}_j)\theta \text{ is } false_{\mathcal{L}} \text{ in } I'.$$

Since $\widetilde{E}_j\theta$ is not *false* in I and $I' \subseteq I$, $\widetilde{E}_j\theta$ is not *false* in I'. By the locality property, the sets X_{j_k} are pairwise disjoint, then one of the formulas in $\exists X_{j_1}\widetilde{D}_{j_1}\theta \cdots \exists X_{j_{r(j)}}\widetilde{D}_{j_{r(j)}}\theta$ is $false_{\mathcal{L}}$ in I'.
Since (15) holds also for θ, the thesis follows. □

Now we state two Lemmata which are the counterpart of Lemmata 6.7 and 6.9, for the case in which the closure axioms adopted are $\text{WDCA}_{\mathcal{L}}$ rather than $\text{DCA}_{\mathcal{L}}$.

Lemma 8.2 Let $cl = A \leftarrow \widetilde{E}, \widetilde{G}.$ be a clause in the normal program P, X be a set of variables not occurring in $(A, \widetilde{E})$ and $\widetilde{H}$ be another conjunction of literals. Then

(a) If for each j there exists a k such that, for each θ, $\Phi_P^j \models_{\mathcal{L}} \exists X\widetilde{G}\theta$ implies $\Phi_P^k \models_{\mathcal{L}} (\exists X\widetilde{G}, \widetilde{H})\theta$, then $\exists X\widetilde{G} \preceq_{T_2^{\mathcal{L}}(P)} \exists X\widetilde{G}, \widetilde{H}$.

(b) If for each j there exist a k such that, for each θ, $\Phi_P^j \models_{\mathcal{L}} \neg(\exists X\widetilde{G}, \widetilde{H})\theta$ implies $\Phi_P^k \models_{\mathcal{L}} \neg\exists X\widetilde{G}\theta$, then $\exists X\widetilde{G}, \widetilde{H} \preceq_{T_2^{\mathcal{L}}(P)} \exists X\widetilde{G}$.

(c) If m is an integer such that, for each n and θ, $\Phi_P^n \models_{\mathcal{L}} \exists X\widetilde{G}\theta$ implies $\Phi_P^{n+m} \models_{\mathcal{L}} (\exists X\widetilde{G}, \widetilde{H})\theta$ then
- $\exists X\widetilde{G} \preceq_{T_2^{\mathcal{L}}(P)} \exists X\widetilde{G}, \widetilde{H}$;
- the delay of $\exists X\widetilde{G}, \widetilde{H}$ wrt $\exists X\widetilde{G}$ in $Comp(P) \cup \text{WDCA}_{\mathcal{L}}$ is smaller or equal to m.
If m is the least of such integers, then the delay of $\exists X\widetilde{G}, \widetilde{H}$ wrt $\exists X\widetilde{G}$ in $Comp(P) \cup \text{WDCA}_{\mathcal{L}}$ is exactly m.

Proof. It is a straightforward application of Theorem 2.9 together with the fact that, for any interpretation I, $I \models_{\mathcal{L}} \neg\widetilde{G}\theta$ implies that $I \models_{\mathcal{L}} \neg(\widetilde{G}, \widetilde{H})\theta$. □

Lemma 8.3 Let $cl = A \leftarrow \widetilde{E}, \widetilde{G}, \widetilde{H}.$ be a clause in P and X be a set of variables not occurring in A, $\widetilde{E}$. The following property holds:

- If m is an integer such that, for each integer n and substitution θ, $\Phi_P^n \models_{\mathcal{L}} \neg\exists X(\widetilde{G}, \widetilde{H})\theta$ implies that $\Phi_P^{n+m} \models_{\mathcal{L}} \neg\exists X\widetilde{G}\theta$, then
- $\exists X\widetilde{G}, \widetilde{H} \preceq_{T_2^{\mathcal{L}}(P)} \exists X\widetilde{G}$,
- the delay of $\exists X\widetilde{G}$ wrt $\exists X\widetilde{G}, \widetilde{H}$ in Φ_P^ω is less or equal to m.
If m is the least of such integers, then the delay of $\exists X\widetilde{G}, \widetilde{H}$ wrt $\exists X\widetilde{G}$ in Φ_P^ω is exactly m.

Proof. It is a straightforward application of the fact that for any interpretation I, if $I \models_{\mathcal{L}} (\widetilde{G}, \widetilde{H})\theta$ then also $I \models_{\mathcal{L}} \widetilde{G}\theta$. □

9 Appendix B (Safeness of the Unfolding Operation)

First we need the following technical Lemma.

Lemma 9.1 Let P' be the program obtained by unfolding an atom in a clause of program P. Then for each integer i and limit ordinal β,

(a) $\Phi_P^i \subseteq \Phi_{P'}^i$ and $\Phi_{P'}^i \subseteq \Phi_P^{2i}$;

(b) $\Phi_P^i(\Phi_P^\beta) \subseteq \Phi_{P'}^i(\Phi_{P'}^\beta)$ and $\Phi_{P'}^i(\Phi_{P'}^\beta) \subseteq \Phi_P^{2i}(\Phi_P^\beta)$.

Proof. Here we adopt the same notation of definition 6.4, so $cl : A \leftarrow H, \widetilde{K}.$ is the clause of P to which we apply the unfold operation, $\{H_1 \leftarrow \widetilde{B}_1., \ldots, H_n \leftarrow \widetilde{B}_n.\}$ are the clauses of P whose heads unify with H, $\{cl'_1, \ldots, cl'_n\}$ are the resulting clauses, where, for each i, $cl'_i : (A \leftarrow \widetilde{B}_i, \widetilde{K})\theta_i.$ and $\theta_i = mgu(H, H_i)$. We also suppose that all this clauses are disjoint.

The next Claim is crucial

Claim 3 Suppose that α is an ordinal such that, for each ground τ,

(i) $\Phi_P^\alpha = \Phi_{P'}^\alpha$;

(ii) if $H\tau \in \Phi_P^{\alpha\,+}$ then there exist a substitution ϕ and an integer i such that $H\tau = H_i\theta_i\phi$ and $\widetilde{B}_i\theta_i\phi$ is *true* in Φ_P^α;

(iii) if $H\tau \in \Phi_P^{\alpha\,-}$ then for each substitution ϕ and integer i if $H\tau = H_i\theta_i\phi$ then $\widetilde{B}_i\theta_i\phi$ is *false* in Φ_P^α.

Then, for each integer j,

- $\Phi_P^j(\Phi_P^\alpha) \subseteq \Phi_{P'}^j(\Phi_{P'}^\alpha)$;
- $\Phi_{P'}^j(\Phi_{P'}^\alpha) \subseteq \Phi_P^{2j}(\Phi_P^\alpha)$.

Proof. First we prove the first statement, and we show by induction that if a ground atom R is *true* or *false* in $\Phi_P^j(\Phi_P^\alpha)$ then it is also so in $\Phi_{P'}^j(\Phi_{P'}^\alpha)$.
The base case $j = 0$ is trivial, since $\Phi_P^0(\Phi_P^\alpha) = \Phi_P^\alpha$, and from (i) we have the thesis.
Induction step, $j > 0$; we have to distinguish two cases:

1) Suppose R is *true* in $\Phi_P^j(\Phi_P^\alpha)$; then there exists a clause $d \in P$ and a substitution θ such that $R = head(d)\theta$ and $body(d)\theta$ is *true* in $\Phi_P^{j-1}(\Phi_P^\alpha)$.
If $d \neq cl$ then d belongs both to P and P', by the inductive hypothesis $body(d)\theta$ is *true* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$, and the result follows.
Otherwise, $d = cl$, $R = A\theta$ and $(H, \widetilde{K})\theta$ is *true* in $\Phi_P^{j-1}(\Phi_P^\alpha)$. So $H\theta$ is *true* in $\Phi_P^{j-1}(\Phi_P^\alpha)$.
If $j > 1$ this implies that for some integer i and substitution ϕ, $H\theta = H\theta_i\phi = H_i\theta_i\phi$ and $\widetilde{B}_i\theta_i\phi$ is *true* in $\Phi_P^{j-2}(\Phi_P^\alpha)$.
On the other hand, if $j = 1$ the fact that $H\theta$ is *true* in Φ_P^α implies, by (ii), that for some integer i and some substitution ϕ, $\widetilde{B}_i\theta_i\phi$ is *true* in Φ_P^α.
In any case, $(\widetilde{B}_i, \widetilde{K})\theta_i\phi$ is *true* in $\Phi_P^{j-1}(\Phi_P^\alpha)$ and, by inductive hypothesis, in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$. Then $body(cl'_i)\phi$ is *true* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$, it follows that, $head(cl'_i)\phi$ is *true* in $\Phi_{P'}^j(\Phi_{P'}^\alpha)$.
We can assume that $\theta|_{Var(d)} = \theta_i\phi|_{Var(d)}$, and hence that $A\theta = A\theta_i\phi$.
As $R = A\theta = A\theta_i\phi = head(cl'_i)\phi$, the result follows.

2) Suppose that R is *false* in $\Phi_P^j(\Phi_P^\alpha)$, we prove this part by contradiction.We assume that R is not *false* in $\Phi_{P'}^j(\Phi_{P'}^\alpha)$; then there exists a clause $d' \in P'$ and a substitution θ such that $R = head(d')\theta$ and $body(d')\theta$ is not *false* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$.
If $d' \notin \{cl'_1, \ldots, cl'_n\}$, then d' belongs both to P' and P, by the inductive hypothesis $body(d')\theta$ is not *false*

in $\Phi_P^{j-1}(\Phi_P^\alpha)$, and $R = head(d')\theta$ is not *false* in $\Phi_P^j(\Phi_P^\alpha)$, which is a contradiction.
Otherwise, for some integer i and substitution ϕ, $d' = cl_i'$, $R = head(cl_i')\phi = A\theta_i\phi$, and $body(cl_i')\phi$ is not *false* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$. Recall that $body(cl_i')\phi = (\widetilde{B}_i, \widetilde{K})\theta_i\phi$.
If $j > 1$, the fact that $\widetilde{B}_i\theta_i\phi$ is not *false* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$ implies that $\widetilde{B}_i\theta_i\phi$ is not *false* in $\Phi_{P'}^{j-2}(\Phi_{P'}^\alpha)$, and since $H_i \leftarrow \widetilde{B}_i.$ is a clause of P', $H\theta_i\phi = H_i\theta_i\phi$ is not *false* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$.
On the other hand, if $j = 1$, the fact that $\widetilde{B}_i\theta_i\phi$ is not *false* in $\Phi_{P'}^\alpha$ implies by (ii) that $H\theta_i\phi$ is not *false* in $\Phi_{P'}^\alpha$.
In any case $(H, \widetilde{K})\theta_i\phi$ is not *false* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$, and by the inductive hypothesis, in $\Phi_P^{j-1}(\Phi_P^\alpha)$. Since $H, \widetilde{K} = body(cl)$ it follows that $R = A\theta_i\phi = head(cl)\theta_i\phi$ is not *false* in $\Phi_P^j(\Phi_P^\alpha)$, which gives a contradiction.

Now we prove the second statement: we show by induction that if a ground atom R is *true* or *false* in $\Phi_{P'}^j(\Phi_{P'}^\alpha)$ then it is also so in $\Phi_P^{2j}(\Phi_P^\alpha)$.
As above, the base case $j = 0$ is trivial.
Induction step $j > 0$: we have to distinguish two cases.

1) Suppose that R is *true* in $\Phi_{P'}^j(\Phi_{P'}^\alpha)$, then there exists a clause $d' \in P'$ and a substitution θ such that $R = head(d')\theta$ and $body(d')\theta$ is *true* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$.
If $d' \notin \{cl_1', \ldots, cl_n'\}$ then d' belongs both to P' and P, by the inductive hypothesis $body(d')\theta$ is *true* in $\Phi_P^{j-1}(\Phi_P^\alpha)$, $R = head(d')\theta$ is *true* in $\Phi_P^j(\Phi_P^\alpha)$ and the result follows.
Otherwise for some integer i and substitution ϕ, $d' = cl_i'$, $R = head(cl_i')\phi = A\theta_i\phi$, and $body(cl_i')\phi$ is *true* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$.
Recall that $body(cl_i')\phi = (\widetilde{B}_i, \widetilde{K})\theta_i\phi$; by inductive hypothesis, $(\widetilde{B}_i, \widetilde{K})\theta_i\phi$ is also *true* in $\Phi_P^{2j-2}(\Phi_P^\alpha)$.
Since $\widetilde{B}_i\theta_i\phi$ is *true* in $\Phi_P^{2j-2}(\Phi_P^\alpha)$ and $H_i \leftarrow \widetilde{B}_i.$ is a clause of P, $H_i\theta_i\phi$ is *true* in $\Phi_P^{2j-1}(\Phi_P^\alpha)$. But $H_i\theta_i\phi = H\theta_i\phi$, so $(H, \widetilde{K})\theta_i\phi = body(cl)\theta_i\phi$ is *true* in $\Phi_P^{2j-1}(\Phi_P^\alpha)$, hence $R = A\theta_i\phi = head(cl)\theta_i\phi$ is *true* in $\Phi_P^{2j}(\Phi_P^\alpha)$.

2) Let R be *false* in $\Phi_{P'}^j(\Phi_{P'}^\alpha)$; we prove this part by contradiction, so we assume that R is not *false* in $\Phi_P^{2j}(\Phi_P^\alpha)$. Then there exists a clause $d \in P$ and a substitution θ such that $R = head(d)\theta$ and $body(d)\theta$ is not *false* in $\Phi_P^{2j-1}(\Phi_P^\alpha)$.
If $d \neq cl$ then d belongs both to P and P', by the monotonicity of the Kleene sequence, $body(d)\theta$ is not *false* in $\Phi_P^{2j-2}(\Phi_P^\alpha)$ either, hence, by the inductive hypothesis $body(d)\theta$ is not *false* in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$. It follows that $head(d)\theta = R$ is not *false* in $\Phi_{P'}^j(\Phi_{P'}^\alpha)$ which gives a contradiction.
Otherwise, $d = cl$, $R = A\theta$ and $(H, \widetilde{K})\theta$ is not *false* in $\Phi_P^{2j-1}(\Phi_P^\alpha)$. So $H\theta$ is not *false* in $\Phi_P^{2j-1}(\Phi_P^\alpha)$. This implies that for some integer i and substitution ϕ, $H\theta = H\theta_i\phi = H_i\theta_i\phi$ and $\widetilde{B}_i\theta_i\phi$ is not *false* in $\Phi_P^{2j-2}(\Phi_P^\alpha)$.
Hence $(\widetilde{B}_i, \widetilde{K})\theta_i\phi$ is not *false* in $\Phi_P^{2j-2}(\Phi_P^\alpha)$, and by the inductive hypothesis, in $\Phi_{P'}^{j-1}(\Phi_{P'}^\alpha)$. Since $\widetilde{B}_i\theta_i\phi = body(cl_i')\phi$, this implies that $head(cl_i')\phi = A\theta_i\phi = R$ is not *false* in $\Phi_{P'}^j(\Phi_{P'}^\alpha)$ which is a contradiction. □

Now, in order to prove (a) we observe that $\alpha = 0$ is an ordinal that trivially satisfies the hypothesis of Claim 3.

In order to prove (b) we have to show that Claim 3 also applies when α is any limit ordinal.
First consider the case $\alpha = \omega$. From (a) it follows that $\Phi_P^\omega = \Phi_{P'}^\omega$, moreover, if $H\tau$ is *true* (resp. *false*) in Φ_P^ω, then, it is also *true* in some Φ_P^m, $(m < \omega)$. By applying the definition of Fitting's operator we have that condition (ii) (resp. (iii)) hold for $\alpha = \omega$. So $\alpha = \omega$ satisfies the requirements of Claim 3.
It follows that, for each i, $\Phi_P^{\omega+i} \subseteq \Phi_{P'}^{\omega+i}$ and that $\Phi_{P'}^{\omega+i} \subseteq \Phi_P^{\omega+2i}$. By the same reasoning it turns out that the ordinal 2ω, and iterating, all the other limit ordinals, satisfy the requirements of Claim 3. □

This brings us to the desired conclusions.

Corollary 9.2 (safeness of the unfolding operation) Let P' be the result of unfolding an atom of a clause in P. Then P is equivalent to P' wrt both $\mathcal{T}_1^{\mathcal{L}}$ and $\mathcal{T}_2^{\mathcal{L}}$

Proof. By Lemmata 9.1, 3.1 and Theorem 4.3. □