Simultaneous Replacement in Normal Programs

ANNALISA BOSSI, *Dipartimento di Matematica Pura ed Applicata,
Università di Padova, Via Belzoni 7, 35131 Padova, Italy.*
E-mail: bossi@zenone.math.unipd.it

NICOLETTA COCCO, *Dipartimento di Informatica, Università di
Venezia-Ca' Foscari, Via Torino 155, Mestre-Venezia, Italy.*
E-mail: cocco@moo.dsi.unive.it

SANDRO ETALLE, *Dipartimento di Matematica Pura ed Applicata
Università di Padova, Via Belzoni 7, 35131 Padova, Italy.*
and
CWI P.O. Box 94079, 1090 GB Amsterdam, The Netherlands.
E-mail sandro@cw.nl, sandro@disi.unige.it

Abstract

The simultaneous replacement transformation operation is here defined and studied w.r.t. normal programs. We give applicability conditions able to ensure the correctness of the operation w.r.t. 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.

Keywords: Program transformation, logic programming, semantics, negation, replacement.

1 Introduction

1.1 The replacement operation

The *replacement* operation was introduced for transforming definite programs by Tamaki and Sato in [20] and after that it was rather neglected by people working on program transformations apart from Sato himself [18], Maher [16] and Gardner and Shepherdson [12]. 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.

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 we find different proposals. In [20] 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 than or equal to the size of the smallest proof tree for D . Gardner and Shepherdson [12] 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 [16, 17] studies replacement w.r.t. Success set, Finite Failure Set, Ground Finite Failure Set and Perfect Model semantics. Sato [18] considers also replacement of formulas whose equivalence can be proved in first-order logic and does not depend on the program. Bossi *et al.* studied the correctness of this operation w.r.t. the S-semantics for definite programs [4], Fitting's semantics [5] and the well-founded semantics for normal programs [9].

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 semantics of the logical consequences of the program completion (Kunen's semantics). We also take into consideration the case in which we adopt some domain closure axioms, this will allow us to draw conclusions for Fitting's semantics as well. As a side-effect, we also provide a characterization of program equivalence w.r.t. Kunen's semantics by referring solely to the Kleene's sequence of Fitting's operator Φ_P .

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 alone is not sufficient to avoid the risk of introducing a loop. For this reason we introduce two new concepts: the *semantic delay between two conjunctions of literals* and the *dependency degree of a conjunction of literals w.r.t. a clause*: the applicability conditions for replacement we propose 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 some situation in which 'there is no space to introduce a loop'. Such applicability conditions are undecidable in general, but decidable syntactic conditions can be derived for special cases. For instance in [5] we consider two such cases when replacement simulates folding, while in [7] these results are used for proving the correctness of an unfold/fold transformation sequence w.r.t. Fitting's semantics.

1.2 Structure of the paper

In Section 2 the main definitions related to the semantics given by the program's completion are briefly recalled. In Section 3 we restrict ourselves to the case of an infinite language, define equivalence among programs and characterize it via the three-valued operator Φ_P . In Section 4 simultaneous replacement is introduced and the correctness of a transformation operation is defined. Then we state and prove the results on the correctness and completeness of the operation w.r.t. to the considered semantics. We also show how reversible folding and recursive folding can be dealt with as special cases of the replacement operation. In Section 5 we consider the case of a finite language and henceforth the semantics given by the program's completion together with some closure axioms. Both DCA and WDCA are considered and the results of Section 4 are reformulated for such cases. 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. A short conclusion follows. Part of the proofs are given in the Appendices.

2 Preliminaries

2.1 Notation

We assume that the reader is familiar with the basic concepts of logic programming; throughout the paper we use the standard terminology of [15] and [1]. We consider *normal programs*, that is finite collections of *normal rules*, $A \leftarrow L_1, \dots, L_m$, where A is an atom and $L_1, \dots, L_m$ are literals. Symbols with a $\sim$ on top denote tuples of objects, for instance $\tilde{x}$ denotes a tuple of variables $x_1, \dots, x_n$, and $\tilde{x} = \tilde{y}$ stands for $x_1 = y_1 \wedge \dots \wedge x_n = y_n$. We also adopt the usual logic programming notation that uses “,” instead of $\wedge$, hence a conjunction of literals $L_1 \wedge \dots \wedge L_n$ will be denoted by $L_1, \dots, L_n$ or by $\tilde{L}$.

In this paper we always work with three-valued logic: the truth values are then *true*, *false* and *undefined*. We adopt the truth tables of [13], which can be summarized as follows: 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. In particular in the sequel we use the symbol $\Leftrightarrow$ corresponding to 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*. As opposed to this, the usual $\leftrightarrow$ is *undefined* when one of its arguments is *undefined*.

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

DEFINITION 2.1

- A logic connective $\Diamond$ is *allowed* iff the following property holds: when $a \Diamond b$ is *true* or *false* then its truth value does not change if the interpretation of one of its argument is changed from *undefined* to *true* or *false*.
- A first-order formula is *allowed* iff 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 $\{\text{undefined}, \text{true}, \text{false}\}$ which has *undefined* as bottom element and *true* and *false* are not comparable.

2.2 Completion for normal programs

In this paper we consider as semantics for a normal logic program P the set of all logical consequences of its completion $\text{Comp}(P)$ [8]; the problem of the consistency of $\text{Comp}(P)$ is here avoided by using three-valued logic instead of the classical two-valued logic.

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

DEFINITION 2.2

Let P be a program and $p(\tilde{t}_1) \leftarrow \tilde{B}_1, \dots, p(\tilde{t}_r) \leftarrow \tilde{B}_r$ be all the clauses which define predicate

symbol p in P . The *completed definition* of p is

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

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

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

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

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. First recall that 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.

DEFINITION 2.3

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

- $f(x_1, \dots, x_n) \neq g(y_1, \dots, y_m)$ for all distinct f, g in $\mathcal{L}$;
- $f(x_1, \dots, x_n) = f(y_1, \dots, y_n) \rightarrow (x_1 = y_1) \wedge \dots \wedge (x_n = y_n)$ 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, since an expression of the form $t = s$, with t, s ground terms cannot be *undefined*.

DEFINITION 2.4

The *Clark's completion* of P w.r.t. the language $\mathcal{L}$, $\text{Comp}_{\mathcal{L}}(P)$ consists in the conjunction of the completed definition of all the predicates in P together with $\text{CET}_{\mathcal{L}}$.

2.3 The language problem

The semantics determined by $\text{Comp}(P)$ depends 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 $\text{Comp}(P)$ is not complete. This problem can be solved by adding to $\text{Comp}(P)$ some domain closure axioms which are intended to restrict the interpretation of the quantification to $\mathcal{L}$ -terms. The situation is further complicated by the fact that in the literature we find two different kind of such axioms: the strong (DCA) and the weak (WDCA) ones. In total there exist three different 'main' approaches, namely we may:

(a) Consider an infinite language, with no domain closure axioms. This is the approach followed by Kunen [14].

(b) Consider a finite language and adopt the weak domain closure axioms (WDCA). This has been studied by Shepherdson [19], and the results are similar to the ones found for the case of an infinite language (case (a) above).

(c) 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 $\mathcal{L}(P)$.

In this paper we consider the three cases separately: first we analyse the case in which the language is infinite, then in Section 6 we discuss how the results have to be modified when we drop the infiniteness assumption.

2.4 Fitting's operator

Fitting's operator can be considered the three-valued counterpart of the usual (two-valued) immediate consequence operator T_P , and it is extremely useful for characterizing the semantics we are going to refer to in the sequel. We begin with the following definition.

DEFINITION 2.5

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 $\{\text{true}, \text{false}, \text{undefined}\}$.

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 $I = I \cap J$, that is iff $I^+ \subseteq J^+$ and $I^- \subseteq J^-$.

The underlying universe of an $\mathcal{L}$ -interpretation is the universe of $\mathcal{L}$ -terms, consequently when we say that a first-order formula ϕ is *true* in I , $I \models \phi$, 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 [11]. 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 and by $\mathcal{L}(P)$ the (finite) language consisting of the functions and predicate symbols actually occurring in the program P .

DEFINITION 2.6

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 $c : B \leftarrow \tilde{L}$ in P whose head unifies with A , $\theta = mgu(A, B)$, and $\exists \tilde{w} \tilde{L}\theta$ is *true* in I
where $\tilde{w}$ is the set of local variables of c , $\tilde{w} = Var(\tilde{L}) \setminus Var(B)$.
- A ground atom A is *false* in $\Phi_P(I)$, ($A \in \Phi_P(I)^-$)
iff for all clauses $c : B \leftarrow \tilde{L}$ in P for which there exists $\theta = mgu(A, B)$ we have that $\exists \tilde{w} \tilde{L}\theta$ is *false* in I
where $\tilde{w}$ is the set of local variables of c , $\tilde{w} = Var(\tilde{L}) \setminus Var(B)$.

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 $\Phi_P^0, \Phi_P^1, \dots, \Phi_P^k, \dots, \Phi_P^\omega, \dots$ 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 model semantics of $Comp_{\mathcal{L}}(P)$, in fact Fitting in [11] shows that the three-valued Herbrand models of $Comp_{\mathcal{L}}(P)$ are exactly the fixpoints of Φ_P ; it follows that any program has a *least* (w.r.t. $\subseteq$) three-valued Herbrand model, which coincides with the least fixed point of Φ_P . This model is usually referred to as Fitting's model.

EXAMPLE 2.7

Let P be the following program:

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

And let $\mathcal{L} = \mathcal{L}(P)$. We have that

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

3 Semantic issues

In this section and in the following one, we will always refer to a fixed but unspecified *infinite* language $\mathcal{L}$, that we assume contains all the function symbols of the programs we are considering. Here by *infinite* language, we mean a language that contains infinitely many function symbols (including those of arity 0).

Later, in Section 5, we discuss the problems that arise when the language is finite and we show how the results we give here have to be modified in order to be applied in this other context.

The aim of this section is to define and characterize program's equivalence, this will provide the theoretical background for the analysis of the correctness of the transformation. The result we prove here is partially a strengthening of [18, Proposition 3.4] (however, in [18] the more general setting of first-order programs under any base theory is considered).

As far as we are concerned in this paper, the semantics of a normal program P is the set of logical consequences of $Comp_{\mathcal{L}}(P)$. Consequently, program's equivalence is defined as follows.

DEFINITION 3.1

We say that P and P' are *equivalent* iff for each allowed formula ϕ

- $Comp_{\mathcal{L}}(P) \models \phi$ iff $Comp_{\mathcal{L}}(P') \models \phi$.

Three-valued program completion semantics in the case of an infinite language has been studied by Kunen [14] and successively by Shepherdson [19]. For this reason, following the literature, we refer to it as *Kunen's semantics*. The main result is the following.

THEOREM 3.2 ([14])

Let P be a normal program and ϕ an allowed formula

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

PROOF. This is basically Theorem 6.3 in [14], however, in [14] it is assumed that the language contains a countably infinite number of symbols of each arity. Later, Shepherdson noticed that the result holds for any infinite language [19, Theorem 5b]. ■

Equivalence of two programs can be inferred by comparing the Kleene's sequences of the Φ_P operator.

THEOREM 3.3

Let P_1 and P_2 be two normal programs.

If

$$\forall n \exists m \Phi_{P_1}^n \subseteq \Phi_{P_2}^m$$

then for all ϕ ,

$$Comp_{\mathcal{L}}(P_1) \models \phi \text{ implies } Comp_{\mathcal{L}}(P_2) \models \phi$$

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

PROOF. Let us assume $\forall n \exists m \Phi_{P_1}^n \subseteq \Phi_{P_2}^m$, and let ϕ be any allowed formula such that $Comp_{\mathcal{L}}(P_1) \models \phi$. By Theorem 3.2, there exists an integer n such that $\Phi_{P_1}^n \models \phi$; by the hypothesis there exists an m such that $\Phi_{P_1}^n \subseteq \Phi_{P_2}^m$, hence $\Phi_{P_2}^m \models \phi$.

Again, by Theorem 3.2, this implies that $Comp_{\mathcal{L}}(P_2) \models \phi$. ■

A similar result has been proved by Sato in [18] where the more general setting of first-order programs under any base theory is considered.

Interestingly, also the inverse implication holds. As the proof is quite long, we defer it to Appendix A.

THEOREM 3.4

Let P_1 and P_2 be two normal programs.

If for all ϕ ,

$$Comp_{\mathcal{L}}(P_1) \models \phi \text{ implies } Comp_{\mathcal{L}}(P_2) \models \phi$$

then

$$\forall n \exists m \Phi_{P_1}^n \subseteq \Phi_{P_2}^m$$

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

PROOF. The proof is given in Appendix A. ■

These results allow us to characterize program equivalence: following Sato [18], we say that two programs P_1, P_2 are *chain equivalent* iff $\forall n \exists m \ \Phi_{P_1}^n \subseteq \Phi_{P_2}^m$ and $\Phi_{P_1}^m \supseteq \Phi_{P_2}^n$. Using this notation, from the previous theorems, we immediately have the following.

COROLLARY 3.5

Let P_1 and P_2 be normal programs, then

- P_1 and P_2 are equivalent iff they are chain equivalent.

Notice that, given two programs P_1, P_2 , the fact that $\Phi_{P_1}^\omega = \Phi_{P_2}^\omega$ is necessary but not sufficient to ensure that P_1 is equivalent to P_2 . This is due to the fact that the set of ground atomic logical consequences of $\text{Comp}_{\mathcal{L}}(P)$ (which coincide with Φ_P^ω) is not sufficient to fully characterize Kunen's semantics of a program P . Consider for instance the following two programs [14]: $P_1 = \{\text{void}(s(X)) \leftarrow \text{void}(X).\}$ and $P_2 = \{\text{void}(X) \leftarrow f.\}$ where the predicate f has no clause defining it in either programs, and consequently it is always *false*. For any term t , the predicate $\text{void}(t)$ is *false* before $\Phi_{P_1}^\omega$, and indeed we have that $\Phi_{P_1}^\omega = \Phi_{P_2}^\omega$, however P_1 is not equivalent to P_2 , in fact we have that $\text{Comp}_{\mathcal{L}}(P_2) \models \forall X \neg \text{void}(X)$ while $\text{Comp}_{\mathcal{L}}(P_1) \not\models \forall X \neg \text{void}(X)$. This is reflected by the fact that $\Phi_{P_2}^2 \models \forall X \neg \text{void}(X)$ while there is no integer n such that $\Phi_{P_1}^n \models \forall X \neg \text{void}(X)$. Indeed, P_1 has a model which contains, besides the (representation of) natural numbers, also an infinite chain of terms t_i such that for each i , $\text{void}(t_i)$ is *true*.

4 Correctness of the replacement operation

4.1 The simultaneous replacement operation

The replacement operation has been introduced by Tamaki and Sato in [20] for definite programs. Syntactically it consists in substituting a conjunction, $\tilde{C}$, of literals with another one, $\tilde{D}$, in the body of a clause. Similarly, *simultaneous* replacement consists in substituting a set of conjunctions of literals $\{\tilde{C}_1, \dots, \tilde{C}_n\}$, with another corresponding set of conjunctions $\{\tilde{D}_1, \dots, \tilde{D}_n\}$ in the bodies of some clauses $\{cl_1, \dots, cl_p\}$ of a program P . We assume that if $i \neq j$ then $\tilde{C}_i$ and $\tilde{C}_j$ do not overlap, even if they may actually represent identical literals, that is, they are either in different clauses or in disjoint subsets of the same clause.

Note that, because of the semantics we consider, the order of literals in the bodies of the clauses is irrelevant.

Correctness of a transformation

Assume P' is obtained by transforming P , then Definition 3.1 (program equivalence) is used to define the *correctness* of a transformation operation as follows.

DEFINITION 4.1

Let P, P' be normal programs. Suppose that P' is obtained by applying a transformation operation to P . We say that the transformation is

- *Partially Correct* when for each allowed formula ϕ , if $\text{Comp}_{\mathcal{L}}(P') \models \phi$ then also $\text{Comp}_{\mathcal{L}}(P) \models \phi$.
- *Complete* when for each allowed formula ϕ , if $\text{Comp}_{\mathcal{L}}(P) \models \phi$ then also $\text{Comp}_{\mathcal{L}}(P') \models \phi$.

- *Totally Correct* or *Safe* when it is both partially correct and complete. This is the case in which P and P' are *equivalent*.

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.

4.2 Partial correctness

When we replace the conjunction $\tilde{C}$ with $\tilde{D}$ in the body of a clause, we are actually replacing a subformula inside a formula, the clause itself. Clearly, some conditions are needed to guarantee the safeness of the operation. When we abstract from the particular context, that is from the specific clause where the replacement occurs, a natural condition for replacing a (possibly open) formula χ by a (possibly open) formula ϕ is their *equivalence* in the sense of the following definition.

Before stating it we need to establish some further notation: given the formulas ζ , χ and ϕ , we denote by $\zeta[\phi/\chi]$ the formula obtained from ζ by replacing all occurrences of the subformula χ by ϕ .

DEFINITION 4.2 (equivalence of formulas)

Let χ , ϕ be first order formulas. We say that

- χ is *less specific or equal to* ϕ w.r.t. $\text{Comp}_{\mathcal{L}}(P)$, $\chi \preceq_{\text{Comp}_{\mathcal{L}}(P)} \phi$, iff for each allowed formula ζ and each substitution σ ,

$$\text{Comp}_{\mathcal{L}}(P) \models \zeta\sigma \text{ implies } \text{Comp}_{\mathcal{L}}(P) \models \zeta[\phi/\chi]\sigma;$$

- χ is *equivalent to* ϕ w.r.t. $\text{Comp}_{\mathcal{L}}(P)$, $\chi \cong_{\text{Comp}_{\mathcal{L}}(P)} \phi$, iff $\chi \preceq_{\text{Comp}_{\mathcal{L}}(P)} \phi$ and $\phi \preceq_{\text{Comp}_{\mathcal{L}}(P)} \chi$.

The following example shows how the problem of the equivalence of formulas naturally arises when using the replacement operation.

EXAMPLE 4.3

Let us consider the following program:

```

m1(El, [El | Tail], s(0)).
m1(El, [X | Tail], s(N))    ← m1(El, Tail, N).
m2(El, [El | Tail]).
m2(El, [X | Tail])          ← m2(El, Tail).
d : common_element(L1, L2) ← m1(El, L1, N1), m1(El, L2, N2).
```

Both 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 *common_element* goes, this is totally unnecessary, and we can replace the conjunction $m1(El, L1, N1), m1(El, L2, N2)$ with the 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

$d' : \text{common_element}(L1, L2) \leftarrow m2(El, L1), m2(El, L2).$

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

$$\text{common_element}(L1, L2) \Leftrightarrow \exists N, M. m1(El, L1, N), m1(El, L2, M), \quad (4.1)$$

while after the transformation it is

$$\text{common_element}(L1, L2) \Leftrightarrow m2(El, L1), m2(El, L2). \quad (4.2)$$

When applying a replacement we want the replacing conjunction to be semantically equivalent to the replaced one. In this particular case we can formalize this statement by requiring the equivalence of the two 'bodies', (4.1) and (4.2), of the completed definition of *common_element*, that is, we require that

$$\exists N, M. m1(El, L1, N), m1(El, L2, M) \cong_{Comp_L(P)} m2(El, L1), m2(El, L2). \quad (4.3)$$

Which is easy to prove true.

In (4.3) we have specified two existentially quantified variables: N and M which are local to the replaced conjunct. If we didn't do so, (4.3) would not hold, as $m1(El, L1, N), m1(El, L2, M) \not\cong_{Comp_L(P)} m2(El, L1), m2(El, L2)$. In the sequel, when replacing, say, $\tilde{C}$ with $\tilde{D}$, we always specify a set $\tilde{x}$ of 'local' variables, which are variables that can appear in either $\tilde{C}$ or $\tilde{D}$ (or both) but cannot occur in the rest of the clause where $\tilde{C}$ is found. Consequently, our first requirement is the equivalence of $\exists \tilde{x} \tilde{C}$ and $\exists \tilde{x} \tilde{D}$. Such an equivalence is weaker than the equivalence between $\tilde{C}$ and $\tilde{D}$, but still sufficient for our purposes.

We now formalize this concept of local variables for simultaneous replacement. First let us establish the notation we will use throughout the paper.

NOTATION 4.4

P is the normal program we want to transform.

$\tilde{C}_1, \dots, \tilde{C}_n$ are the conjunctions of literals we want to replace with $\tilde{D}_1, \dots, \tilde{D}_n$.

$\{cl_1, \dots, cl_p\}$ is the subset of P consisting of the clauses that are going to be affected by the transformation.

P' is the result of the transformation.

DEFINITION 4.5 (locality property)

Referring to Notation 4.4, we say that a set of variables $\tilde{x}_i$ satisfies the *locality property* with respect to $\tilde{C}_i$ and $\tilde{D}_i$ if the following holds:

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

Note that the locality property is trivially satisfied when $\tilde{x}_i$ is empty. Note also that the locality property implies that if $\tilde{C}_h$ and $\tilde{C}_k$ occur in the same clause then the corresponding $\tilde{x}_h$ and $\tilde{x}_k$ are disjoint.

Before we state the result on partial correctness, we have to give a characterization of the equivalence of formulas w.r.t. Kunen's semantics, which refers solely to the Kleene sequence of the operator Φ_P . Here we denote by $FV(\chi)$ the set of free variables in a formula χ .

LEMMA 4.6

Let P be a normal program, χ, ϕ be first-order allowed formulas and $\tilde{x} = \{x_1, \dots, x_k\} = FV(\chi) \cup FV(\phi)$. The following statements are equivalent

- (a) $\chi \preceq_{Comp_L(P)} \phi$;
- (b) $\forall n \exists m \forall \tilde{t} \quad \Phi_P^n \models (\neg)\chi(\tilde{t}/\tilde{x}) \text{ implies } \Phi_P^m \models (\neg)\phi(\tilde{t}/\tilde{x})$;

where n, m are quantified over natural numbers and $\tilde{t}$ is quantified over k -tuples of $\mathcal{L}$ -terms.

PROOF. The proof is given in Appendix A. ■

We can finally state the result on partial correctness of the replacement operation we were aiming at. As we anticipated at the beginning of this Section, when replacing $\tilde{C}$ with $\tilde{D}$, our first requirement is the equivalence of $\exists \tilde{x} \tilde{C}$ and $\exists \tilde{x} \tilde{D}$, where $\tilde{x}$ is a set of variables satisfying the locality property. However, if we are only interested in proving the *partial* correctness of the operation, a partial equivalence (namely, that $\exists \tilde{x} \tilde{D} \preceq_{Comp_{\mathcal{L}}(P)} \exists \tilde{x} \tilde{C}$) is perfectly sufficient. This is shown by the following theorem. Again we adopt Notation 4.4.

THEOREM 4.7 (partial correctness)

If for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, there exists a (possibly empty) set of variables $\tilde{x}_i$ satisfying the locality property w.r.t. $\tilde{C}_i$ and $\tilde{D}_i$ such that

$$\exists \tilde{x}_i \tilde{D}_i \preceq_{Comp_{\mathcal{L}}(P)} \exists \tilde{x}_i \tilde{C}_i,$$

then the simultaneous replacement operation is partially correct.

PROOF. First let us make the following observation. With the exception of clauses $\{cl_1, \dots, cl_p\}$, P is just like P' . Hence if for each i , $\exists \tilde{x}_i \tilde{C}_i$ and $\exists \tilde{x}_i \tilde{D}_i$ had the same meaning in a given interpretation I (that is, if $I \models \exists \tilde{x}_i \tilde{C}_i \Leftrightarrow \exists \tilde{x}_i \tilde{D}_i$), then we would have that $\Phi_P(I) = \Phi_{P'}(I)$. It follows that whenever $\Phi_P(I) \neq \Phi_{P'}(I)$, there has to be an index j such that $\exists \tilde{x}_j \tilde{C}_j$ and $\exists \tilde{x}_j \tilde{D}_j$ have different meanings in I . This idea is formalized and extended in the following Lemma, whose proof is given in Appendix A.

LEMMA 4.8

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 $\tilde{C}_j \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$ and a ground substitution θ such that:

- either $I' \models \exists \tilde{x}_j \tilde{D}_j \theta$, while $I \not\models \exists \tilde{x}_j \tilde{C}_j \theta$;
- or $I' \models \neg \exists \tilde{x}_j \tilde{D}_j \theta$, while $I \models \neg \exists \tilde{x}_j \tilde{C}_j \theta$.

Now we proceed with the proof, which is by contradiction. By Theorems 3.3 and 3.4 the operation is partially correct iff $\forall n \exists m \Phi_P^m \supseteq \Phi_{P'}^n$, so 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 Lemma 4.8, it follows that for each integer l there exist an integer $j(l) \in \{1, \dots, n\}$ and a ground substitution θ_l such that:

$$\begin{aligned} \exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \theta_l \text{ is true (or false) in } \Phi_P^i, \text{ while} \\ \exists \tilde{x}_{j(l)} \tilde{C}_{j(l)} \theta_l \text{ is not true (resp. false) in } \Phi_P^{l+i}. \end{aligned} \tag{4.4}$$

By hypothesis $\exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \preceq_{\text{Comp}_{\mathcal{L}}(P)} \exists \tilde{x}_{j(l)} \tilde{C}_{j(l)}$, we can then apply Lemma 4.6 to the left-hand side of (4.4). It follows that there has to be an integer r such that for each l ,

$$\exists \tilde{x}_{j(l)} \tilde{C}_{j(l)} \theta_l \text{ is true (resp false) in } \Phi_P^r;$$

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

$$\text{for each } l \text{ such that } l + i > r, \exists \tilde{x}_{j(l)} \tilde{C}_{j(l)} \theta_l \text{ is true (resp false) in } \Phi_P^{l+i}.$$

This contradicts (4.4). ■

An immediate consequence of Theorem 4.7 is the following simple corollary on total correctness.

COROLLARY 4.9

Using Notation 4.4, if for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, there exists a (possibly empty) set of variables $\tilde{x}_i$ satisfying the locality property w.r.t. $\tilde{C}_i$ and $\tilde{D}_i$ such that

$$\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_{\mathcal{L}}(P)} \exists \tilde{x}_i \tilde{C}_i,$$

then P is equivalent to P' iff, for each i , $\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_{\mathcal{L}}(P')} \exists \tilde{x}_i \tilde{C}_i$.

PROOF. ‘if’. From the assumption that $\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_{\mathcal{L}}(P)} \exists \tilde{x}_i \tilde{C}_i$ and Theorem 4.7 it follows that for each allowed formula ϕ , if $\text{Comp}_{\mathcal{L}}(P') \models \phi$ then $\text{Comp}_{\mathcal{L}}(P) \models \phi$. Now P can be re-obtained from P' by replacing each $\tilde{D}_i$ with $\tilde{C}_i$, moreover each set of variables $\tilde{x}_i$ satisfies the locality property w.r.t. $\tilde{C}_i$ and $\tilde{D}_i$ also in P' . Since by hypothesis $\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_{\mathcal{L}}(P')} \exists \tilde{x}_i \tilde{C}_i$, from Theorem 4.7 it also follows that, if $\text{Comp}_{\mathcal{L}}(P) \models \phi$, then $\text{Comp}_{\mathcal{L}}(P') \models \phi$.

‘only if’. It is easy to see that if $\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_{\mathcal{L}}(P)} \exists \tilde{x}_i \tilde{C}_i$ and P is equivalent to P' then $\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_{\mathcal{L}}(P')} \exists \tilde{x}_i \tilde{C}_i$. ■

Roughly speaking, this corollary states that if the replacing and the replaced conjunctions are equivalent both in the initial and the resulting program, then the transformation is safe.

Of course this result requires some knowledge of the semantics of the resulting program and therefore it is not quite satisfactory: what we want are applicability conditions for the replacement operation which are based solely on the semantic properties of the *initial* program. To this is devoted the rest of this section.

4.3 Semantic delay and dependency degree

As we proved in the previous section, if $\tilde{x}$ is a set of variables that satisfies the locality property, the equivalence of $\exists \tilde{x} \tilde{C}$ and $\exists \tilde{x} \tilde{D}$ w.r.t. $\text{Comp}_{\mathcal{L}}(P)$ is sufficient to guarantee the partial correctness of the replacement. Unfortunately this is not enough to ensure total correctness.

This is shown by the next example.

EXAMPLE 4.10

Let P be the following definite program:

$$P = \left\{ \begin{array}{ll} p \leftarrow q. \\ cl : q \leftarrow r. \\ r. \end{array} \right\}$$

Let also $\mathcal{L} = \mathcal{L}(P)$. In this case p, q and r are all *true* in all the models of $Comp_{\mathcal{L}}(P)$, they are actually *equivalent* w.r.t. $Comp_{\mathcal{L}}(P)$. However, if we replace r with p in the body of cl we obtain

$$P' = \left\{ \begin{array}{l} p \leftarrow q. \\ cl' : \quad q \leftarrow p. \\ \quad r. \end{array} \right\}$$

which is by no means equivalent to the previous program. In fact we have introduced a loop and p and q are no longer *true* in all the models of $Comp_{\mathcal{L}}(P)$.

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:

$$P = \left\{ \begin{array}{l} m(X) \leftarrow n(s(X)). \\ n(0). \\ n(s(X)) \leftarrow n(X). \end{array} \right\}$$

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 w.r.t. n is 2.

DEFINITION 4.11 (Semantic delay in Φ_P^ω)

Let P be a normal program, χ and ϕ be first-order formulas, and $\tilde{x} = \{x_1, \dots, x_k\} = FV(\chi) \cup FV(\phi)$. Suppose that $\phi \preceq_{Comp_{\mathcal{L}}(P)} \chi$.

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

Notice that since we are assuming that $\phi \preceq_{Comp_{\mathcal{L}}(P)} \chi$, if $\phi(\tilde{t}/\tilde{x})$ is *true* in some Φ_P^n , then there has to exist an integer m such that $\chi(\tilde{t}/\tilde{x})$ is *true* in Φ_P^m .

Intuitively, $\phi(\tilde{t}/\tilde{x})$ is *true* in Φ_P^n iff its truth has been proved from scratch in at most n steps. The semantic delay of χ w.r.t. 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 4.12

Let P be the following program:

$$P = \left\{ \begin{array}{l} p(0). \quad \quad \quad q(0). \\ p(s(0)). \quad \quad q(s(X)) \leftarrow q(X). \\ p(s(s(X))) \leftarrow p(X). \end{array} \right\}$$

p and q both compute natural numbers, and $p(X) \cong_{Comp_{\mathcal{L}}(P)} q(X)$, 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 $p(X)$ w.r.t.

$q(X)$ in Φ_P^ω is zero, in fact if for some ground term t and integer n , $q(t)$ is *true* (resp. *false*) in Φ_P^n , then $p(t)$ is also *true* (resp. *false*) in Φ_P^n . Vice versa, the delay of $q(X)$ w.r.t. $p(X)$ is not definable, in fact there exists no integer $m < \omega$ such that if, for some ground term t and integer n , $p(t)$ is *true* (resp. *false*) in Φ_P^n , then $q(t)$ is *true* (resp. *false*) in Φ_P^{n+m} .

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

LEMMA 4.13

If $d : A \leftarrow \tilde{L}$ is the only clause in a program P whose head unifies with an atom A , and $\tilde{w}$ is the set of variables local to the body of d , $\tilde{w} = \text{Var}(\tilde{L}) \setminus \text{Var}(A)$, then

- $A \cong_{\text{Comp}_L(P)} \exists \tilde{w} \tilde{L}$;
- the delay of A w.r.t. $\exists \tilde{w} \tilde{L}$ in Φ_P^ω is one.

PROOF. It is a straightforward application of the definition of Fitting's operator, since, by Definition 2.6, for all integers r and substitutions θ , $(\exists \tilde{w} \tilde{L})\theta$ is *true* (*false*) 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:

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

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 4.14 (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 , $\text{depen}_P(A, cl)$, 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 $C \leftarrow C_1, \dots, C_k$ in P , whose head unifies with A via mgu, say, θ , and, for some i , $\text{depen}_P(C_i, \theta, cl) = n$;
- ω when there exists no such n . In this case we say that A is *independent from* cl .

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

The following example shows how the concepts of dependency degree and semantic delay can be used to prove the safeness of the replacement operation.

EXAMPLE 4.15

Consider the following normal program:

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

where d is the only clause defining the predicate symbol p .

By Lemma 4.13, $p(X) \cong_{Comp_{\mathcal{L}}(P)} \neg q(X)$. Now, if we replace $\neg q(t)$ with $p(t)$ in cl , we obtain the following program:

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

which has the same Kunen's semantics as the previous one, that is the set of logical consequences of $Comp_{\mathcal{L}}(P)$ and of $Comp_{\mathcal{L}}(P')$ are identical. 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

- the dependency degree of p on cl (this is how big the loop would be) is greater or equal to the semantic delay of $p(X)$ w.r.t. $\neg q(X)$ (this can be seen as the 'space' where the loop would have to be introduced).

By Lemma 4.13 the delay of $p(X)$ w.r.t. $\neg q(X)$ in Φ_P^0 is one; moreover, since d is the only clause defining the predicate p and $d \neq cl$, $depen_P(p(X), cl) > 0$, thus satisfying the above conditions.

4.4 Completeness

The aim of this section is to provide a completeness result which formalizes the idea outlined in Example 4.15 and that matches with Theorem 4.7. Throughout this section we adopt Notation 4.4.

Let us first state a few simple results.

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

REMARK 4.16

Let $\tilde{L}$ be a conjunction of literals independent from the clauses $\{cl_1, \dots, cl_p\}$ in P . Let $\tilde{w} = Var(\tilde{L})$. Then, for each ordinal α ,

- $\Phi_P^\alpha \models (\neg)\exists \tilde{w} \tilde{L}$ iff $\Phi_{P'}^\alpha \models (\neg)\exists \tilde{w} \tilde{L}$.

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 $\{cl_1, \dots, cl_p\}$ is greater or equal to m , then the clauses $\{cl_1, \dots, cl_p\}$ cannot have been used in the proof of B , hence B is true in $\Phi_{P'}^m(I)$ too.

LEMMA 4.17

Let B be a ground atom, m a natural number such that

$$\text{depen}_P(B, \{cl_1, \dots, cl_p\}) \geq m$$

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 $\text{head}(c)\gamma = B$ and $\text{body}(c)\gamma$ is *true* (resp. not *false*) in $\Phi_P^{m-1}(I)$. It follows that, for each literal L belonging to $\text{body}(c)\gamma$:

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

Then, from the inductive hypothesis, each L is *true* (resp. not *false*) in $\Phi_{P'}^{m-1}(I)$.

Since $\text{depen}_P(B, \{cl_1, \dots, cl_p\}) \geq m > 0$, B does not unify with the head of any clause in $\{cl_1, \dots, cl_p\}$, that is $c \notin \{cl_1, \dots, cl_p\}$. 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 4.18

Let $\tilde{L}$ be a conjunction of literals, $\tilde{w} = \text{Var}(\tilde{L})$ and I be an $\mathcal{L}$ -interpretation. Suppose that, for some integer m , $\text{depen}_P(\tilde{L}, \{cl_1, \dots, cl_p\}) \geq m$, then,

- $\Phi_P^m(I) \models (\neg)\exists \tilde{w} \tilde{L}$ iff $\Phi_{P'}^m(I) \models (\neg)\exists \tilde{w} \tilde{L}$.

PROOF. Let $\tilde{L} = L_1, \dots, L_j$. Observe that $\text{depen}_P(\tilde{L}, \{cl_1, \dots, cl_p\}) \geq m$ implies that for $i \in [1, j]$, $\text{depen}_P(L_i, \{cl_1, \dots, cl_p\}) \geq m$.

Suppose first that $\exists \tilde{w} \tilde{L}$ is *true* in $\Phi_P^m(I)$. Then for some ground substitution θ , with $\text{Dom}(\theta) = \tilde{w}$, $\tilde{L}\theta$ 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 4.17, it is true also in $\Phi_{P'}^m(I)$. Hence the conjunction $\tilde{L}\theta$ is *true* in $\Phi_{P'}^m(I)$. It follows that $\exists \tilde{w} \tilde{L}$ is *true* in $\Phi_{P'}^m(I)$.

Now suppose that $\exists \tilde{w} \tilde{L}$ is *false* in $\Phi_P^m(I)$. Then for each ground substitution θ , with $\text{Dom}(\theta) = \tilde{w}$, $\tilde{L}\theta$ 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 4.17 $L_i\theta$ is also *false* in $\Phi_{P'}^m(I)$. Hence $\tilde{L}\theta$ is *false* in $\Phi_{P'}^m(I)$. It follows that $\exists \tilde{w} \tilde{L}$ is *false* in $\Phi_{P'}^m(I)$. ■

We can now state the completeness result. As before, we refer to Notation 4.4.

Recall that, when replacing $\tilde{C}$ with $\tilde{D}$, in order to prove the partial correctness of the replacement operation, we required that $\exists \tilde{x} \tilde{D} \preceq_{\text{Comp}_{\mathcal{L}}(P)} \exists \tilde{x} \tilde{C}$, where $\tilde{x}$ is a set of variables satisfying the locality property. It should be no surprise that to prove the completeness of the operation we have to require the opposite side of the equivalence, namely that $\exists \tilde{x} \tilde{C} \preceq_{\text{Comp}_{\mathcal{L}}(P)} \exists \tilde{x} \tilde{D}$.

THEOREM 4.19 (Completeness)

If for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, there exists a (possibly empty) set of variables $\tilde{x}_i$ satisfying the locality property w.r.t. $\tilde{C}_i$ and $\tilde{D}_i$ such that

$$\exists \tilde{x}_i \tilde{C}_i \preceq_{Comp_L(P)} \exists \tilde{x}_i \tilde{D}_i,$$

and if one of the following two conditions holds:

- (a) $\{\tilde{D}_1, \dots, \tilde{D}_n\}$ are all independent from the clauses $\{cl_1, \dots, cl_p\}$; or
- (b) there exists an integer m such that, for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, and each $cl_j \in \{cl_1, \dots, cl_p\}$:
 - the delay of $\exists \tilde{x}_i \tilde{D}_i$ w.r.t. $\exists \tilde{x}_i \tilde{C}_i$ in Φ_P^ω is less or equal to m , and
 - $depend_P(\tilde{D}_i, cl_j) \geq m$;

then the simultaneous replacement operation is complete.

PROOF. First we need to establish a lemma similar to the one in the proof of Theorem 4.7.

LEMMA 4.20

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 $\tilde{C}_j \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$ and a ground substitution θ such that:

- either $I \models \exists \tilde{x}_j \tilde{C}_j \theta$, while $I' \not\models \exists \tilde{x}_j \tilde{D}_j \theta$;
- or $I \models \neg \exists \tilde{x}_j \tilde{C}_j \theta$, while $I' \models \neg \exists \tilde{x}_j \tilde{D}_j \theta$.

PROOF. The proof is identical to the one given in the Appendix A for Lemma 4.8 in Theorem 4.7, and it is omitted. ■

Again the proof of the theorem is by contradiction. By Theorems 3.3 and 3.4 the operation is complete iff $\forall n \exists m \Phi_P^n \subseteq \Phi_{P'}^m$, so let us suppose that there exist two integers i and j such that

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

Since $\Phi_P^{j+1} = \Phi_P(\Phi_P^j)$, from Lemma 4.20 we have that for each integer l there exists an integer $j(l) \in \{1, \dots, n\}$ and a ground substitution θ_l such that

$$\begin{aligned} \exists \tilde{x}_{j(l)} \tilde{C}_{j(l)} \theta_l \text{ is true (or false) in } \Phi_P^j, \text{ while} \\ \exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \theta_l \text{ is not true (resp. not false) in } \Phi_{P'}^{i+l}. \end{aligned} \tag{4.5}$$

Let us distinguish two cases.

(1) Hypothesis (a) is satisfied and each conjunction in $\{\tilde{D}_1, \dots, \tilde{D}_n\}$ is independent from $\{cl_1, \dots, cl_p\}$. By hypothesis $\exists \tilde{x}_i \tilde{C}_i \preceq_{Comp_L(P)} \exists \tilde{x}_i \tilde{D}_i$, we can then apply Lemma 4.6 to the left-hand side of (4.5); it follows that there has to be an integer r such that for each l ,

$$\exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \theta_l \text{ is true (resp. false) in } \Phi_{P'}^r.$$

From Remark 4.16, it follows that for each integer l , $\exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \theta_l$ is true (resp. false) in $\Phi_{P'}^r$.

This contradicts (4.5); 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 $\exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \theta_l$ is *true* (resp. *false*) in $\Phi_{P'}^r$, it must be *true* (resp. *false*) in $\Phi_{P'}^{i+l}$.

(2) Hypothesis (b) is satisfied. We know that for each integer l , the delay of $\exists \tilde{x}_{j(l)} \tilde{D}_{j(l)}$ w.r.t. $\exists \tilde{x}_{j(l)} \tilde{C}_{j(l)}$ is not greater than m , hence from the left-hand side of (4.5) it follows that,

$$\text{for each } l, \exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \theta_l \text{ is true or false in } \Phi_P^{j+m}.$$

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

$$\text{for each } l, \exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \theta_l \text{ is true (resp. false) in } \Phi_P^m(\Phi_P^j).$$

$\text{depen}_P(\tilde{D}_{j(l)} \theta_l, \{cl_1, \dots, cl_p\}) \geq m$, then, from Lemma 4.18 it follows that,

$$\text{for each } l, \exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \theta_l \text{ is true (resp. false) in } \Phi_P^m(\Phi_P^j).$$

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

$$\text{for each } l, \exists \tilde{x}_{j(l)} \tilde{D}_{j(l)} \theta_l \text{ is true (resp. false) in } \Phi_{P'}^m(\Phi_{P'}^i) = \Phi_{P'}^{m+i},$$

this contradicts the right-hand side of (4.5). ■

Finally, from Theorems 4.7 and 4.19 we obtain the following safeness result for the replacement operation.

COROLLARY 4.21 (applicability conditions for the replacement operation)

Using Notation 4.4, if for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, there exists a (possibly empty) set of variables $\tilde{x}_i$ satisfying the locality property w.r.t. $\tilde{C}_i$ and $\tilde{D}_i$ such that

$$\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_L(P)} \exists \tilde{x}_i \tilde{C}_i$$

and one of the following two conditions holds:

1. $\{\tilde{D}_1, \dots, \tilde{D}_n\}$ are all independent from the clauses in $\{cl_1, \dots, cl_p\}$; or
2. there exists an integer m such that, for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, and each $cl_j \in \{cl_1, \dots, cl_p\}$:
 - the delay of $\exists \tilde{x}_i \tilde{D}_i$ w.r.t. $\exists \tilde{x}_i \tilde{C}_i$ in Φ_P^0 is less or equal to m , and
 - $\text{depen}_P(\tilde{D}_i, cl_j) \geq m$;

then the simultaneous replacement operation is safe, that is P is *equivalent* to P' . □

Conditions 1 and 2 reflect two different ways in which we can guarantee that we are not introducing dangerous loops. Condition 2 is automatically satisfied when, for each i , the semantic delay of $\exists \tilde{x}_i \tilde{D}_i$ w.r.t. $\exists \tilde{x}_i \tilde{C}_i$ in Φ_P^0 is zero. This is probably the most interesting situation in which it can be applied. Recall that the semantic delay of $\exists \tilde{x}_i \tilde{D}_i$ w.r.t. $\exists \tilde{x}_i \tilde{C}_i$ shows (for each θ) how many steps later than $\exists \tilde{x}_i \tilde{C}_i \theta$, we determine the truth value of $\exists \tilde{x}_i \tilde{D}_i \theta$ (at worse). Therefore, when the delay is zero, we can determine the truth value of $\exists \tilde{x}_i \tilde{D}_i \theta$ 'faster' than the truth value $\exists \tilde{x}_i \tilde{C}_i \theta$. By stretching the notation we could say that in this case $\exists \tilde{x}_i \tilde{D}_i$ is 'more efficient' than $\exists \tilde{x}_i \tilde{C}_i$. By the above Corollary we have that if the replacing conjunctions are 'equivalent to' and 'more efficient than' the replaced ones, then the replacement is safe. This fits well in a context where transformation operations are intended

for increasing the performances of programs. Of course here we are referring to a bottom-up way of determining truth values, while most resolutions methods employ a top-down search, hence what is considered 'more efficient' here may not necessarily be 'more efficient' when we actually run the program.

Other semantics

Corollary 4.21 can easily be applied to other declarative semantics. Basically what we need is a definition of equivalence and semantic delay: any model theoretic semantics which can be defined in terms of the Kleene sequence of some operator is potentially suitable. For example the well-founded semantics is appropriate, while the two-valued completion semantics (considered in [12]) is not, as it lacks a constructive definition. Of course, when we change the semantics we refer to, the concept of equivalence of programs and formulas can differ significantly.

Let us for example consider the S-semantics [10], a model theoretic reconstruction of the computed answer semantics¹. The S-semantics does not take into consideration the negative information that can be inferred from (the completion of) a program. This influences significantly the applicability conditions of replacement. Consider for instance the following program:

$$P = \{cl : p \leftarrow q, p.\}$$

q has no definition and therefore it fails. If we eliminate q from the body of cl , we obtain

$$P' = \{cl : p \leftarrow p.\}$$

The S-semantics (as well as the least Herbrand model semantics) of P and P' coincide (they are both empty as both p and q do not succeed in either program), so this transformation is (S-)safe. Now let us show how the S-correspondent of Corollary 4.21 can be applied to this situation: the transformation of P into P' can be seen as a replacement of q, p with p in the body of cl , and we have that

- q, p is equivalent to p in the S-semantics of P (neither succeeds),
- the delay of p w.r.t. q, p in $T_S^\omega(P)$ ² is zero,
- $depen_P(p, cl) = 0$.

Hence the applicability conditions for the S-version of Corollary 4.21 are satisfied.

Now, if we switch back to Kunen's semantics, P is no longer equivalent to P' , in fact, $Comp_L(P) \models \neg p$ while $Comp_L(P') \not\models \neg p$. In the transformation we have lost some negative information, the replacement is therefore not (Kunen-)safe. Indeed, the applicability conditions of Corollary 4.21 are not satisfied as

- $q, p \cong_{Comp_L(P)} p$,
- the delay of p w.r.t. q, p in Φ_P^ω is **one**. ($\Phi_P^1 \models \neg(q, p)$, while $\Phi_P^2 \models \neg p$),
- $depen_P(p, cl) = 0$.

Here the delay of p w.r.t. q, p is greater than $depen_P(p, cl)$ and consequently Corollary 4.21 is no longer applicable. This is due to the fact that, since we are now taking into account also the negative information, the delay of p w.r.t. q, p is no longer zero.

However, there exists a semantics, the well-founded semantics, that does take into consideration negative information, but for which the above programs P and P' are nevertheless

¹ A result similar to Corollary 4.21 for the S-semantics is given in [5]

² $T_S(P)$ is the S-semantics counterpart of Φ_P

equivalent. Loosely speaking, the well-founded semantics does not distinguish *finite* from *infinite* failure. So the query $\leftarrow p$ fails both in P (finitely) and in P' (infinitely). The authors have also stated a counterpart of Corollary 4.21 for this semantics [9]. It can be applied to the transformation performed above: we have that q, p is equivalent to p and that the delay of p w.r.t. q, p is zero. The applicability conditions for the replacement operation are then, in this context, satisfied.

4.5 Checking applicability conditions

Determining whether two conjunctions of literals are equivalent is in general an undecidable problem, moreover, the *semantic delay* is not a computable function, and for this reason Corollary 4.21 must be regarded as a theoretical result. It is therefore important to single out some situations in which its hypothesis can be guaranteed either by a syntactic check or, when the replacement belongs to a transformation sequence, by the previous *history* of the transformation. This section shows some of these situations. Later, in Section 6 we also show an example of a transformation sequence in which the conditions of Corollary 4.21 are checked by hand. We hope that this provides a better understanding of the concepts we use.

Reversible folding

We now show how Corollary 4.21 can be used to prove the correctness of the reversible folding operation, which is the kind of folding operation studied in [16, 12]. First of all let us state its definition.

DEFINITION 4.22 (reversible folding)

Let $cl : A \leftarrow \tilde{B}', \tilde{S}$, and $d : H \leftarrow \tilde{B}$ be distinct clauses in a program P ; let also $\tilde{w}$ be the set of local variables of d , $\tilde{w} = Var(\tilde{B}) \setminus Var(H)$. If there exists a substitution θ , $Dom(\theta) = Var(d)$ such that

- (i) $\tilde{B}' = \tilde{B}\theta$;
- (ii) θ does not bind the local variables of d , that is for any $x, y \in \tilde{w}$ the following three conditions hold
 - $x\theta$ is a variable;
 - $x\theta$ does not appear in $A, \tilde{S}, H\theta$;
 - if $x \neq y$ then $x\theta \neq y\theta$;
- (iii) d is the only clause of P whose head unifies with $H\theta$;

then we can fold $H\theta$ in cl , obtaining $cl' : A \leftarrow H\theta, \tilde{S}$.

EXAMPLE 4.23

Let us consider the following program:

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

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:

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

This operation can be seen as a special case of replacement in which the conditions of Corollaries 4.21 are always satisfied. First of all notice that, by using the notation of Definition 4.22, the operation reduces to a replacement of $\tilde{B}'$ with $H\theta$. Now by the conditions on folding (i). . . (iii) and Lemma 4.13, we have that

- $\tilde{w}$ satisfies the locality property w.r.t. $\tilde{B}'$ and H (recall that $\tilde{w}$ is the set of local variables of d);

- $H\theta$ is equivalent to $\exists \tilde{w}\theta \tilde{B}'$ (Lemma 4.13);

- the delay of $H\theta$ w.r.t. $\exists \tilde{w}\theta \tilde{B}'$ in Φ_P^ω is one, (Lemma 4.13).

Finally, from (iii) we also have that the dependency degree of $\text{depend}_P(H\theta, cl) > 0$.

Hence, the applicability conditions of Corollary 4.21 are satisfied and the operation is safe.

Recursive folding

The reversible folding operation is a rather restrictive kind of folding, in particular it lacks the possibility of introducing recursion in the definition of predicates. This can be done via an *unfold/fold transformation sequence*. Unfold/fold transformation sequences were introduced in the area of logic programming by Tamaki and Sato [20] and, as a large literature shows, proved to be an effective methodology for program's development and optimization.

The following example shows how this kind of folding can be used for introducing recursion in definitions.

First 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 4.24 (unfold)

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

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

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

This operation is safe w.r.t. all the semantics we consider in this paper, the proof can be found in [6].

EXAMPLE 4.25

We start with the following program where *initial* defines the property of being a prefix of a list.

$$P_0 = \left\{ \begin{array}{l} d : \text{initial}(Xs, Zs) \quad \leftarrow \text{append}(Xs, Ys, Zs). \\ \text{append}([A|Xs], Ys, [A|Zs]) \quad \leftarrow \text{append}(Xs, Ys, Zs). \\ \text{append}([], Ys, Ys). \end{array} \right\}$$

We now unfold the body of the first clause, obtaining the two clauses

$$P_1 = \left\{ \begin{array}{l} cl : \text{initial}([A|Xs], [A|Zs]) \quad \leftarrow \text{append}(Xs, Ys, Zs). \\ \text{initial}([], Zs). \\ \dots \end{array} \right. \text{together with the clauses defining } \text{append} \left. \right\}$$

Now we can fold $\text{append}(Xs, Ys, Zs)$ in the body of the first clause, using d as folding clause. We obtain

$$P_2 = \left\{ \begin{array}{l} cl' : \text{initial}([A|Xs], [A|Zs]) \quad \leftarrow \text{initial}(Xs, Zs) \\ \text{initial}([], Zs). \\ \dots \end{array} \right. \text{together with the clauses defining } \text{append} \left. \right\}$$

The predicate *initial* has now a recursive definition.

Notice that the folding operation of the above example can be seen as a replacement of $\text{append}(Xs, Ys, Zs)$ with $\text{initial}(Xs, Zs)$, and also in this case the applicability conditions of Corollary 4.21 are satisfied, in fact we have that:

- Ys satisfies the locality property w.r.t. $\text{append}(Xs, Ys, Zs)$ and $\text{initial}(Xs, Zs)$ in P_1 ;
- $\text{initial}(Xs, Zs) \cong_{\text{Comp}_{\mathcal{L}}(P_1)} \exists Ys \text{ append}(Xs, Ys, Zs)$;
- the delay of $\text{initial}(Xs, Zs)$ w.r.t. $\exists Ys \text{ append}(Xs, Ys, Zs)$ in P_1 is zero.

The last two statements are due to the following general result which is stated in [5]:

OBSERVATION 4.26

Let $H \leftarrow \tilde{B}$ be a non-recursive clause in a program P and, $\tilde{w}$ be its set of local variables $\tilde{w} = \text{Var}(\tilde{B}) \setminus \text{Var}(H)$. If P' is a program obtained from P by unfolding all the atoms in $\tilde{B}$ then $H \cong_{\text{Comp}_{\mathcal{L}}(P')} \exists \tilde{w} B$, and the delay of H w.r.t. $\exists \tilde{w} B$ in P' is zero.

This provides a further example of the kind of situations to which Corollary 4.21 can be applied. Actually, in [7] the authors prove a correctness result of an unfold/fold transformation sequence using the above observation and Fitting's counterpart of Corollary 4.21, Corollary 5.12.

5 Adopting a (possibly) finite language

Our aim now is to analyse how the results given in the previous two sections have to be modified when the language adopted is no longer infinite (or at least not necessarily infinite). Therefore in the sequel we still refer to a fixed but unspecified language $\mathcal{L}$, but we no longer assume it to be infinite.

As we mentioned in Section 2, the main problem we have to face when adopting a finite language is that $\text{CET}_{\mathcal{L}}$ becomes an incomplete theory. The consequences of this are best shown by the following example, which is borrowed from [19]. Let P be the program:

$$P = \left\{ \begin{array}{l} p \leftarrow \neg q(X). \\ q(a). \end{array} \right\}$$

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 here is that we do not have a ‘witness’ that allows us to say that $\exists X X \neq a$ holds, nor can we formally infer that such a witness does not exist. The two main approaches used in logic programming in order to obtain a complete theory out of $CET_{\mathcal{L}}$ are the following:

- adopting an infinite language (that is a language with infinitely many functions symbols, and that consequently contains infinitely many ‘witnesses’);
- adopting a finite language together with some domain closure axioms, which are axioms that commit us to a specific universe.

For an extended discussion of the subject, we refer the reader to [19].

As we mentioned before, in the literature we find two different kinds of domain closure axioms.

DEFINITION 5.1

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

- The *Domain Closure Axiom*, $DCA_{\mathcal{L}}$, is

$$x = t_1 \vee x = t_2 \vee \dots$$

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

- The *Weak Domain Closure Axiom*, $WDCA_{\mathcal{L}}$, is

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

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

Note that when $\mathcal{L}$ contains a function of arity greater than zero, $DCA_{\mathcal{L}}$ is an infinite disjunction and hence it is not a first-order formula. For this reason, the notation $Comp_{\mathcal{L}}(P) \cup DCA_{\mathcal{L}}$, that we are going to use often in the sequel is actually overloaded, nevertheless we shall use it for uniformity with the rest of the paper. As opposed to $DCA_{\mathcal{L}}$, $WDCA_{\mathcal{L}}$ is a first-order formula.

The following simple example shows how the semantics of a program changes depending on the kind of closure axioms adopted.

EXAMPLE 5.2

Let P be the same program we used in Example 2.7.

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

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 $\text{CET}_{\mathcal{L}}$.

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

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

In fact assuming $\text{DCA}_{\mathcal{L}}$ is equivalent to restrict ourselves to $\mathcal{L}$ -Herbrand interpretations and models, and the formula $\forall x n(x)$ is *true* in the unique Herbrand model of P . From this it follows that

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

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

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

In fact $\text{WDCA}_{\mathcal{L}}$ allows a model which contains, besides the natural numbers, also an infinite chain of 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

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

By assuming $\text{WDCA}_{\mathcal{L}}$ we obtain a semantics which is stronger than the one adopting $\text{DCA}_{\mathcal{L}}$. In fact $\text{DCA}_{\mathcal{L}} \models \text{WDCA}_{\mathcal{L}}$, and hence if $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}} \models \phi$, then also $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \phi$.

It is important to observe that we have to modify the definitions of program equivalence (3.1), of formulas equivalence (4.2) and of correctness of a transformation (4.1) according to the domain closure axioms we adopt.

Let us now give another example showing how program's equivalence may be affected by the choices of the language and of the closure axioms.

EXAMPLE 5.3

Consider the three programs:

$$P_1 = \{ \begin{array}{l} n(0). \\ n(s(X)) \quad \leftarrow n(X). \end{array} \}$$

$$P_2 = \{ \begin{array}{l} n(0). \\ n(s(X)). \end{array} \}$$

$$P_3 = \{ \begin{array}{l} n(X). \end{array} \}$$

Let $\mathcal{L} = \mathcal{L}(P_1)$. If we assume $\text{DCA}_{\mathcal{L}}$, for all three programs we have

$$\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \forall x n(x), \quad P \in \{P_1, P_2, P_3\}.$$

Actually, all the programs are pairwise *equivalent* w.r.t. this semantics.

If we assume $\text{WDCA}_{\mathcal{L}}$,

$$\text{Comp}_{\mathcal{L}}(P_1) \cup \text{WDCA}_{\mathcal{L}} \not\models \forall x n(x),$$

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

$$\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}} \models \forall x n(x), \tag{5.1}$$

then only P_2 and P_3 are *equivalent* w.r.t. this semantics.

Finally if we assume that $\mathcal{L}$ strictly contains $\mathcal{L}(P_1)$, then P_3 is the only program for which (5.1) holds. In this case no program is equivalent to any of the other ones, no matter which axioms we adopt.

This example shows that two programs may be equivalent w.r.t. $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$ and not equivalent w.r.t. $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$. But there are also cases in which the converse of this statement is true. So even though the semantics obtained by assuming $\text{WDCA}_{\mathcal{L}}$ is stronger than the one obtained by assuming $\text{DCA}_{\mathcal{L}}$, no program equivalence is stronger than the other.

5.1 Correctness results w.r.t. $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$

As far as we are concerned the semantics given by $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$ (with $\mathcal{L}$ possibly finite) behaves exactly as Kunen's semantics. This fact is due to the following result.

THEOREM 5.4 ([19])

Let P be a normal program, $\mathcal{L}$ a finite language and ϕ an allowed formula

- $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}} \models \phi$ iff for some integer n , $\Phi_P^n \models \phi$.

Here $\mathcal{L}$ is required to be finite uniquely because otherwise $\text{WDCA}_{\mathcal{L}}$ is not a first-order formula. Notice that Theorem 5.4 is identical to Theorem 3.2, 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. In particular, Theorems 3.3 and 3.4 and Corollaries 3.5 and 4.9 hold also for $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$. The same reasoning applies to Lemma 4.6 on the equivalence of formulas. Finally, the results on the replacement operation, that is Theorems 4.7, 4.19 and Corollary 4.21 hold also for this semantics. Let us now restate this corollary.

COROLLARY 5.5 (Applicability conditions w.r.t. $\text{Comp}_{\mathcal{L}} \cup \text{WDCA}_{\mathcal{L}}$)

Using Notation 4.4, if for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, there exists a (possibly empty) set of variables $\tilde{x}_i$ satisfying the locality property w.r.t. $\tilde{C}_i$ and $\tilde{D}_i$ such that

$$\exists \tilde{x}_i \tilde{D}_i \text{ is equivalent to } \exists \tilde{x}_i \tilde{C}_i \text{ w.r.t. } \text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}},$$

and one of the following two conditions holds:

1. $\{\tilde{D}_1, \dots, \tilde{D}_n\}$ are all independent from the clauses in $\{cl_1, \dots, cl_p\}$; or
2. there exists an integer m such that, for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, and each $cl_j \in \{cl_1, \dots, cl_p\}$:
 - the delay of $\exists \tilde{x}_i \tilde{D}_i$ w.r.t. $\exists \tilde{x}_i \tilde{C}_i$ in Φ_P^n is less or equal to m , and
 - $\text{depen}_P(D_i, cl_j) \geq m$;

then the simultaneous replacement operation is safe, that is P is equivalent to P' (w.r.t. $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$).

5.2 Correctness results w.r.t. $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$

In this section we refer to the semantics given by $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$. As opposed to what happened in the previous section, there is no point in requiring $\mathcal{L}$ to be a finite language. Since

$\text{DCA}_{\mathcal{L}}$ is (usually) already a non-first-order axiom, we have to leave the first-order context anyhow, and there is no reason here in restricting the domain.

As we said before, adopting $\text{DCA}_{\mathcal{L}}$ is equivalent to restricting our attention to Herbrand interpretations and models (on the language $\mathcal{L}$). This particular semantics enjoys a remarkable property: namely that there always exists a *minimal* Herbrand model (w.r.t. $\subseteq$), and this coincides with the interpretation given by the least fixpoint of the operator $\Phi_P, \text{lfp}(\Phi_P)$.

THEOREM 5.6 ([11])

Let P be a normal program and ϕ an allowed formula

- $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \phi$ iff $\text{lfp}(\Phi_P) \models \phi$.

To check if an allowed formula is a logical consequence of $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$ it is sufficient to check if it is true in $\text{lfp}(\Phi_P)$. If $\mathcal{L} = \mathcal{L}(P)$, this semantics is called Fitting's model semantics [11]. Moreover, since Φ_P is a monotonic operator, we also have that, for some ordinal α , $\text{lfp}(\Phi_P) = \Phi_P^\alpha$, however, Φ_P not being continuous, α could be greater than ω .

Using Theorem 5.6 we can easily characterize the correctness of the transformation w.r.t. to this semantics by referring to the least fixed point of the Φ_P operator.

LEMMA 5.7

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

- partially correct iff $\text{lfp}(\Phi_P) \supseteq \text{lfp}(\Phi_{P'})$;
- complete iff $\text{lfp}(\Phi_P) \subseteq \text{lfp}(\Phi_{P'})$;
- totally correct (safe) iff $\text{lfp}(\Phi_P) = \text{lfp}(\Phi_{P'})$.

Partial correctness

We now consider the problem of proving partial correctness of the replacement operation. When we replace the conjunction $\tilde{C}$ with $\tilde{D}$, the first natural requirement we ask for is the equivalence of $\tilde{C}$ and $\tilde{D}$ w.r.t. $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$.

Here again we need Theorem 5.6 in order to give a characterization of the equivalence of formulas w.r.t. $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$. First we introduce the three-valued operator $\Rightarrow$, which is 'one side' of $\Leftrightarrow$ and it is defined as follows: $\phi \Rightarrow \chi$ is *true* iff ϕ is less specific than χ , 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*.

LEMMA 5.8

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

- (a) $\chi \preceq_{\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}} \phi$;
- (b) $\text{lfp}(\Phi_P) \models \chi \Rightarrow \phi$.

PROOF. The proof is given in Appendix A. ■

Statement (b) differs from the corresponding one of Lemma 4.6. In Lemma 4.6 we were considering the completion with an infinite language, which as far as this lemma is concerned, is equivalent to assuming a finite language and $\text{WDCA}_{\mathcal{L}}$. In such cases the universe of a model of $\text{Comp}_{\mathcal{L}}(P)$ 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 alone is not sufficient to ensure the equivalence between χ and ϕ .

For example, if we consider the following program where, for simplicity, we refer to $\text{WDCA}_{\mathcal{L}}$:

$$P = \left\{ \begin{array}{l} n(0). \\ n(s(X)) \leftarrow n(X). \\ m(X). \end{array} \right\}$$

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 $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$, but $n(X) \not\equiv_{\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}} m(X)$. In fact, let $\zeta \equiv \forall x m(x)$, then $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}} \models \zeta$, while $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}} \not\models \zeta[n(x)/m(x)]$ (see Example 5.3). Indeed $m(X)$ and $n(X)$ must not be considered equivalent w.r.t.

$\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$, in fact if we consider the following extension to program P :

$$P_1 = P \cup \left\{ \begin{array}{l} q_1 \leftarrow \neg n(X). \\ q_2 \leftarrow \neg m(X). \end{array} \right\}$$

and $\mathcal{L} = \mathcal{L}(P_1)$, $n(X)$ is equivalent to $m(X)$ while q_1 is not equivalent to q_2 .

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

THEOREM 5.9 (partial correctness)

Let us adopt Notation 4.4, if for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, there exists a (possibly empty) set of variables $\tilde{x}_i$ satisfying the locality property w.r.t. $\tilde{C}_i$ and $\tilde{D}_i$ such that

$$\exists \tilde{x}_i \tilde{D}_i \preceq_{\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}} \exists \tilde{x}_i \tilde{C}_i$$

then the simultaneous replacement operation is partially correct.

PROOF. The proof is by contradiction. By Lemma 5.7, we have that the operation is partially correct iff $\text{lfp}(\Phi_P) \supseteq \text{lfp}(\Phi_{P'})$, so let us suppose $\text{lfp}(\Phi_P) \not\supseteq \text{lfp}(\Phi_{P'})$. Since the sequence $\Phi_{P'}^0, \Phi_{P'}^1, \dots$ is monotonically increasing and $\Phi_{P'}^0 = (\emptyset, \emptyset) \subseteq \text{lfp}(\Phi_P)$, there has to be an ordinal α such that

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

Hence $\text{lfp}(\Phi_P) \not\supseteq \Phi_{P'}(\text{lfp}(\Phi_P))$ and $\Phi_{P'}(\text{lfp}(\Phi_P)) \supseteq \Phi_{P'}(\Phi_{P'}^\alpha)$, since Φ is monotone.

Since $\Phi_P(\text{lfp}(\Phi_P)) = \text{lfp}(\Phi_P)$ we have that

$$\Phi_P(\text{lfp}(\Phi_P)) \not\supseteq \Phi_{P'}(\text{lfp}(\Phi_P)). \quad (5.2)$$

From Lemma 4.8 and (5.2) it follows that there exists an integer j and a ground substitution θ such that $\exists \tilde{x}_j \tilde{D}_j \theta$ is *true* (or *false*) in $\text{lfp}(\Phi_P)$, while $\exists \tilde{x}_j \tilde{C}_j \theta$ is not. This, by Lemma 5.8, contradicts the hypothesis. $\blacksquare$

As happened with Theorem 4.7, this result brings us to a first completeness result: with the notation of the previous theorem, if for each i we also have that $\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}} \exists \tilde{x}_i \tilde{C}_i$, then the transformation is safe iff for each i , $\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_{\mathcal{L}}(P') \cup \text{DCA}_{\mathcal{L}}} \exists \tilde{x}_i \tilde{C}_i$. The proof is identical to the one given for Corollary 4.9.

Completeness

We want a completeness result which matches with Theorem 4.19. First of all we need a slightly stronger definition of semantic delay.

DEFINITION 5.10 (Semantic delay in $lfp(\Phi_P)$)

Let P be a normal program, χ and ϕ be first-order formulas, and $\tilde{x} = \{x_1, \dots, x_k\} = FV(\chi) \cup FV(\phi)$. Suppose that $\phi \preceq_{Comp_L(P) \cup DCA_L} \chi$.

- The semantic delay of χ w.r.t. 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 $\Phi_P^\alpha \models (\neg)\phi(\tilde{t}/\tilde{x})$, then $\Phi_P^{\alpha+k} \models (\neg)\chi(\tilde{t}/\tilde{x})$.

Unsurprisingly, the difference between this definition and the one of semantic delay in Φ_P^ω (4.11) is that here we also have to consider ordinals which are greater than ω .

Now we can prove the completeness result in this case.

THEOREM 5.11 (completeness)

In the hypothesis of 4.4, if for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, there exists a (possibly empty) set of variables $\tilde{x}_i$ satisfying the locality property w.r.t. $\tilde{C}_i$ and $\tilde{D}_i$ such that

$$\exists \tilde{x}_i \tilde{C}_i \preceq_{Comp_L(P) \cup DCA_L} \exists \tilde{x}_i \tilde{D}_i,$$

and if one of the following two conditions holds:

- (a) $\{\tilde{D}_1, \dots, \tilde{D}_n\}$ are all independent from the clauses $\{cl_1, \dots, cl_p\}$; or
- (b) there exists an integer m such that, for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, and each $cl_j \in \{cl_1, \dots, cl_p\}$:
 - the delay of $\exists \tilde{x}_i \tilde{D}_i$ w.r.t. $\exists \tilde{x}_i \tilde{C}_i$ in $lfp(\Phi_P)$ is less or equal to m , and
 - $depen_P(\tilde{D}_i, cl_j) \geq m$;

then the simultaneous replacement operation is complete.

PROOF. The proof is by contradiction. By Lemma 5.7 the operation is complete iff $lfp(\Phi_P) \subseteq lfp(\Phi_{P'})$, so let us suppose that $lfp(\Phi_P) \not\subseteq lfp(\Phi_{P'})$. By the same argument used in the proof of Theorem 5.9, 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 Lemma 4.20 there exists an integer j and a ground substitution θ such that

$$\begin{aligned} \exists \tilde{x}_j \tilde{C}_j \theta \text{ is true (or false) in } \Phi_P^\alpha, \text{ while} \\ \exists \tilde{x}_j \tilde{D}_j \theta \text{ is not true (resp. not false) in } lfp(\Phi_{P'}). \end{aligned} \tag{5.3}$$

Let us distinguish two cases.

(1) Condition (a) of the hypothesis applies, and $\tilde{D}_j$ is independent from $\{cl_1, \dots, cl_p\}$. Since $\Phi_P^\alpha \subseteq lfp(\Phi_P)$, from the left-hand side of (5.3), $\exists \tilde{x}_j \tilde{C}_j \theta$ is also true (resp. false) in $lfp(\Phi_P)$. Hence, by the hypothesis and Lemma 5.8, also $\exists \tilde{x}_j \tilde{D}_j \theta$ is true (resp. false) in $lfp(\Phi_P)$. Because of condition (a) and Remark 4.16, $\exists \tilde{x}_j \tilde{D}_j \theta$ is true (resp. false) in $lfp(\Phi_{P'})$. This contradicts the left-hand side of (5.3).

(2) Condition (b) of the hypothesis applies. The delay of $\exists \tilde{x}_j \tilde{D}_j$ w.r.t. $\exists \tilde{x}_j \tilde{C}_j$ is not greater than m , hence from the left-hand side of (5.3) it follows that $\exists \tilde{x}_j \tilde{D}_j \theta$ is *true* (or *false*) in $\Phi_P^{\alpha+m}$, that is, $\exists \tilde{x}_j \tilde{D}_j \theta$ is *true* (resp. *false*) in $\Phi_P^m(\Phi_P^\alpha)$.

Since by (b), $\text{depen}_P(\tilde{D}_j \theta, \{cl_1, \dots, cl_p\}) \geq m$, from Lemma 4.18 it follows that

$$\exists \tilde{x}_j \tilde{D}_j \theta \text{ is true (resp. false) in } \Phi_{P'}^m(\Phi_P^\alpha).$$

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

$$\exists \tilde{x}_j \tilde{D}_j \theta \text{ is true (resp. false) in } \Phi_{P'}^m(\text{lfp}(\Phi_{P'})).$$

But since $\Phi_{P'}^m(\text{lfp}(\Phi_{P'})) = \text{lfp}(\Phi_{P'})$, this contradicts the right-hand side of (5.3). ■

Finally, from Theorems 5.9 and 5.11 we obtain the following result on the safeness of the replacement operation.

COROLLARY 5.12 (Applicability conditions w.r.t. $\text{Comp}_{\mathcal{L}} \cup \text{DCA}_{\mathcal{L}}$ with $\mathcal{L}$ finite)
In the hypothesis of 4.4, if for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, there exists a (possibly empty) set of variables $\tilde{x}_i$ satisfying the locality property w.r.t. $\tilde{C}_i$ and $\tilde{D}_i$ such that

$$\exists \tilde{x}_i \tilde{D}_i \cong_{\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}} \exists \tilde{x}_i \tilde{C}_i$$

and one of the following two conditions holds:

1. $\{\tilde{D}_1, \dots, \tilde{D}_n\}$ are all independent from the clauses in $\{cl_1, \dots, cl_p\}$; or
2. there exists an integer m such that, for each $\tilde{C}_i \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$, and each $cl_j \in \{cl_1, \dots, cl_p\}$:
 - the delay of $\exists \tilde{x}_i \tilde{D}_i$ w.r.t. $\exists \tilde{x}_i \tilde{C}_i$ in $\text{lfp}(\Phi_P)$ is less or equal to m , and
 - $\text{depen}_P(D_i, cl_j) \geq m$;

then the simultaneous replacement operation is safe, that is, P is equivalent to P' (w.r.t. $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$).

6 Replacement versus other operations

In this section we consider the operations of thinning and fattening, and show how they can be seen as particular cases of replacement. We introduce them by means of an example of transformation sequence. This also give us the opportunity of illustrating how the applicability conditions for the replacement operation can be checked ‘by hand’.

For the sake of simplicity, we consider the semantics given by $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$. The results hold also in the case we adopt $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$ (and therefore also for Kunen’s semantics) although the proofs are then more complicated.

EXAMPLE 6.1 (Sorting by permutation and check, part I)

The following program is borrowed from [20]. The transformation process is intentionally redundant in order to be more explanatory.

Let P_0 be the following program:

- $\exists \tilde{x} \tilde{G} \preceq_{\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}} \exists \tilde{x} \tilde{G}, \tilde{H}$,

- the delay of $\exists \tilde{x} \tilde{G}, \tilde{H}$ w.r.t. $\exists \tilde{x} \tilde{G}$ in $\text{lfp}(\Phi_P)$ is less or equal to m .

If m is the least of such integers, then the delay of $\exists \tilde{x} \tilde{G}, \tilde{H}$ w.r.t. $\exists \tilde{x} \tilde{G}$ in $\text{lfp}(\Phi_P)$ is exactly m .

PROOF. It is a straightforward application of Theorem 5.6 together with the fact that if $\tilde{G}\theta$ is *false* in some interpretation I , then also $(\tilde{G}, \tilde{H})\theta$ is *false* in I . ■

This lemma also applies to the semantics given by $\text{Comp}_{\mathcal{L}}(P) \cup \text{WDCA}_{\mathcal{L}}$, as is shown by Lemma B.1 in Appendix B.

EXAMPLE 6.4 (Sorting by permutation and check, part II)

(3) Now we can fatten clause $c10$ by adding $\text{ord}(Zs)$ to its body.

Let P_3 be the resulting program:

$P_3 = \{c1, \dots, c7\} \cup$

$\{ \begin{array}{l} c11 : \text{sort}([], []). \\ c12 : \text{sort}([A \mid Xs], Ys) \leftarrow \text{perm}(Xs, Zs), \text{ord}(Zs), \text{ins}(A, Zs, Ys), \text{ord}(Ys). \end{array} \}$

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

We now use Theorem 5.11 to prove that the operation is complete. Observe that if $(\text{ins}(A, Zs, Ys), \text{ord}(Ys))\theta$ is *true* in $\text{lfp}(\Phi_{P_2})$ then $Ys\theta$ is an ordered list and $Zs\theta$ is a sublist of $Ys\theta$; hence also $Zs\theta$ is ordered and $(\text{ord}(Zs), \text{ins}(A, Zs, Ys), \text{ord}(Ys))\theta$ is also *true* in $\text{lfp}(\Phi_{P_2})$. By Lemma 6.3, this is sufficient to state that:

$$\text{ins}(A, Zs, Ys), \text{ord}(Ys) \preceq_{\text{Comp}_{\mathcal{L}}(P_2) \cup \text{DCA}_{\mathcal{L}}} \text{ord}(Zs), \text{ins}(A, Zs, Ys), \text{ord}(Ys).^3$$

Moreover, the conjunction $\text{ord}(Zs), \text{ins}(A, Zs, Ys), \text{ord}(Ys)$ is independent from clause $c10$, hence, by Theorem 5.11, the operation is $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$ -complete.

To show that the operation is safe we could use Corollary 5.12, but in this case it is easier to observe that $\text{lfp}(\Phi_{P_2})$ is also a *total* model,⁴ that is, no ground atom is *undefined* in it, and therefore that $\text{lfp}(\Phi_{P_2}) \subseteq \text{lfp}(\Phi_{P_3})$ implies that $\text{lfp}(\Phi_{P_2}) = \text{lfp}(\Phi_{P_3})$. By Lemma 5.7 this implies that the operation is also safe.

(4) We can now fatten $c12$ with $\text{sort}(Xs, Zs)$. The resulting program is:

$P_4 = \{c1, \dots, c7\} \cup$

$\{ \begin{array}{l} c11 : \text{sort}([], []). \\ c13 : \text{sort}([A \mid Xs], Ys) \leftarrow \begin{array}{l} \text{sort}(Xs, Zs), \\ \text{perm}(Xs, Zs), \\ \text{ord}(Zs), \\ \text{ins}(A, Zs, Ys), \\ \text{ord}(Ys). \end{array} \end{array} \}$

³When 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 $(\text{ins}(A, Zs, Ys), \text{ord}(Ys))\theta$ is *true* in $\Phi_{P_2}^n$ then also $\text{ord}(Zs)\theta$ is *true* in $\Phi_{P_2}^n$.

⁴This 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.

This operation corresponds to a replacement of $\text{perm}(Xs, Zs), \text{ord}(Zs)$ with $\text{sort}(Xs, Zs), \text{perm}(Xs, Zs), \text{ord}(Zs)$. Using Corollary 5.12 we can prove that the operation is safe, in order to do it we prove that:

- (a) $\text{sort}(Xs, Zs), \text{perm}(Xs, Zs), \text{ord}(Zs) \cong_{\text{CompL}(P_3) \cup \text{DCA}_L} \text{perm}(Xs, Zs), \text{ord}(Zs)$;
- (b) the delay of $\text{sort}(Xs, Zs), \text{perm}(Xs, Zs), \text{ord}(Zs)$ w.r.t. $\text{perm}(Xs, Zs), \text{ord}(Zs)$ in $\text{lfp}(\Phi_{P_3})$ is zero.

To prove (a) we proceed as follows: since $\text{sort}(Xs, Zs) \leftarrow \text{perm}(Xs, Zs), \text{ord}(Zs)$, is a clause of P_0 , by Lemma 4.13, $\text{sort}(Xs, Zs) \cong_{\text{CompL}(P_0) \cup \text{DCA}_L} \text{perm}(Xs, Zs), \text{ord}(Zs)$. This clearly implies that $\text{sort}(Xs, Zs), \text{perm}(Xs, Zs), \text{ord}(Zs) \cong_{\text{CompL}(P_0) \cup \text{DCA}_L} \text{perm}(Xs, Zs), \text{ord}(Zs)$. Moreover, by the safeness of the previous transformation steps, P_0 is equivalent to P_3 and therefore, by a straightforward application of Lemma 5.8, we have that also (a) holds.

In order to prove (b), let us first prove a few properties. Here we denote the length of a list l by $|l|$.

- (i) $\text{ins}(A, Zs, Ys)\theta$ becomes *true* at step $\Phi_{P_3}^n$, where $n \leq |Ys\theta|$. In fact n is precisely the place where A ends up in Ys .

For example: $\text{ins}(a, [t, s, \dots], [a, t, s, \dots])$ is *true* in $\Phi_{P_3}^1$.

$\text{ins}(\bar{a}, [t, s, \dots], [t, a, s, \dots])$ is *true* in $\Phi_{P_3}^2$.

$\text{ins}(a, [t, s, \dots], [t, s, a, \dots])$ is *true* in $\Phi_{P_3}^3$

Moreover, when $\text{ins}(A, Zs, Ys)\theta$ is *true* in $\text{lfp}(\Phi_{P_3})$, we have that

$$|Ys\theta| = |Zs\theta| + 1. \quad (6.1)$$

- (ii) $\text{perm}(Xs, Zs)\theta$ becomes *true* in $\Phi_{P_3}^{|Zs\theta|+1}$.

This can be proven by induction on the length of $|Zs\theta|$.

$\text{perm}([], [])$ is *true* in $\Phi_{P_3}^1$;

if $|Zs\theta| > 0$ then $\text{perm}(Xs, Zs)\theta$ is *true* in $\Phi_{P_3}^\alpha$ iff there exists an instance of c2,

$(\text{perm}([A'|Xs'], Ys') \leftarrow \text{perm}(Xs', Zs'), \text{ins}(A', Zs', Ys'))\theta'$,

such that

- $\text{perm}([A'|Xs'], Ys')\theta' = \text{perm}(Xs, Zs)\theta$ and

- $(\text{perm}(Xs', Zs'), \text{ins}(A', Zs', Ys'))\theta'$ is *true* in $\Phi_{P_3}^{\alpha-1}$.

Now we can apply the inductive hypothesis and the previous results in order to determine $\alpha - 1$:

- $\text{perm}(Xs', Zs')\theta'$ is, by the inductive hypothesis, *true* in $\Phi_{P_3}^{|Zs'\theta'|+1}$;

- $\text{ins}(A', Zs', Ys')\theta'$ becomes *true* at step $\Phi_{P_3}^n$, where $n \leq |Ys'\theta'|$.

By (6.1), $|Ys'\theta'| = |Zs'\theta'| + 1$, hence the conjunction $(\text{perm}(Xs', Zs'), \text{ins}(A', Zs', Ys'))\theta'$ becomes *true* exactly at step $\Phi_{P_3}^{|Ys'\theta'|}$. But $|Ys'\theta'| = |Zs\theta|$, hence $\text{perm}(Xs, Zs)\theta$ becomes *true* at step $\Phi_{P_3}^{|Zs\theta|+1}$.

- (iii) $\text{ord}(Zs)\theta$ becomes *true* at step $\Phi_{P_3}^{m \times \alpha(1, |Zs\theta|)}$.

This can be proven by induction on $|Zs\theta|$.

- (iv) $\text{sort}(Xs, Zs)\theta$ becomes *true* at step $\Phi_{P_3}^{|Zs\theta|+1}$.

This can also be proven by induction on $|Zs\theta|$.

$\text{sort}([], [])$ is *true* in $\Phi_{P_3}^1$.

When $|Zs\theta| > 0$, $\text{sort}(Xs, Zs)\theta$ is in $\Phi_{P_3}^\alpha$ iff there exists an instance of c12:

$(\text{sort}([A \mid Xs'], Ys') \leftarrow \text{perm}(Xs', Zs'), \text{ord}(Zs'), \text{ins}(A, Zs', Ys'), \text{ord}(Ys')).\theta'$
such that

- $\text{sort}([A \mid Xs'], Ys')\theta' = \text{sort}(Xs, Zs)\theta$ and
- $(\text{perm}(Xs', Zs'), \text{ord}(Zs'), \text{ins}(A, Zs', Ys'), \text{ord}(Ys')).\theta'$ is true in $\Phi_{P_3}^{\alpha-1}$.

Now to determine the value of $\alpha - 1$, we can use (i), (ii) and (iii):

- $\text{perm}(Xs', Zs')\theta'$ is true in $\Phi_{P_3}^{|Zs'\theta'|+1}$;
- $\text{ord}(Zs')\theta'$ is true in $\Phi_{P_3}^{\max(1, |Zs'\theta'|)}$;
- $\text{ins}(A, Zs', Ys')\theta'$ is true in $\Phi_{P_3}^n$, where $n \leq |Ys'\theta'|$;
- $\text{ord}(Ys')\theta'$ is true in $\Phi_{P_3}^{\max(1, |Ys'\theta'|)}$.

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

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

This implies the following statement: for all θ , if $(\text{perm}(Xs, Zs), \text{ord}(Zs))\theta$ is true in some $\Phi_{P_3}^k$, then also $\text{sort}(Xs, Zs)\theta$ is true in $\Phi_{P_3}^k$.

Clearly, this can be restated as follows: for all θ , if $(\text{perm}(Xs, Zs), \text{ord}(Zs))\theta$ is true in some $\Phi_{P_3}^k$, then also $(\text{sort}(Xs, Zs), \text{perm}(Xs, Zs), \text{ord}(Zs))\theta$ is true in $\Phi_{P_3}^k$.

By Lemma 6.3 this implies (b).

Thinning

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

DEFINITION 6.5 (thin)

Let $cl : A \leftarrow \tilde{L}, \tilde{H}$. be a clause in a program P .

- *Thinning cl of the literals $\tilde{H}$* consists of substituting cl' for cl , where $cl' : A \leftarrow \tilde{L}$;

$$\text{thin}(P, cl, \tilde{H}) \stackrel{\text{def}}{=} P \setminus \{cl\} \cup \{cl'\}.$$

As for fattening, thinning can be interpreted as a replacement and then its applicability conditions can be inferred from Corollaries 5.12 and 5.5. Moreover Lemma 6.3 applies in a natural way also to this operation; only statement (c) requires a symmetric formulation. We now restate only this last point.

LEMMA 6.6

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

- If m is an integer such that, for each α and θ , $\Phi_P^\alpha \models \neg(\exists \tilde{x} \tilde{G}, \tilde{H})\theta$ implies $\Phi_P^{\alpha+m} \models \neg \exists \tilde{x} \tilde{G}\theta$, then
- $\exists \tilde{x} \tilde{G}, \tilde{H} \preceq_{\text{Comp}_L(P) \cup \text{DCA}_L} \exists \tilde{x} \tilde{G}$,
- the delay of $\exists \tilde{x} \tilde{G}$ w.r.t. $\exists \tilde{x} \tilde{G}, \tilde{H}$ in $\text{lfp}(\Phi_P)$ is smaller or equal to m .

If m is the least of such integers, then the delay of $\exists \tilde{x} \tilde{G}, \tilde{H}$ w.r.t. $\exists \tilde{x} \tilde{G}$ in $\text{lfp}(\Phi_P)$ is exactly m .

PROOF. It is a straightforward application of the fact that if $(\tilde{G}, \tilde{H})\theta$ is *true* in some interpretation I , then also $\tilde{G}\theta$ is *true* in I . ■

In Appendix B (Lemma B.2) we state a corresponding lemma for the case in which we adopt $Comp_{\mathcal{L}}(P) \cup WDCA_{\mathcal{L}}$ instead of $Comp_{\mathcal{L}}(P) \cup DCA_{\mathcal{L}}$.

EXAMPLE 6.7 (Sorting by permutation and check, part III)

6.1

(5) We can eliminate $ord(Zs)$ from the body of $c13$ by thinning it. The resulting program is:

$P_5 = \{c1, \dots, c7\} \cup$

{ $c11 : sort([], []).$
 $c14 : sort([A|Xs], Ys) \leftarrow sort(Xs, Zs), perm(Xs, Zs), ins(A, Zs, Ys),$
 $ord(Ys).$ }

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 $Comp_{\mathcal{L}}(P) \cup DCA_{\mathcal{L}}$ -complete, we apply Theorem 5.11. First we have to prove that

if $ord(Zs)\theta$ is *false* in $lfp(\Phi_{P_4})$ then $(ins(A, Zs, Ys), ord(Ys))\theta$ is *false* in $lfp(\Phi_{P_4})$.⁵ (6.2)

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 $Zs\theta$ is a sublist of $Ys\theta$, hence if $ord(Zs)\theta$ is *false* in $lfp(\Phi_{P_4})$, so is $ord(Ys)\theta$; and (6.2) follows.

Now (6.2) 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 6.3, that

$$ord(Zs), ins(A, Zs, Ys), ord(Ys) \preceq_{Comp_{\mathcal{L}}(P_4) \cup DCA_{\mathcal{L}}} ins(A, Zs, Ys), ord(Ys).$$

Since we also have that $ins(A, Zs, Ys), ord(Ys)$ is independent from $c13$, from Theorem 5.11 it follows that the operation is $Comp_{\mathcal{L}}(P) \cup DCA_{\mathcal{L}}$ -complete.

As in part (3), since $lfp(\Phi_{P_4})$ is a total interpretation, $lfp(\Phi_{P_4}) \supseteq lfp(\Phi_{P_5})$ implies that $lfp(\Phi_{P_4}) = lfp(\Phi_{P_5})$. In other words, the completeness of the operation implies its safeness (w.r.t. $Comp_{\mathcal{L}}(P) \cup DCA_{\mathcal{L}}$).

(6) Finally we can eliminate $perm(Xs, Zs)$ from the body of $c14$ by a further thinning, thus obtaining:

⁵When 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 $Ys\theta$ 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 $Xs\theta$ 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 $\Phi_{P_4}^{|Xs\theta|}$.

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 $\Phi_{P_4}^{j+1}$.

$$P_6 = \{c1, \dots, c7\} \cup$$

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

This is an $O(n^3)$ sorting program, while P_0 runs in $O(n!)$.

To prove the $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$ -completeness of this last step, we use Theorem 5.11. Let us distinguish two cases.

- If $Xs\theta = []$, then $\text{perm}(Xs, Zs)\theta$ is *false* in $\Phi_{P_6}^1$ iff $Zs\theta \neq []$, but in this case also $\text{sort}(Xs, Zs)\theta$ is *false* in $\Phi_{P_6}^1$;
- 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 $\text{body}(c2)$ is *false* in some interpretation I , then the corresponding instance of $\text{body}(c14)$ is *false* in I . Hence, if $\text{perm}([A|Xs], Zs)\theta$ is *false* in $\Phi_{P_6}(I)$ then $\text{sort}([A|Xs], Zs)\theta$ is *false* in $\Phi_{P_6}(I)$.

It follows that

$$\text{if } (\text{sort}(Xs, Zs), \text{perm}(Xs, Zs))\theta \text{ is false in } \Phi_{P_6}^j \text{ then } \text{sort}(Xs, Zs)\theta \text{ is false in } \Phi_{P_6}^j.$$

By Lemma 6.6, this is sufficient to show that $\text{sort}(Xs, Zs), \text{perm}(Xs, Zs) \preceq_{\text{Comp}_{\mathcal{L}}(P_6) \cup \text{DCA}_{\mathcal{L}}} \text{sort}(Xs, Zs)$ and that the semantic delay of $\text{sort}(Xs, Zs), \text{perm}(Xs, Zs)$ w.r.t. $\text{sort}(Xs, Zs)$ is zero, and hence, by Theorem 5.11, the operation is $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$ -complete.

On the other hand, if $\text{sort}(Xs, Zs)\theta$ is *true* in some interpretation I , then $Zs\theta$ must be a reordering of $Xs\theta$, therefore $\text{perm}(Xs, Zs)\theta$ is also *true* in I . It follows that

$$\begin{array}{l} \text{if } \text{sort}(Xs, Zs)\theta \text{ is true in } \text{lfp}(\Phi_{P_6}), \text{ then also} \\ (\text{sort}(Xs, Zs), \text{perm}(Xs, Zs))\theta \text{ is true in } \text{lfp}(\Phi_{P_6}). \end{array}$$

By Lemma 6.3, this implies that $\text{sort}(Xs, Zs) \preceq_{\text{Comp}_{\mathcal{L}}(P_6) \cup \text{DCA}_{\mathcal{L}}} \text{sort}(Xs, Zs), \text{perm}(Xs, Zs)$, and hence, by Theorem 5.9, that the operation is also $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$ -partially correct.

7 Conclusions

In this paper we have studied the simultaneous replacement operation w.r.t. normal programs. Simultaneous replacement is a transformation operation which consists in substituting a set of conjunctions of literals $\{\tilde{C}_1, \dots, \tilde{C}_n\}$ in the bodies of some clauses, with a set of equivalent conjunctions $\{\tilde{D}_1, \dots, \tilde{D}_n\}$. The set of logical consequences of the program's completion is considered 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:

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

- $Comp_{\mathcal{L}}(P) \cup DCA_{\mathcal{L}}$,
where $\mathcal{L}$ is a finite language and DCA is the set of Domain Closure Axioms.

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 and characterize formula equivalence, program equivalence and safeness of program transformations, namely their correctness and completeness, and express them 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 $\tilde{C}_i$ and $\tilde{D}_i$ is obviously necessary but it is generally not sufficient. In fact, as is shown by Corollary 4.9, we also need the equivalence to hold after the transformation. Such equivalence can be destroyed when a $\tilde{D}_i$ depends on one of the clauses on which the replacement is performed. Hence we establish a relation between the level of dependency of $\{\tilde{D}_1, \dots, \tilde{D}_n\}$ over the modified clauses and the difference in 'semantic complexity' between each $\tilde{C}_i$ and $\tilde{D}_i$. 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.

We thank the anonymous referees for many useful comments and for suggesting Example 4.25.

This research was done partly during the third author's stay at CWI, Centre for Mathematics and Computer Science, Amsterdam.

References

- [1] K. R. Apt. Introduction to logic programming. In *Handbook of Theoretical Computer Science*, Volume B: Formal Models and Semantics, J. van Leeuwen, ed. Elsevier, Amsterdam and The MIT Press, Cambridge, MA, 1990.
- [2] K. R. 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 *Programming Language Implementation and Logic Programming - Proceedings PLILP'92*, M. Bruynooghe and M. Wirsing, eds, Volume 631 of *Lecture Notes in Computer Science*, pp. 172–186. Springer-Verlag, Berlin, 1992.
- [5] A. Bossi, N. Cocco and S. Etalle. Transforming normal programs by replacement. In *Meta Programming in Logic - Proceedings META'92*, A. Pettorossi, ed., Volume 649 of *Lecture Notes in Computer Science*, pp. 265–279. Springer-Verlag, Berlin, 1992.
- [6] A. Bossi, N. Cocco and S. Etalle. Simultaneous replacement in normal programs. Technical Report CS-R9357, CWI, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, August 1993. Available via anonymous ftp at ftp.cwi.nl, or via xmosaic at <http://www.cwi.nl/cwi/publications/index.html>.

- [7] A. Bossi and S. Etalle. More on unfold/fold transformations of normal programs: preservation of Fitting's semantics. In *Proc. Fourth International Workshop on Meta Programming in Logic*, F. Turini, ed., Springer-Verlag, Berlin, 1994.
- [8] K. L. Clark. Negation as failure rule. In *Logic and Data Bases*, H. Gallaire and G. Minker, eds, pp. 293–322. Plenum Press, New York, 1978.
- [9] S. Etalle. Trasformazione dei programmi logici con negazione, tesi di laurea, Dip. Matematica Pura e Applicata, Università di Padova, Padova, Italy, July 1991.
- [10] M. Falaschi, G. Levi, M. Martelli and C. Palamidessi. Declarative modelling of the operational behavior of logic languages. *Theoretical Computer Science*, **69**, 289–318, 1989.
- [11] M. Fitting. A Kripke–Kleene semantics for logic programs. *Journal of Logic Programming*, **2**, 295–312, 1985.
- [12] P. A. Gardner and J. C. Shepherdson. Unfold/fold transformations of logic programs. In *Computational Logic: Essays in Honor of Alan Robinson*, J.-L. Lassez and G. Plotkin, eds. MIT Press, Boston, MA, 1991.
- [13] S. C. Kleene. *Introduction to Metamathematics*. D. van Nostrand, Princeton, NJ, 1952.
- [14] K. Kunen. Negation in logic programming. *Journal of Logic Programming*, **4**, 289–308, 1987.
- [15] J. W. Lloyd. *Foundations of Logic Programming*, second edition, Springer-Verlag, Berlin, 1987.
- [16] M. J. Maher. Correctness of a logic program transformation system. IBM Research Report RC13496, T. J. Watson Research Center, 1987.
- [17] M. J. Maher. A transformation system for deductive databases with perfect model semantics. *Theoretical Computer Science*, **110**, 377–403, 1993.
- [18] T. Sato. Equivalence-preserving first-order unfold/fold transformation system. *Theoretical Computer Science*, **105**, 57–84, 1992.
- [19] J. C. Shepherdson. Language and equality theory in logic programming. Technical Report PM-88-08, University Walk, Bristol, England, 1988.
- [20] H. Tamaki and T. Sato. Unfold/fold transformations of logic programs. In *Proc. Second Int'l Conf. on Logic Programming*, Sten-Åke Tärnlund, ed., pp. 127–139, 1984.

Appendices

A Proofs

A.1 Proof of Theorem 3.4

We need a lemma first.

LEMMA A.1

Let P be a normal program and χ an allowed formula with free variables $\bar{x}$. For each integer n , there exist two formulas in the language of equality, T_χ^n and F_χ^n , with free variables $\bar{x}$ such that, for any tuple $\bar{i}$ of ground terms,

- $T_\chi^n(\bar{i}/\bar{x})$ is true in Φ_P^n iff $\chi(\bar{i}/\bar{x})$ is;
in any other case $T_\chi^n(\bar{i}/\bar{x})$ is false in Φ_P^n .
- $F_\chi^n(\bar{i}/\bar{x})$ is true in Φ_P^n iff $\chi(\bar{i}/\bar{x})$ is false in Φ_P^n ;
in any other case $F_\chi^n(\bar{i}/\bar{x})$ is false in Φ_P^n .

PROOF. From Lemma 4.1 in [19] it follows that $T_\chi^n(\bar{i}/\bar{x})$ is true in Φ_P^n iff $\chi(\bar{i}/\bar{x})$ is, and that $F_\chi^n(\bar{i}/\bar{x})$ is true in Φ_P^n iff $\chi(\bar{i}/\bar{x})$ is false in Φ_P^n . From the completeness of CET_L in the case that the underlying universe is the Herbrand universe, we have that when $T_\chi^n(\bar{i}/\bar{x})$ (resp. $F_\chi^n(\bar{i}/\bar{x})$) is not true in Φ_P^n , it has to be false in Φ_P^n . ■

Actually, this result holds for any choice of L . 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

$$P = \left\{ \begin{array}{l} n(0). \\ n(s(x)) \leftarrow n(x) \end{array} \right\}.$$

We have that

$$\begin{aligned} T_n^1(x) &\equiv x = 0, \\ T_n^2(x) &\equiv x = 0 \vee x = 1, \end{aligned}$$

...

On the other hand,

$$\begin{aligned} F_n^1(x) &\equiv x \neq 0 \wedge \neg \exists y \, x = s(y), \\ F_n^2(x) &\equiv (x \neq 0 \wedge \neg \exists y \, x = s(y)) \vee (\exists y \, x = s(y) \vee (y \neq 0 \wedge \neg \exists z \, y = s(z))), \dots \end{aligned}$$

We can now prove the result we were aiming at.

THEOREM A.2 (3.4)

Let P_1 and P_2 be two normal programs.

If for all ϕ ,

$$Comp_{\mathcal{L}}(P_1) \models \phi \text{ implies } Comp_{\mathcal{L}}(P_2) \models \phi$$

then

$$\forall n \exists m \, \Phi_{P_1}^n \subseteq \Phi_{P_2}^m$$

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

PROOF. The proof is by contradiction. Assume that for all ϕ , $Comp_{\mathcal{L}}(P_1) \models \phi$ implies $Comp_{\mathcal{L}}(P_2) \models \phi$ and that there exists a fixed n such that

$$\text{for all } m, \, \Phi_{P_1}^n \not\subseteq \Phi_{P_2}^m. \quad (\text{A.1})$$

For each predicate symbol p let $T_{p(\bar{x})}^n$ and $F_{p(\bar{x})}^n$ be the equality formulas described in Lemma A.1. Hence $T_{p(\bar{x})}^n(\bar{t}/\bar{x})$ is true in Φ_P^n iff $p(\bar{t}/\bar{x})$ is, and $F_{p(\bar{x})}^n(\bar{t}/\bar{x})$ is true in Φ_P^n iff $p(\bar{t}/\bar{x})$ is false in Φ_P^n . Let also

$$\chi \equiv \bigwedge_{p \in pred(P_1)} \forall \bar{x} (T_{p(\bar{x})}^n \rightarrow p(\bar{x}) \wedge F_{p(\bar{x})}^n \rightarrow \neg p(\bar{x}))$$

where p ranges over the finite set of predicate symbols occurring in P_1 . From Lemma A.1 it follows that $\Phi_{P_1}^n \models \chi$, and, by Theorem 3.2

$$Comp_{\mathcal{L}}(P_1) \models \chi.$$

By hypothesis we have that $Comp_{\mathcal{L}}(P_2) \models \chi$, and, by Theorem 3.2 there exists an integer r such that

$$\Phi_{P_2}^r \models \chi.$$

By (A.1) $\Phi_{P_1}^n \not\subseteq \Phi_{P_2}^r$, hence there exists a ground atom $q(\bar{t})$ such that

$$\text{either } \Phi_{P_1}^n \models q(\bar{t}) \text{ and } \Phi_{P_2}^r \not\models q(\bar{t}) \quad \text{or} \quad \Phi_{P_1}^n \models \neg q(\bar{t}) \text{ and } \Phi_{P_2}^r \not\models \neg q(\bar{t}).$$

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

$$\Phi_{P_1}^n \models q(\bar{t}) \quad \text{and} \quad \Phi_{P_2}^r \not\models q(\bar{t}) \quad (\text{A.2})$$

By the left-hand side of A.2 and the definition of $T_{q(\bar{x})}^n$ in Lemma A.1,

$$\Phi_{P_1}^n \models T_{q(\bar{x})}^n(\bar{t}/\bar{x}).$$

$T_{q(\bar{x})}^n(\bar{t}/\bar{x})$ is a formula of the equality language and contains no predicate symbols other than '=', so if it is true in $\Phi_{P_1}^n$ it must be true also in $\Phi_{P_1}^0$, i.e. $\Phi_{P_1}^0 \models T_{q(\bar{x})}^n(\bar{t}/\bar{x})$. But $\Phi_{P_1}^0 = (\emptyset, \emptyset) \subseteq \Phi_{P_2}^r$, hence

$$\Phi_{P_2}^r \models T_{q(\bar{x})}^n(\bar{t}/\bar{x}).$$

Since $\Phi_{P_2}^r \models \chi$, from the definition of χ , it follows that also $\Phi_{P_2}^r \models \forall \bar{x} (T_{q(\bar{x})}^n(\bar{x}) \rightarrow q(\bar{x}))$, hence $\Phi_{P_2}^r \models T_{q(\bar{x})}^n(\bar{t}/\bar{x}) \rightarrow q(\bar{t})$; and, from the above statement,

$$\Phi_{P_2}^r \models q(\bar{t})$$

which contradicts the right hand side of (A.2). ■

A.2 Proof of Lemma 4.6

LEMMA A.3 (4.6)

Let P be a normal program, χ and ϕ be first-order allowed formulas and $\bar{x} = \{x_1, \dots, x_k\} = FV(\chi) \cup FV(\phi)$. The following statements are equivalent

(a) $\chi \preceq_{Comp_{\mathcal{L}}(P)} \phi$;

(b) $\forall n \exists m \forall \bar{t} \quad \Phi_P^n \models (\neg)\chi(\bar{t}/\bar{x})$ implies $\Phi_P^m \models (\neg)\phi(\bar{t}/\bar{x})$;

where n, m are quantified over natural numbers and $\bar{t}$ 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 -tuple of $\mathcal{L}$ -terms $\bar{t}_m$ for which the following hold

(i) $\Phi_P^n \models (\neg)\chi(\bar{t}_m/\bar{x})$;

(ii) $\Phi_P^m \not\models (\neg)\phi(\bar{t}_m/\bar{x})$.

By Lemma A.1 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

$$\Phi_P^n \models \forall \bar{x} (T_\chi^n \rightarrow \chi \wedge F_\chi^n \rightarrow \neg\chi).$$

By Theorem 3.2

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

By (a),

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

This is an allowed formula, then by Theorem 3.2 there exists an r such that

$$\Phi_P^r \models \forall \bar{x} (T_\chi^n \rightarrow \phi \wedge F_\chi^n \rightarrow \neg\phi). \quad (A.3)$$

But by (i) $\chi(\bar{t}_r/\bar{x})$ is either *true* or *false* in Φ_P^n ; let us now consider just the first possibility, that is

$$\Phi_P^n \models \chi(\bar{t}_r/\bar{x}),$$

the other case is perfectly symmetrical and omitted here.

From this and the definition of T_χ^n in Lemma A.1, we have $\Phi_P^n \models T_\chi^n(\bar{t}_r/\bar{x})$, and since $T_\chi^n(\bar{t}_r)$ is a formula in the language of equality, if it is *true* in Φ_P^n it must be *true* already at stage 0, that is $\Phi_P^0 \models T_\chi^n(\bar{t}_r/\bar{x})$, but $\Phi_P^0 \subseteq \Phi_P^r$, hence

$$\Phi_P^r \models T_\chi^n(\bar{t}_r/\bar{x}).$$

But then, by (A.3), $\Phi_P^r \models \phi(\bar{t}_r/\bar{x})$, contradicting (ii).

(b) implies (a)

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

$$\Phi_P^n \models \zeta \sigma \text{ implies } \Phi_P^m \models \zeta[\phi/\chi]\sigma. \quad (A.4)$$

By Theorem 3.2 this implies (a).

Fix an n , and let m be an integer that satisfies hypothesis (b). It is not restrictive to assume that $m \geq n$. Let ζ be an allowed formula and σ a substitution such that

$$\Phi_P^n \models \zeta \sigma.$$

If ζ does not contain χ as a subformula then (A.4) follows immediately from the assumption that $m \geq n$. In the case that ζ contains χ as a subformula we proceed by induction on the structure of ζ .

Base step: $\zeta = \chi$, then (A.4) follows immediately from (b).

Induction step: we consider three cases:

(1) If $\zeta = \Delta \zeta_1$, where Δ 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 χ as a subformula (and the result holds trivially) or the inductive hypothesis applies.

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

For each $\mathcal{L}$ -term t , let γ_t be the substitution $[t/w]$. Since $\Phi_P^n \models \zeta \sigma$, we have that

$$\text{for each } \mathcal{L}\text{-term } t, \Phi_P^n \models \zeta_1 \gamma_t \sigma.$$

By the inductive hypothesis there exists an m such that

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

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

$$\Phi_P^m \models (\forall w \zeta_1[\phi/\chi])\sigma.$$

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

A.3 Proof of Lemma 4.8

Let us first state a simple property of existentially quantified formulas.

REMARK A.4

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

- If $\exists \tilde{z} \tilde{L}\theta$ is true in I then $\exists \tilde{w} \tilde{L}\theta$ is true in I .
- If $\exists \tilde{z} \tilde{L}\theta$ is not false in I then $\exists \tilde{w} \tilde{L}\theta$ is not false in I .

This is true in particular when $\tilde{z}$ is empty and $\exists \tilde{z} \tilde{L}\theta = \tilde{L}\theta$.

LEMMA A.5 (4.8)

Notation as in Theorem 4.7. 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 $\tilde{C}_j \in \{\tilde{C}_1, \dots, \tilde{C}_n\}$ and a ground substitution θ such that:

- either $I' \models \exists \tilde{x}_j \tilde{D}_j\theta$ while $I \not\models \exists \tilde{x}_j \tilde{C}_j\theta$;
- or $I' \models \neg \exists \tilde{x}_j \tilde{D}_j\theta$ while $I \not\models \neg \exists \tilde{x}_j \tilde{C}_j\theta$.

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 $B \in \Phi_{P'}(I')^+ \setminus \Phi_P(I)^+$. There has to be a clause $c \in P' \setminus P$, a ground substitution θ' such that: $\text{head}(c)\theta' = B$ and $\text{body}(c)\theta'$ is true in I' . $P' \setminus P = \{cl'_1, \dots, cl'_p\}$, then there is an integer j such that: $c = cl'_j$ and $\text{body}(cl'_j)\theta' = (\tilde{D}_{j_1}, \dots, \tilde{D}_{j_{r(j)}}, \tilde{E}_j)\theta'$ is true in I' . Hence the conjunctions $\tilde{D}_{j_1}\theta', \dots, \tilde{D}_{j_{r(j)}}\theta'$ are all true in I' . From Remark A.4 it follows that the formulas

$$\exists \tilde{x}_{j_1} \tilde{D}_{j_1}\theta', \dots, \exists \tilde{x}_{j_{r(j)}} \tilde{D}_{j_{r(j)}}\theta' \text{ are true in } I', \quad (\text{A.5})$$

where the $\tilde{x}_i$ are sets of variables that satisfy the locality property w.r.t. to $\tilde{C}_i$ and $\tilde{D}_i$. We know that $B = \text{head}(cl'_j)\theta' = \text{head}(cl_j)\theta'$, but since $B \notin \Phi_P(I)^+$, by Definition 2.6 we have that $(\exists \tilde{w} \text{body}(cl_j))\theta'$ is not true in I , where $\tilde{w} = \text{Var}(\text{body}(cl_j)) \setminus \text{Var}(\text{head}(cl_j))$, that is, $(\exists \tilde{w} \tilde{C}_{j_1}, \dots, \tilde{C}_{j_{r(j)}}, \tilde{E}_j)\theta'$ is not true in I .

For each k , $\tilde{w} \supseteq \tilde{x}_{j_k} \cap \text{Var}(\text{body}(cl_j))$, now let $\tilde{y} = \tilde{w} \setminus \tilde{x}_{j_1} \cup \dots \cup \tilde{x}_{j_{r(j)}}$ and θ be a ground extension of θ' whose domain contains $\tilde{y}$. Then from Remark A.4 it follows that

$$(\exists \tilde{x}_{j_1}, \dots, \tilde{x}_{j_{r(j)}} \tilde{C}_{j_1}, \dots, \tilde{C}_{j_{r(j)}}, \tilde{E}_j)\theta \text{ is not true in } I.$$

Since $\tilde{E}_j\theta$ is true in I' and $I' \subseteq I$, then $\tilde{E}_j\theta$ is true in I , by the locality property, the sets $\tilde{x}_{j_k}$ are pairwise disjoint, hence one of the formulas in $\exists \tilde{x}_{j_1} \tilde{C}_{j_1}\theta, \dots, \exists \tilde{x}_{j_{r(j)}} \tilde{C}_{j_{r(j)}}\theta$ is not true in I .

Since (A.5) 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 $B \in \Phi_{P'}(I')^- \setminus \Phi_P(I)^-$. There has to be a clause $c \in P \setminus P'$, a ground substitution θ' such that $\text{head}(c)\theta' = B$ and $\text{body}(c)\theta'$ is not false in I . $P \setminus P' = \{cl_1, \dots, cl_p\}$, then there is an integer j such that: $c = cl_j$, and then the conjunction $(\tilde{C}_{j_1}, \dots, \tilde{C}_{j_{r(j)}}, \tilde{E}_j)\theta'$ is not false in I . Hence the conjunctions $\tilde{C}_{j_1}\theta', \dots, \tilde{C}_{j_{r(j)}}\theta'$ are all not false in I . From Remark A.4 it follows that:

$$\exists \tilde{x}_{j_1} \tilde{C}_{j_1}\theta', \dots, \exists \tilde{x}_{j_{r(j)}} \tilde{C}_{j_{r(j)}}\theta' \text{ are not false in } I. \quad (\text{A.6})$$

We know that $B = \text{head}(cl_j)\theta' = \text{head}(cl'_j)\theta'$, but since $B \in \Phi_{P'}(I')^-$, by Definition 2.6 we have that $(\exists \tilde{w} \text{body}(cl'_j))\theta'$ is false in I' , with $\tilde{w} = \text{Var}(\text{body}(cl'_j)) \setminus \text{Var}(\text{head}(cl'_j))$, that is, $(\exists \tilde{w} \tilde{D}_{j_1}, \dots, \tilde{D}_{j_{r(j)}}, \tilde{E}_j)\theta'$ is false in I' . For each k , $\tilde{w} \supseteq \tilde{x}_{j_k} \cap \text{Var}(\text{body}(cl_j))$, now let $\tilde{y} = \tilde{w} \setminus \tilde{x}_{j_1} \cup \dots \cup \tilde{x}_{j_{r(j)}}$ and θ be a ground extension of θ' whose domain contains $\tilde{y}$. From Remark A.4 it follows that

$$(\exists \tilde{x}_{j_1}, \dots, \tilde{x}_{j_{r(j)}} \tilde{D}_{j_1}, \dots, \tilde{D}_{j_{r(j)}}, \tilde{E}_j)\theta \text{ is false in } I'.$$

Since $\tilde{E}_j\theta$ is not false in I and $I' \subseteq I$, $\tilde{E}_j\theta$ is not false in I' . By the locality property, the sets $\tilde{x}_{j_k}$ are pairwise disjoint, then one of the formulas in $\exists \tilde{x}_{j_1} \tilde{D}_{j_1} \theta \dots \exists \tilde{x}_{j_{r(j)}} \tilde{D}_{j_{r(j)}} \theta$ is false in I' .

Since (A.6) holds also for θ , the thesis follows. ■

A.4 Proof of Lemma 5.8

LEMMA A.6 (5.8)

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

- 9a) $\chi \preceq_{\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}} \phi$;
- (b) $\text{lfp}(\Phi_P) \models \chi \Rightarrow \phi$.

PROOF. (a) implies (b).

By the definition of the operator $\Rightarrow$, (b) is equivalent to

for each tuple of $\mathcal{L}$ -terms $\tilde{t}$, $\text{lfp}(\Phi_P) \models (\neg)\chi(\tilde{t}/\tilde{x})$ implies $\text{lfp}(\Phi_{P'}) \models (\neg)\phi(\tilde{t}/\tilde{x})$.

By Theorem 5.6 this is equivalent to the following statement:

for each tuple of $\mathcal{L}$ -terms $\tilde{t}$, $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models (\neg)\chi(\tilde{t}/\tilde{x})$ implies $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models (\neg)\phi(\tilde{t}/\tilde{x})$.

This is immediate by Definition 4.2.

(b) implies (a).

Let ζ be any allowed formula such that $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \zeta$, σ be any ground substitution; we have to prove that $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \zeta[\phi\sigma/\chi\sigma]$. 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 5.6, $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \chi\sigma$ implies that $\text{lfp}(\Phi_P) \models \chi\sigma$. By (b) this implies that $\text{lfp}(\Phi_P) \models \phi\sigma$, and, by Theorem 5.6, that $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \phi\sigma$.

Since $\phi\sigma \equiv \zeta[\phi\sigma/\chi\sigma]$, this implies the thesis.

Induction step: we have to consider four cases:

(1) $\zeta \equiv \Delta \zeta_1$, where Δ is any allowed unary connective. The result holds trivially, since by the inductive hypothesis, $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models (\neg)\zeta_1$ implies $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models (\neg)\zeta_1[\phi\sigma/\chi\sigma]$.

(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 $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \forall w \zeta_1(w)$. This is equivalent to: for any $\mathcal{L}$ -term t , $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \zeta_1(t)$. 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 , $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \zeta_1(t)[\phi\sigma\gamma_t/\chi\sigma\gamma_t]$. Since $\text{DCA}_{\mathcal{L}}$ forces the quantification to be over $\mathcal{L}$ -terms, and $\text{DCA}_{\mathcal{L}}$ is included in $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}}$, this implies that $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \forall w \zeta_1(w)[\phi\sigma/\chi\sigma]$. On the other hand, for the case when $\text{Comp}_{\mathcal{L}}(P) \cup \text{DCA}_{\mathcal{L}} \models \neg \forall w \zeta_1(w)$, a similar reasoning applies.

(4) $\zeta \equiv \exists w \zeta_1(w)$

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

B Fattening and Thinning: case $\text{WDCA}_{\mathcal{L}}$

Now we state two lemmata which are the counterpart of Lemmata 6.3 and 6.6, for the case in which the closure axioms adopted are $\text{WDCA}_{\mathcal{L}}$ rather than $\text{DCA}_{\mathcal{L}}$.

LEMMA B.1

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

- (a) If for each j there exists a k such that, for each θ , $\Phi_P^j \models \exists \tilde{x} \tilde{G}\theta$ implies $\Phi_P^k \models (\exists \tilde{x} \tilde{G}, \tilde{H})\theta$, then $\exists \tilde{x} \tilde{G} \preceq_{\text{Comp}_{\mathcal{L}}(P)} \exists \tilde{x} \tilde{G}, \tilde{H}$.

- (b) If for each j there exists a k such that, for each θ , $\Phi_P^j \models \neg(\exists \tilde{x} \tilde{G}, \tilde{H})\theta$ implies $\Phi_P^k \models \neg\exists \tilde{x} \tilde{G}\theta$, then $\exists \tilde{x} \tilde{G}, \tilde{H} \preceq_{Comp_{\mathcal{L}}(P)} \exists \tilde{x} \tilde{G}$.
- (c) If m is an integer such that, for each n and θ , $\Phi_P^n \models_{\mathcal{L}} \exists \tilde{x} \tilde{G}\theta$ implies $\Phi_P^{n+m} \models_{\mathcal{L}} (\exists \tilde{x} \tilde{G}, \tilde{H})\theta$ then
- $\exists \tilde{x} \tilde{G} \preceq_{Comp_{\mathcal{L}}(P)} \exists \tilde{x} \tilde{G}, \tilde{H}$;
 - the delay of $\exists \tilde{x} \tilde{G}, \tilde{H}$ w.r.t. $\exists \tilde{x} \tilde{G}$ in $Comp_{\mathcal{L}}(P) \cup WDCA_{\mathcal{L}}$ is smaller or equal to m .
- If m is the least of such integers, then the delay of $\exists \tilde{x} \tilde{G}, \tilde{H}$ w.r.t. $\exists \tilde{x} \tilde{G}$ in $Comp_{\mathcal{L}}(P) \cup WDCA_{\mathcal{L}}$ is exactly m .

PROOF. It is a straightforward application of Theorem 5.4 together with the fact that, if $\tilde{G}\theta$ is *false* in some interpretation I , then also $(\tilde{G}, \tilde{H})\theta$ is *false* in I . ■

LEMMA B.2

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

- If m is an integer such that, for each integer n and substitution θ , $\exists \tilde{x} (\tilde{G}, \tilde{H})\theta$ *false* in Φ_P^n implies $\exists \tilde{x} \tilde{G}\theta$ *false* in Φ_P^{n+m} , then
 - $\exists \tilde{x} \tilde{G}, \tilde{H} \preceq_{Comp_{\mathcal{L}}(P)} \exists \tilde{x} \tilde{G}$,
 - the delay of $\exists \tilde{x} \tilde{G}$ w.r.t. $\exists \tilde{x} \tilde{G}, \tilde{H}$ in Φ_P^w is less or equal to m .
- If m is the least of such integers, then the delay of $\exists \tilde{x} \tilde{G}, \tilde{H}$ w.r.t. $\exists \tilde{x} \tilde{G}$ in Φ_P^w is exactly m .

PROOF. It is a straightforward application of the fact that if $(\tilde{G}, \tilde{H})\theta$ is *true* in some interpretation I , then also $\tilde{G}\theta$ is *true* in I . ■

Received 26 July 1993