A Hierarchy of Semantics for Non-deterministic Term Rewriting Systems*

(Extended version)

Tech. Rep. SIC-10-08, 2008

Juan Rodríguez-Hortalá

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain
`juanrh@fdi.ucm.es`

Abstract. Formalisms involving some degree of nondeterminism are frequent in computer science. In particular, various programming or specification languages are based on term rewriting systems where confluence is not required. In this paper we examine three concrete possible semantics for non-determinism that can be assigned to those programs. Two of them –call-time choice and run-time choice– are quite well-known, while the third one –plural semantics– is investigated for the first time in the context of term rewriting based programming languages. We investigate some basic intrinsic properties of the semantics and establish some relationships between them: we show that the three semantics form a hierarchy in the sense of set inclusion, and we prove that call-time choice and plural semantics enjoy a remarkable compositionality property that fails for run-time choice; finally, we show how to express plural semantics within run-time choice by means of a program transformation, for which we prove its adequacy.

1 Introduction

Term rewriting systems (*TRS's*) [4] have a long tradition as a suitable basic formalism to address a wide range of tasks in computer science, in particular, many specification languages [5, 8], theorem provers [25, 24, 6] and programming languages are based on TRS's. For instance, the syntax and theory of TRS's was the basis of the first formulations of *functional logic programming* (*FLP*) [12] through the idea of narrowing [11]. On the other hand, non-determinism is an expressive feature that has been used for a long time in system specification (e.g., non-deterministic Turing machines or automata) or for programming (the constructions of McCarthy [22] and Dijkstra [7] are classical examples). One of the appeals of term rewriting is its elegant way to express non-determinism through the use of a non-confluent TRS, obtaining a clean and high level representation of complex systems. In the field of FLP, non-confluent TRS's are used as programs to support non-strict non-deterministic functions, which are one of the most distinctive features of the paradigm [10, 3]. Those TRS's follow the constructor discipline also, corresponding to a value-based semantic view, in which the purpose of computations is to produce values made of constructors.

Therefore non-confluent constructor-based TRS's can be used as a common syntactic framework for FLP and rewriting. The set of rewrite rules constitutes a program and so we also call them *program rules*. Nevertheless the behaviour of current implementations of FLP and rewriting differ substantially, because the introduction of non-determinism in a functional setting gives rise to a variety of semantic decisions, that were explored in [23]. There the different language variants that result after adding non-determinism to a basic functional language were expounded, structuring the comparison as a choice among different options over several dimensions: strict/non-strict functions, angelic/demonic/erratic non-deterministic choices and *singular/plural semantics* for parameter passing. In the present paper we assume non-strict angelic non-determinism, and we are concerned about the last dimension only. The key difference is that under a singular

* This work has been partially supported by the Spanish projects Merit-Forms-UCM (TIN2005-09207-C03-03), Promesas-CAM (S-0505/TIC/0407) and FAST-STAMP (TIN2008-06622-C03-01/TIN).

semantics, in the substitutions used to instantiate the program rules for function application, the variables of the program rules should range over single objects of the set of values considered; in a plural semantics those range over sets of objects. This has been traditionally identified with the distinction between *call-time choice* and *run-time choice* [14] parameter passing mechanisms. Under call-time choice a value for each argument is computed before performing parameter passing, this corresponds to call-by-value in a strict setting and to call-by-need in a non-strict setting, in which a partial value instead of a total value is computed. On the other hand, run-time-choice corresponds to call-by-name, each argument is copied without any evaluation and so the different copies of any argument may evolve in different ways afterwards. Thus, traditionally it has been considered that call-time choice parameter passing inducts a singular semantics while run-time choice inducts a plural semantics.

Example 1. Consider the TRS $\mathcal{P} = \{f(c(X)) \rightarrow d(X,X), X \ ? \ Y \rightarrow X, X \ ? \ Y \rightarrow Y\}$. With call-time choice/singular semantics to compute a value for the term $f(c(0?1))$ we must first compute a (partial) value for $c(0?1)$, and then we may continue the computation with $f(c(0))$ or $f(c(1))$ which yield $d(0,0)$ or $d(1,1)$. Note that $d(0,1)$ and $d(1,0)$ are not correct values for $f(c(0?1))$ in that setting.
On the other hand with run-time choice/plural semantics to evaluate the term $f(c(0?1))$:

- Under the run-time choice point of view, the step $f(c(0?1)) \rightarrow d(0?1, 0?1)$ is sound, hence not only $d(0,0)$ and $d(1,1)$ but also $d(0,1)$ and $d(1,0)$ are valid values for $f(c(0?1))$.
- Under the plural semantics point of view, we consider the set $\{c(0), c(1)\}$ which is a subset of the set of values for $c(0?1)$ in which every element matches the argument pattern $c(X)$. Therefore the set $\{0,1\}$ can be used for parameter passing obtaining a kind of "set expression" $d(\{0,1\},\{0,1\})$, which evaluation yields the values $d(0,0)$, $d(1,1)$, $d(0,1)$ and $d(1,0)$.

In general, call-time choice/singular semantics produces less results than run-time choice/ plural semantics.

A standard formulation for call-time choice[1] in FLP is the *CRWL*[2] logic [9, 10], which is implemented by current FLP languages like Toy [18] or Curry [13]; traditional term rewriting may be considered the standard semantics for run-time choice[3], and is the basis for the semantics of languages like Maude [5], but has been rarely [1] thought as a valuable global alternative to call-time choice for the value-based view of FLP. However, there might be parts in a program or individual functions for which run-time choice could be a better option, and therefore it would be convenient to have both possibilities (run-time/call-time) at programmer's disposal [16]. Nevertheless the use of an operational notion like term rewriting as the semantic basis of a FLP language can lead us to confusing situations, not very compatible with the value-based semantic view that we wanted for the constructor-based TRS's used in FLP.

Example 2. Starting with the TRS of Example 1 we want to evaluate the expression $f(c(0) \ ? \ c(1))$ with run-time choice/plural semantics:

- Under the run-time choice point of view, that is, using term rewriting, the evaluation of the subexpression $c(0)?c(1)$ is needed in order to get an expression that matches the left hand side $f(c(X))$. Hence the derivations $f(c(0)?c(1)) \rightarrow f(c(0)) \rightarrow d(0,0)$ and $f(c(0)?c(1)) \rightarrow f(c(1)) \rightarrow d(1,1)$ are sound and compute the values $d(0,0)$ and $d(1,1)$, but neither $d(0,1)$ nor $d(1,0)$ are correct values for $f(c(0)?c(1))$.
- Under the plural semantics point of view, we consider the set $\{c(0), c(1)\}$ which is a subset of the set of values for $c(0)?c(1)$ in which every element matches the argument pattern $c(X)$. Therefore the set $\{0,1\}$ can be used for parameter passing obtaining a kind of "set expression" $d(\{0,1\},\{0,1\})$ that yields the values $d(0,0)$, $d(1,1)$, $d(0,1)$ and $d(1,0)$.

Which of these is the more suitable perspective for FLP?

[1] In fact angelic non-strict call-time choice.
[2] **C**onstructor-based **R**e**W**riting **L**ogic.
[3] In fact angelic non-strict run-time choice.

This problem did not appear in [23] because no pattern matching was present, nor in [14] because only call-time choice was adopted (and this conflict does not appear between the call-time choice and the singular semantics views). Choosing the run-time choice perspective of term rewriting has some unpleasant consequences. First of all the expression $f(c(0?1))$ has more values than the expression $f(c(0)?c(1))$, even when the only difference between them is the subexpressions $c(0?1)$ and $c(0)?c(1)$, which have the same values both in call-time choice, run-time choice and plural semantics. This is pretty incompatible with the value-based semantic view we are looking for in FLP. And this has to do with the loss of some desirable properties, present in *CRWL*, when switching to run-time choice. We will see how plural semantics recovers those properties, which are very useful for reasoning about computations. Furthermore it allows natural encodings of some programs that need to do some collecting work, as we will see later (Example 4). Hence we claim that the plural semantics perspective is more suitable for a value-based programming language.

The rest of the paper is organized as follows. Section 2 contains some technical preliminaries and notations about *CRWL* and term rewriting systems. In Section 3 we introduce $\pi CRWL$, a variation of *CRWL* to express plural semantics, and present some of its properties. In Section 4 we discuss about the different properties of these semantics and prove the inclusion chain $CRWL \subseteq$ rewriting $\subseteq \pi CRWL$, that constitutes a hierarchy of semantics for non-determinism. Section 5 recalls that no straight simulation of *CRWL* in term rewriting can be done by a program transformation, and vice versa, and shows a novel transformation to simulate $\pi CRWL$ using term rewriting. Finally Section 6 summarizes some conclusions and future work. Fully detailed proofs, including some auxiliary results, can be found in Appendix A.

2 Preliminaries

2.1 Constructor based term rewriting systems

We consider a first order signature $\Sigma = CS \cup FS$, where CS and FS are two disjoint set of *constructor* and defined *function* symbols respectively, all them with associated arity. We write CS^n (FS^n resp.) for the set of constructor (function) symbols of arity n. We write $c, d, \ldots$ for constructors, $f, g, \ldots$ for functions and $X, Y, \ldots$ for variables of a numerable set $\mathcal{V}$. The notation $\overline{o}$ stands for tuples of any kind of syntactic objects. Given a set $\mathcal{A}$ we denote by $\mathcal{A}^*$ the set of finite sequences of elements of that set. For any sequence $a_1 \ldots a_n \in \mathcal{A}^*$ and function $f : \mathcal{A} \to \{true, false\}$, by $a_1 \ldots a_n \mid f$ we denote the sequence constructed taking in order every element from $a_1 \ldots a_n$ for which f holds.

The set Exp of *expressions* is defined as $Exp \ni e ::= X \mid h(e_1, \ldots, e_n)$, where $X \in \mathcal{V}$, $h \in CS^n \cup FS^n$ and $e_1, \ldots, e_n \in Exp$. The set $CTerm$ of *constructed terms* (or *c-terms*) is defined like Exp, but with h restricted to CS^n (so $CTerm \subseteq Exp$). The intended meaning is that Exp stands for evaluable expressions, i.e., expressions that can contain function symbols, while $CTerm$ stands for data terms representing **values**. We will write $e, e', \ldots$ for expressions and $t, s, \ldots$ for c-terms. The set of variables occurring in an expression e will be denoted as $var(e)$. We will frequently use *one-hole contexts*, defined as $Cntxt \ni \mathcal{C} ::= [\,] \mid h(e_1, \ldots, \mathcal{C}, \ldots, e_n)$, with $h \in CS^n \cup FS^n$. The application of a context $\mathcal{C}$ to an expression e, written by $\mathcal{C}[e]$, is defined inductively as $[\,][e] = e$ and $h(e_1, \ldots, \mathcal{C}, \ldots, e_n)[e] = h(e_1, \ldots, \mathcal{C}[e], \ldots, e_n)$.

Substitutions $\theta \in Subst$ are finite mappings $\theta : \mathcal{V} \longrightarrow Exp$, extending naturally to $\theta : Exp \longrightarrow Exp$. We write ϵ for the identity (or empty) substitution. We write $e\theta$ for the application of θ to e, and $\theta\theta'$ for the composition, defined by $X(\theta\theta') = (X\theta)\theta'$. The domain and range of θ are defined as $dom(\theta) = \{X \in \mathcal{V} \mid X\theta \neq X\}$ and $vran(\theta) = \bigcup_{X \in dom(\theta)} var(X\theta)$. If $dom(\theta_0) \cap dom(\theta_1) = \emptyset$, their disjoint union $\theta_0 \uplus \theta_1$ is defined by $(\theta_0 \uplus \theta_1)(X) = \theta_i(X)$, if $X \in dom(\theta_i)$ for some θ_i; $(\theta_0 \uplus \theta_1)(X) = X$ otherwise. Given $W \subseteq \mathcal{V}$ we write $\theta|_W$ for the restriction of θ to W, and $\theta|_{\setminus D}$ is a shortcut for $\theta|_{(\mathcal{V} \setminus D)}$. We will sometimes write $\theta = \sigma[W]$ instead of $\theta|_W = \sigma|_W$. *C-substitutions* $\theta \in CSubst$ verify that $X\theta \in CTerm$ for all $X \in dom(\theta)$.

A *constructor-based term rewriting system* $\mathcal{P}$ (*CS*) is a set of c-rewrite rules of the form $f(\overline{t}) \to r$ where $f \in FS^n$, $e \in Exp$ and $\overline{t}$ is a linear n-tuple of c-terms, where linearity means that variables occur only once in

$\bar{t}$. In the present work we restrict ourselves to CS's not containing *extra variables*, i.e., CS's for which $var(r) \subseteq var(f(\bar{t}))$ holds for any rewrite rule; the extension of this work to rules with extra variables is a subject of future work. We assume that every CS $\mathcal{P}$ contains the rules $\{X \mathbin{?} Y \to X, X \mathbin{?} Y \to Y, if\ true\ then\ X \to X\}$, defining the behaviour of $\_?\_ \in FS^2$, $if\_then\_ \in FS^2$, both used in mixfix mode, and that those are the only rules for that function symbols. For the sake of conciseness we will often omit these rules when presenting a CS.

Given a TRS $\mathcal{P}$, its associated *rewrite relation* $\to_{\mathcal{P}}$ is defined as: $\mathcal{C}[l\sigma] \to_{\mathcal{P}} \mathcal{C}[r\sigma]$ for any context $\mathcal{C}$, rule $l \to r \in \mathcal{P}$ and $\sigma \in Subst$. We write $\xrightarrow{*}_{\mathcal{P}}$ for the reflexive and transitive closure of the relation $\to_{\mathcal{P}}$. In the following, we will usually omit the reference to $\mathcal{P}$ or denote it by $\mathcal{P} \vdash e \to e'$ and $\mathcal{P} \vdash e \to^* e'$.

2.2 The CRWL framework

In the *CRWL* framework [9, 10], programs are CS's, also called *CRWL-programs* (or simply 'programs') from now on. To deal with non-strictness at the semantic level, we enlarge Σ with a new constant constructor symbol $\perp$. The sets $Exp_\perp$, $CTerm_\perp$, $Subst_\perp$, $CSubst_\perp$ of partial expressions, etc., are defined naturally. Notice that $\perp$ does not appear in programs. Partial expressions are ordered by the *approximation* ordering $\sqsubseteq$ defined as the least partial ordering satisfying $\perp \sqsubseteq e$ and $e \sqsubseteq e' \Rightarrow \mathcal{C}[e] \sqsubseteq \mathcal{C}[e']$ for all $e, e' \in Exp_\perp, \mathcal{C} \in Cntxt$. This partial ordering can be extended to substitutions: given $\theta, \sigma \in Subst_\perp$ we say $\theta \sqsubseteq \sigma$ if $X\theta \sqsubseteq X\sigma$ for all $X \in \mathcal{V}$.

The semantics of a program $\mathcal{P}$ is determined in *CRWL* by means of a proof calculus able to derive reduction statements of the form $e \rightarrowtriangle t$, with $e \in Exp_\perp$ and $t \in CTerm_\perp$, meaning informally that t is (or approximates to) a *possible value* of e, obtained by iterated reduction of e using $\mathcal{P}$ under call-time choice. The *CRWL*-proof calculus is presented in Figure 1. Rule **B** (bottom) allows any expression to be undefined

(RR)	$\dfrac{}{X \rightarrowtriangle X}$	$X \in \mathcal{V}$	**(DC)**	$\dfrac{e_1 \rightarrowtriangle t_1 \ldots e_n \rightarrowtriangle t_n}{c(e_1, \ldots, e_n) \rightarrowtriangle c(t_1, \ldots, t_n)}$	$c \in CS^n$
(B)	$\dfrac{}{e \rightarrowtriangle \perp}$		**(OR)**	$\dfrac{e_1 \rightarrowtriangle p_1\theta \ldots e_n \rightarrowtriangle p_n\theta \quad r\theta \rightarrowtriangle t}{f(e_1, \ldots, e_n) \rightarrowtriangle t}$	$f(p_1, \ldots, p_n) \to r \in \mathcal{P}$ $\theta \in CSubst_\perp$

Fig. 1. Rules of *CRWL*

or not evaluated (non-strict semantics). Rule **OR** (outer reduction) expresses that to evaluate a function call we must choose a compatible program rule, perform parameter passing (by means of a $CSubst_\perp$ θ) and then reduce the right-hand side. The use of partial c-substitutions in OR is essential to express call-time choice, as only single partial values are used for parameter passing.

We write $\mathcal{P} \vdash_{CRWL} e \rightarrowtriangle t$ to express that $e \rightarrowtriangle t$ is derivable in the *CRWL*-calculus using the program $\mathcal{P}$. Given a program $\mathcal{P}$, the *CRWL-denotation* of an expression $e \in Exp_\perp$ is defined as $[\![e]\!]^{sg}_{\mathcal{P}} = \{t \in CTerm_\perp \mid \mathcal{P} \vdash_{CRWL} e \rightarrowtriangle t\}$. In the following, we will usually omit the reference to $\mathcal{P}$.

3 π*CRWL*: a plural semantics for FLP

The new calculus $\pi CRWL$ is defined by modifying the rules of *CRWL* to consider sets of partial values for parameter passing instead of single partial values: hence, only the rule OR should be modified. To avoid the need to extend the syntax with new constructions to represent those "set expressions" that we talked about in the introduction, we will exploit the fact that $[\![e_1 \mathbin{?} e_2]\!] = [\![e_1]\!] \cup [\![e_2]\!]$. Therefore the substitutions used for parameter passing will map variables to "disjunctions of values". We define the set $CSubst^?_\perp = \{\theta \in Subst_\perp \mid \forall X \in dom(\theta), \theta(X) = t_1 \mathbin{?} \ldots \mathbin{?} t_n \text{ such that } t_1, \ldots, t_n \in CTerm_\perp, n > 0\}$, for which $CSubst_\perp \subseteq CSubst^?_\perp \subseteq Subst_\perp$ obviously holds. The operator $? : CSubst^*_\perp \to CSubst^?_\perp$ constructs the

$CSubst^?_\perp$ corresponding to a non empty sequence of $CSubst_\perp$, and is defined as $?(\theta_1 \ldots \theta_n)(X) = X$ if $X \notin \bigcup_{i\in\{1,\ldots,n\}} dom(\theta_i)$; $?(\theta_1 \ldots \theta_n)(X) = \rho_1(X)\ ?\ \ldots\ ?\ \rho_m(X)$, where $\rho_1 \ldots \rho_m = \theta_1 \ldots \theta_n \mid \lambda\theta.(X \in dom(\theta))$, otherwise. Then $dom(?(\theta_1 \ldots \theta_n)) = \bigcup_i dom(\theta_i)$. This operator is overloaded to handle non empty sets $\Theta \subseteq CSubst_\perp$ as $?\Theta = ?(\theta_1 \ldots \theta_n)$ where the sequence $\theta_1 \ldots \theta_n$ corresponds to an arbitrary reordering of the elements of Θ.

The $\pi CRWL$-proof calculus is presented in Figure 2. The only difference with the calculus in Figure 1 is that the rule OR has been replaced by **POR** (plural outer reduction), in which we may compute more that one partial value for each argument, and then use a substitution from $CSubst^?_\perp$ instead of $CSubst_\perp$ for parameter passing, achieving a plural semantics[4]. This calculus derives reduction statements of the form $\mathcal{P} \vdash_{\pi CRWL} e \rightarrowtriangle t$ that express that t is (or approximates to) a possible value for e in this semantics, under the program $\mathcal{P}$. The $\pi CRWL$-*denotation* of an expression $e \in Exp_\perp$ under a program $\mathcal{P}$ in $\pi CRWL$ is defined as $[\![e]\!]^{pl}_{\mathcal{P}} = \{t \in CTerm_\perp \mid \mathcal{P} \vdash_{\pi CRWL} e \rightarrowtriangle t\}$.

$$
\textbf{(RR)}\quad \frac{}{X \rightarrowtriangle X}\quad X \in \mathcal{V}
\qquad\qquad
\textbf{(DC)}\quad \frac{e_1 \rightarrowtriangle t_1 \ldots e_n \rightarrowtriangle t_n}{c(e_1,\ldots,e_n) \rightarrowtriangle c(t_1,\ldots,t_n)}\quad c \in CS^n
$$

$$
\textbf{(B)}\quad \frac{}{e \rightarrowtriangle \perp}
\qquad\qquad
\textbf{(POR)}\quad \frac{\begin{array}{ccc} e_1 \rightarrowtriangle p_1\theta_{11} & e_n \rightarrowtriangle p_n\theta_{n1} & \\ \ldots \quad\ \ldots & \ldots & \\ e_1 \rightarrowtriangle p_1\theta_{1m_1} & e_n \rightarrowtriangle p_n\theta_{nm_n} & r\theta \rightarrowtriangle t \end{array}}{f(e_1,\ldots,e_n) \rightarrowtriangle t}
$$

$$
\begin{array}{c}
(f(\overline{p}) \to r) \in \mathcal{P},\ \theta = ?\{\theta_{11},\ldots,\theta_{1m_1}\} \uplus \ldots \uplus ?\{\theta_{n1},\ldots,\theta_{nm_n}\} \\
\forall i,j\ \theta_{ij} \in CSubst_\perp \wedge dom(\theta_{ij}) = var(p_i), \forall i\ m_i > 0
\end{array}
$$

Fig. 2. Rules of $\pi CRWL$

Example 3. Consider the program of Example 1, that is $\mathcal{P} = \{f(c(X)) \to d(X,X), X\ ?\ Y \to X, X\ ?\ Y \to Y\}$. The following is a $\pi CRWL$-proof for the statement $f(c(0)?c(1)) \rightarrowtriangle d(0,1)$ (some steps have been omitted for the sake of conciseness):

$$
\dfrac{
\dfrac{
\dfrac{\dfrac{}{0 \rightarrowtriangle 0}\,DC}{c(0) \rightarrowtriangle c(0)}\,DC
\quad
\dfrac{}{c(1) \rightarrowtriangle \perp}\,B
\quad
c(0) \rightarrowtriangle c(0)
}{c(0)?c(1) \rightarrowtriangle c(0)}\,POR
\quad
c(0)?c(1) \rightarrowtriangle c(1)
\quad
\dfrac{0?1 \rightarrowtriangle 0 \quad 0?1 \rightarrowtriangle 1}{d(0?1,0?1) \rightarrowtriangle d(0,1)}\,DC
}{f(c(0)?c(1)) \rightarrowtriangle d(0,1)}\,POR
$$

$\pi CRWL$ enjoys some nice properties, like the following monotonicity property, where for any proof we define its *size* as the number of applications of rules of the calculus.

Lemma 1. *For any CRWL-program, $e, e' \in Exp_\perp$, $t, t' \in CTerm_\perp$ if $e \sqsubseteq e'$ and $t' \sqsubseteq t$ then $\mathcal{P} \vdash_{\pi CRWL} e \rightarrowtriangle t$ implies $\mathcal{P} \vdash_{\pi CRWL} e' \rightarrowtriangle t'$ with a proof of the same size or smaller.*

One of the most important properties is its compositionality, a property very close to the DET-additivity property for algebraic specifications of [14]:

Theorem 1. *For any CRWL-program, $\mathcal{C} \in Contx$ and $e \in Exp_\perp$, $[\![\mathcal{C}[e]]\!]^{pl} = \bigcup_{\{t_1,\ldots,t_n\} \subseteq [\![e]\!]^{pl}} [\![\mathcal{C}[t_1\ ?\ \ldots\ ?\ t_n]]\!]^{pl}$, for any arrangement of the elements of $\{t_1,\ldots,t_n\}$ in $t_1\ ?\ \ldots\ ?\ t_n$.*

The proof for that theorem is based upon the commutativity, associativity of ?, and the idempotence of its partial application (see Appendix A). With Theorem 1 at hand is very easy to prove the following distributivity property for $\pi CRWL$, also known as the *bubbling* operational rule [2]:

[4] In fact angelic non-strict plural non-determinism.

Theorem 2 (Correctness of bubbling). *For any CRWL-program, $\mathcal{C} \in Contx$ and $e_1, e_2 \in Exp_\perp$, $[\![\mathcal{C}[e_1 \ ?\ e_2]]\!]^{pl} = [\![\mathcal{C}[e_1] \ ?\ \mathcal{C}[e_2]]\!]^{pl}$.*

$\pi CRWL$ also has some monotonicity properties related to substitutions. We define the preorder $\sqsubseteq_\pi$ over $CSubst^?_\perp$ by $\theta \sqsubseteq_\pi \theta'$ iff $\forall X \in \mathcal{V}$, given $\theta(X) = t_1 \ ?\ \ldots\ ?\ t_n$ and $\theta(X) = t'_1 \ ?\ \ldots\ ?\ t'_m$ then $\forall t \in \{t_1, \ldots, t_n\} \exists t' \in \{t'_1, \ldots, t'_m\}$ such that $t \sqsubseteq t'$; and the preorder $\trianglelefteq$ over $Subst_\perp$ by $\sigma \trianglelefteq \sigma'$ iff $\forall X \in \mathcal{V}$, $[\![\sigma(X)]\!]^{pl} \subseteq [\![\sigma'(X)]\!]^{pl}$.

Lemma 2. *For any CRWL-program, $e \in Exp_\perp$, $t \in CTerm_\perp$, $\sigma, \sigma' \in Subst_\perp$, $\theta, \theta' \in CSubst^?_\perp$:*

1. ***Strong monotonicity of $Subst_\perp$**: If $\forall X \in \mathcal{V}, s \in CTerm_\perp$ given $\mathcal{P} \vdash_{\pi CRWL} \sigma(X) \rightarrowtriangle s$ with size K we also have $\mathcal{P} \vdash_{\pi CRWL} \sigma'(X) \rightarrowtriangle s$ with size $K' \leq K$, then $\vdash_{\pi CRWL} e\sigma \rightarrowtriangle t$ with size L implies $\vdash_{\pi CRWL} e\sigma' \rightarrowtriangle t$ with size $L' \leq L$.*
2. ***Monotonicity of $CSubst_\perp$**: If $\theta, \theta' \in CSubst_\perp$ and $\theta \sqsubseteq \theta'$ then $\mathcal{P} \vdash_{\pi CRWL} e\theta \rightarrowtriangle t$ with size K implies $\mathcal{P} \vdash_{\pi CRWL} e\theta' \rightarrowtriangle t$ with size $K' \leq K$.*
3. ***Monotonicity of $Subst_\perp$**: If $\sigma \trianglelefteq \sigma'$ then $[\![e\sigma]\!]^{pl} \subseteq [\![e\sigma']\!]^{pl}$.*
4. ***Monotonicity of $CSubst^?_\perp$**: If $\theta \sqsubseteq_\pi \theta'$ then $[\![e\theta]\!]^{pl} \subseteq [\![e\theta']\!]^{pl}$.*

We end this section with an example of the use of $\pi CRWL$ to model problems in which some collecting work has to be done.

Example 4. We want to represent the database of a bank in which we hold some data about its employees, this bank has several branches and we want to organize the information according to them. So we define a non-deterministic function *branches* to represent the set of branches: a set is identified then with a non-deterministic expression. In this line we define a non-deterministic function *employees* which conceptually returns the set of records containing the information regarding the employees that work in a branch. Now, to search for the names of two clerks we define the function $twoclerks$ which is based upon $find$, which forces the desired pattern $e(N, S, clerk)$ over the set defined by $employees(branches)$.

$$\begin{array}{l}\mathcal{P} = \{branches \to madrid, branches \to vigo, employees(madrid) \to e(pepe, men, clerk), employees(madrid) \to \\ e(paco, men, boss), employees(vigo) \to e(maria, women, clerk), employees(vigo) \to e(jaime, women, boss), \\ twoclerks \to find(employees(branches)), find(e(N, S, clerk)) \to (N, N)\}\end{array}$$

With term rewriting $twoclerks \to find(employees(branches)) \not\to^* (pepe, maria)$, because in that expression the evaluation of *branches* is needed and so one of the branches must be chosen. On the other hand with $\pi CRWL$ (some steps have been omitted for the sake of conciseness):

$$\dfrac{\dfrac{\dfrac{\ldots}{employees(branches) \rightarrowtriangle e(pepe, \perp, clerk)}\,POR \quad \dfrac{\ldots}{employees(branches) \rightarrowtriangle e(maria, \perp, clerk)}\,POR \quad \dfrac{\ldots}{(pepe \ ?\ maria, pepe \ ?\ maria) \rightarrowtriangle (pepe, maria)}\,DC}{find(employees(branches)) \rightarrowtriangle (pepe, maria)}\,POR}{twoclerks \rightarrowtriangle (pepe, maria)}\,POR$$

where

$$\dfrac{\dfrac{}{branches \rightarrowtriangle madrid}\,POR \quad \dfrac{\ldots}{e(pepe, men, clerk) \rightarrowtriangle e(pepe, \perp, clerk)}\,DC}{employees(branches) \rightarrowtriangle e(pepe, \perp, clerk)}\,POR$$

4 Comparison: a hierarchy of semantics

When comparing these semantics is not surprising finding that *CRWL* and $\pi CRWL$ have similar properties. For example the monotonicity Lemma 1 also holds for *CRWL*; this lemma does not even make sense for term rewriting, as it only works with total terms. On the other hand term rewriting is closed under $Subst$ ($e \to^* e'$ implies $e\sigma \to^* e'\sigma$, for any $\sigma \in Subst$); *CRWL* is not closed under $Subst$ but under $CSubst_\perp$, as corresponds to call-time choice; $\pi CRWL$ is closed under $CSubst_\perp$ too (see Appendix A), and we conjecture that for $\theta \in CSubst^?_\perp$ if $\vdash_{\pi CRWL} e \rightarrowtriangle t$ then $[\![t\theta]\!]^{pl} \subseteq [\![e\theta]\!]^{pl}$. For *CRWL* a compositionality result similar to Theorem 1 also holds, and bubbling is correct too [17]. This is not the case for term rewriting, as we saw when switching from $f(c(0?1))$ to $f(c(0)?c(1))$ in examples 1 and 2.

4.1 The hierarchy

As $\pi CRWL$ is a modification of $CRWL$, the relation between them is very direct.

Theorem 3. *For any CRWL-program $\mathcal{P}$, $e \in Exp_\perp, t \in CTerm_\perp$ given a CRWL-proof for $\mathcal{P} \vdash e \rightarrowtriangle t$ we can build a $\pi CRWL$-proof for $\mathcal{P} \vdash_{\pi CRWL} e \rightarrowtriangle t$ just replacing every* ***OR*** *step by the corresponding* ***POR*** *step. As a consequence $[\![e]\!]^{sg}_{\mathcal{P}} \subseteq [\![e]\!]^{pl}_{\mathcal{P}}$.*

Concerning the relation of $CRWL$ and $\pi CRWL$ with term rewriting, we will use the notion of *shell* $|e|$ *of an expression* e that represents the outer constructor (and thus computed) part of e, defined as $|\perp| = \perp$, $|X| = X$, $c(e_1, \ldots, e_n) = c(|e_1|, \ldots, |e_n|)$, $|f(e_1, \ldots, e_n)| = \perp$ (for $X \in \mathcal{V}, c \in CS, f \in FS$). We also define the denotation of $e \in Exp$ under term rewriting as $[\![e]\!]^{rw} = \{t \in CTerm_\perp \mid \exists e' \in Exp \, . \, e \to^* e' \wedge t \sqsubseteq |e'|\}$. In a previous joint work the author explored the relation between $CRWL$ and term rewriting ([15], Theorem 9), recast in the following theorem:

Theorem 4. *For any CRWL-program $\mathcal{P}$, $e \in Exp$, $[\![e]\!]^{sg} \subseteq [\![e]\!]^{rw}$. The converse inclusion does not hold in general.*

As we saw in Example 1, in general call-time choice semantics like $CRWL$ produce less results than run-time choice semantics like the one induced by term rewriting. We will see that this kind of relation also holds for term rewriting and $\pi CRWL$.

Theorem 5. *For any CRWL-program $\mathcal{P}$, $e \in Exp$, $[\![e]\!]^{rw} \subseteq [\![e]\!]^{pl}$. The converse inclusion does not hold in general.*

The key for proving Theorem 5 is a lemma stating that $\forall e, e' \in Exp$ if $e \to e'$ then $[\![e']\!]^{pl} \subseteq [\![e]\!]^{pl}$, that is, that every rewriting step is sound wrt. $\pi CRWL$. The evident corollary for these theorems is the announced inclusion chain.

Corollary 1. *For any CRWL-program $\mathcal{P}$, $e \in Exp$, $[\![e]\!]^{sg} \subseteq [\![e]\!]^{rw} \subseteq [\![e]\!]^{pl}$. Hence $\forall t \in CTerm$, $\vdash_{CRWL} e \rightarrowtriangle t$ implies $e \to^* t$ which implies $\vdash_{\pi CRWL} e \rightarrowtriangle t$.*

5 Simulating plural semantics

In [15, 16] it was shown that neither $CRWL$ can be simulated by term rewriting with a simple program transformation, nor vice versa. Nevertheless, plural semantics can be simulated by rewriting using the transformation presented in the current section, which could be used as the basis for a first implementation of $\pi CRWL$ that we might use for experimentation. First we will present a naive version of this transformation, and show its adequacy; later we will propose some simple optimizations for it.

5.1 A simple transformation

Definition 1. *Given a CRWL-program P, for every rule $(f(p_1, \ldots, p_n) \to r) \in \mathcal{P}$ such that $f \notin \{\_?\_, if\_then\_\}$ we define its transformation as:*

$$pST(f(p_1, \ldots, p_n) \to r) = f(Y_1, \ldots, Y_n) \to if \ match(Y_1, \ldots, Y_n) \ then \ r[\overline{X_{ij}/project_{ij}(Y_i)}]$$

- $\forall i \in \{1, \ldots, n\}$, $\{X_{i1}, \ldots, X_{ik_i}\} = var(p_i) \cap var(r)$ and $Y_i \in \mathcal{V}$ is fresh.
- $match \in FS^n$ fresh is defined by the rule $match(p_1, \ldots, p_n) \to true$.
- Each $project_{ij} \in FS^1$ is a fresh symbol defined by the single rule $project_{ij}(p_i) \to X_{ij}$.

For $f \in \{\_?\_, if\_then\_\}$ the transformation leaves its rules untouched.

The function *match* is used to impose a "guard" that enforces the matching of each argument with its corresponding pattern. If we dropped this condition the translation of, for example, to rule $(null(nil) \to true)$, would be $(null(Y) \to true)$, which is clearly unsound as then $null(0 : nil) \to true$. Besides each pattern p_i

has been replaced by a fresh variable Y_i, to which any expression can match, hence the arguments may be replicated freely by the rewriting process without demanding any evaluation and thus keeping its denotation untouched: this is the key to achieve completeness wrt. $\pi CRWL$. Later on, the functions $project_{ij}$ will just make the projection of each variable when needed.

Example 5. Applying this to Example 1 we get
$\{f(Y) \to if\ match(Y)\ then\ d(project(Y),\ project(Y)), match(c(X)) \to true, project(c(X)) \to X\}$
under which we can do:
$f(c(0)?c(1)) \to if\ match(c(0)?c(1))\ then\ d(project(c(0)?c(1)), project(c(0)?c(1)))$
$\to^* if\ true\ then\ d(project(c(0)?c(1)), project(c(0)?c(1)))$
$\to d(project(c(0)?c(1)), project(c(0)?c(1))) \to^* d(project(c(0)), project(c(1))) \to^* d(0,1)$

Concerning the adequacy of the transformation:

Theorem 6. *For any CRWL-program $\mathcal{P}$, $e \in Exp_\perp$ built up on the signature of $\mathcal{P}$, $[\![e]\!]^{pl}_{pST(\mathcal{P})} \subseteq [\![e]\!]^{pl}_{\mathcal{P}}$.*

Theorem 7. *For any CRWL-program $\mathcal{P}$, $e \in Exp, t \in CTerm_\perp$ built up on the signature of $\mathcal{P}$, if $\mathcal{P} \vdash_{\pi CRWL} e \twoheadrightarrow t$ then exists some $e' \in Exp$ built using symbols of the signature of $pST(\mathcal{P})$ such that $pST(\mathcal{P}) \vdash e \to^* e'$ and $t \sqsubseteq |e'|$.*

Corollary 2. *For any CRWL-program $\mathcal{P}$, $e \in Exp$ built using symbols of the signature of $\mathcal{P}$, $[\![e]\!]^{pl}_{\mathcal{P}} = [\![e]\!]^{rw}_{pST(\mathcal{P})}$. Hence $\forall t \in CTerm$ $\mathcal{P} \vdash_{\pi CRWL} e \twoheadrightarrow t$ iff $pST(\mathcal{P}) \vdash e \to^* t$.*

5.2 An optimized transformation

Concerning the transformation, if a pattern is ground then no parameter passing will be done for it and so no transformation is needed: for $null(nil) \to true$ we get $\{null(Y) \to if\ match(Y)\ then\ true, match(nil) \to true\}$, which is equivalent. Besides, if the pattern is a variable then any expression matches it and the projection functions are trivial, so no transformation is needed neither, as happens with $pair(X) \to (X, X)$ for which $\{pair(Y) \to if\ match(Y)\ then\ (project(Y),\ project(Y)), match(X) \to true, project(X) \to X\}$ are returned.

Definition 2. *Given a CRWL-program P, for every rule $(f(p_1, \ldots, p_n) \to r) \in \mathcal{P}$ we define its transformation as:*

$$pST(f(p_1, \ldots, p_n) \to r)$$
$$= \begin{cases} f(p_1, \ldots, p_n) \to r & \textit{if } \rho_1 \ldots \rho_m \textit{ is empty} \\ f(\tau(p_1), \ldots, \tau(p_n)) \to \begin{array}{l} if\ match(Y_1, \ldots, Y_m) \\ \quad then\ r[X_{ij}/project_{ij}(Y_i)] \end{array} & \textit{otherwise} \end{cases}$$

where $\rho_1 \ldots \rho_m = p_1 \ldots p_n \mid \lambda p.(p \notin \mathcal{V} \wedge var(p) \neq \emptyset)$.
- $\forall \rho_i$, $\{X_{i1}, \ldots, X_{ik_i}\} = var(\rho_i) \cap var(r)$ and $Y_i \in \mathcal{V}$ is fresh.
- $\tau : CTerm \to CTerm$ is defined by $\tau(p) = p$ if $p \in \mathcal{V} \vee var(p) = \emptyset$ and $\tau(p) = Y_i$ otherwise, for $p \equiv \rho_i$.
- $match \in FS^m$ fresh is defined by the rule $match(\rho_1, \ldots, \rho_m) \to true$.
- Each $project_{ij} \in FS^1$ is a fresh symbol defined by the single rule $project_{ij}(\rho_i) \to X_{ij}$.

We will not give a formal proof for the adequacy of the optimization. Nevertheless note how this transformation leaves untouched the rules for $\_?\_$ and $if\_then\_$ without defining an special case for them. As the simple transformation worked well for that rules that suggests that we are doing the right thing. We end this section with an example application of the optimized transformation, over the program of Example 4:

Example 6. The only rule modified is the one for $find$, for which we get $\{find(Y) \to if\ match(Y)\ then\ (project(Y), project(Y)), match(e(N, s, clerk)) \to true, project(e(N, s, clerk)) \to N\}$ so:

$twoclerks \to find(employees(branches))$
$\to if\ match(employees(branches))\ then\ (project(employees(branches)), project(employees(branches)))$
$\to^* if\ match(e(pepe, men, clerk))\ then\ (project(employees(branches)), project(employees(branches)))$
$\to^* (project(employees(branches)), project(employees(branches)))$
$\to^* (project(e(pepe, men, clerk)), project(e(maria, women, clerk))) \to^* (pepe, maria)$

6 Conclusions

In this work we have pointed the different interpretations of run-time choice and plural semantics caused by pattern matching. To the best of our knowledge this distinction is stablished in the present paper for the first time, because in [23] no pattern matching was present and in [14] only call-time choice was adopted. We argue that the run-time choice semantics induced by term rewriting is not the best option for a value-based programming language like current implementations of FLP. For that context a plural semantics has been proposed for which the compositionality properties lost when turning from call-time choice to rewriting are recovered. Nevertheless, for other kind of rewriting based languages like Maude, which are not limited to constructor-based TRS's, term rewriting has been proven to be an effective formalism.

Our concrete contributions can be summarized as follows:

- We have presented the proof calculus $\pi CRWL$, a novel formulation of plural semantics for left-linear constructor-based TRS's, which are the kind of TRS's used in FLP. Some basic properties of the new semantics have been stated and proved, and by some examples we have shown how it allows natural encodings of some programs that need to do some collecting work (Sect. 3).
- We have compared the new calculus with *CRWL* and term rewriting, which are standard formulations for call-time choice and run-time choice respectively. The different properties of these calculi have been discussed and the inclusion chain $CRWL \subseteq$ rewriting $\subseteq \pi CRWL$ has been proved (Sect. 4).
- We have recalled some previous results about the impossibility of a straight simulation of *CRWL* in term rewriting or viceversa by a simple program transformation. Besides we have proposed a novel program transformation to simulate plural semantics with term rewriting, and proved its adequacy (Sect. 5).

From a practical point of view, it might be unrealistic to think that a monolithic semantic view is adequate for addressing all non-determinism present in a large program. In [16] we have started to investigate the combination of call-time choice and run-time choice in a unified framework. But as $\pi CRWL$ seems to be more suitable than run-time choice for a value-based language, we are planning to extend that work to plural semantics.

We contemplate other relevant subjects of future work:

- Extending the current results to programs with extra variables, that is, with rules $l \to r$ in which $var(r) \subseteq var(l)$ does not hold in general. We should also deal with conditional rules and equality constraints to cover the basic features of FLP languages.
- Studying the relation between the determinism of programs under *CRWL* [15] and $\pi CRWL$, which we conjecture is equivalent. We also conjecture that for deterministic programs $\forall e \in Exp, [\![e]\!]^{sg} = [\![e]\!]^{rw} = [\![e]\!]^{pl}$. Getting results about the relation of confluence and determinism of programs could be useful for analyzing the confluence of a TRS through its determinism. In the same line, the inclusion chain $CRWL \subseteq$ rewriting $\subseteq \pi CRWL$ could be used to study the termination of a TRS through its termination in *CRWL* and $\pi CRWL$.
- Developing a more operational rewrite notion for $\pi CRWL$ in the line of [15], which could be extended to narrowing like in [17]. A complexity study would be needed to ensure that the extra nondeterminism does not preclude the design of an efficient implementation. On the other hand the natural value for $\pi CRWL$ seems to be $\mathcal{P}(CTerm_\perp)$ instead of $CTerm_\perp$, a formulation in the line of [19] could be useful to forget about the tricky use of _?_.
- Finally, for the immediate future, it could be interesting implementing the transformation to simulate $\pi CRWL$ in some term rewriting based language like Maude [5]. Maybe the context-sensitive rewriting [21] features of Maude could be used to improve the laziness of the transformed program like in [20]. Besides, the matching-module capacities of Maude could be used to enhance the expressivity of plural semantics.

Acknowledgements: The author would like to thank Paco López Fraguas and Jaime Sánchez Hernández for their support and their useful suggestions. I would also like to thank the referees for their very valuable comments.

References

1. S. Antoy. Optimal non-deterministic functional logic computations. In *Proc. ALP'97*, pages 16–30. Springer LNCS 1298, 1997.

2. S. Antoy, D. Brown, and S. Chiang. Lazy context cloning for non-deterministic graph rewriting. In Proc. Termgraph'06, pages 61–70. ENTCS, 176(1), 2007.
3. S. Antoy and M. Hanus. Functional logic design patterns. In Proc. FLOPS'02, pages 67–87. Springer LNCS 2441, 2002.
4. F. Baader and T. Nipkow. Term Rewriting and All That. Cambridge University Press, United Kingdom, 1998.
5. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, editors. All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic. Springer LNCS 4350, 2007
6. M. Clavel, M. Palomino, and A. Riesco. Introducing the itp tool: a tutorial. J. UCS 12(11), pages 1618–1650, 2006.
7. E. W. Dijkstra. A Discipline of Programming. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.
8. R. Diaconescu and K. Futatsugi. An overview of CafeOBJ. ENTCS 15,1998.
9. J. C. González-Moreno, T. Hortalá-González, F. López-Fraguas, and M. Rodríguez-Artalejo. A rewriting logic for declarative programming. In Proc. ESOP'96, pages 156–172. Springer LNCS 1058, 1996.
10. J. C. González-Moreno, T. Hortalá-González, F. López-Fraguas, and M. Rodríguez-Artalejo. An approach to declarative programming based on a rewriting logic. J. Log. Program. 40(1), pages 47–87, 1999.
11. M. Hanus. The integration of functions into logic programming: From theory to practice. J. Log. Program. 19/20, pages 583–628, 1994.
12. M. Hanus. Multi-paradigm declarative languages. In Proc. ICLP'07, pages 45–75. Springer LNCS 4670, 2007.
13. M. Hanus (ed.). Curry: An integrated functional logic language (version 0.8.2). Available at *http://www.informatik.uni-kiel.de/~curry/report.html*, March 2006.
14. H. Hussmann. Non-Determinism in Algebraic Specifications and Algebraic Programs. Birkhäuser Verlag, 1993.
15. F. López-Fraguas, J. Rodríguez-Hortalá, and J. Sánchez-Hernández. A simple rewrite notion for call-time choice semantics. In Proc. PPDP'07, pages 197–208. ACM Press, 2007.
16. F. López-Fraguas, J. Rodríguez-Hortalá, and J. Sánchez-Hernández. A flexible framework for programming with non-deterministic functions (Extended version). Tech. Rep. SIC-9-08, Universidad Complutense de Madrid, 2008. `http://gpd.sip.ucm.es/juanrh/pubs/tchrRTCT08.pdf`.
17. F. López-Fraguas, J. Rodríguez-Hortalá, and J. Sánchez-Hernández. Rewriting and call-time choice: the HO case. In Proc. FLOPS'08, pages 147–162. Springer LNCS 4989, 2008.
18. F. López-Fraguas and J. Sánchez-Hernández. $\mathcal{TOY}$: A multiparadigm declarative system. In Proc. RTA'99, pages 244–247. Springer LNCS 1631, 1999.
19. F. López-Fraguas and J. Sánchez-Hernández. A proof theoretic approach to failure in functional logic programming. TPLP, 4(1&2), pages 41–74, 2004.
20. S. Lucas. Needed reductions with context-sensitive rewriting. In Proc. ALP/HOA'97, pages 129–143. Springer LNCS 1298, 1997.
21. S. Lucas. Context-sensitive computations in functional and functional logic programs. J. Fun. Log. Program 1998(1), 1998.
22. J. McCarthy. A Basis for a Mathematical Theory of Computation. Computer Programming and Formal Systems, pages 33–70. North-Holland, Amsterdam, 1963.
23. H. Søndergaard and P. Sestoft. Non-determinism in functional languages. The Computer Journal 35(5), pages 514–523, 1992.
24. The Coq Development Team, LogiCal Project. The coq proof assistant reference manual version 8.1. Technical report, INRIA, 2006. `http://pauillac.inria.fr/coq/V8.1pl3/refman`.
25. M. Wenzel. The isabelle/isar reference manual. `http://isabelle.in.tum.de/dist/Isabelle99-2/doc/isar-ref.pdf`.

A Proofs of the results

During the proofs we will often use the notation *IH* to refer to the induction hypothesis. We will also use the following notions:

Definition 3.

- *The set of positions of an expression e is the set $\mathcal{O}(e)$ of strings over the alfabet of positive integers defined as $\mathcal{O}(X) = \{\epsilon\}$, if $X \in \mathcal{V}$; $\mathcal{O}(h(e_1, \dots, e_n)) = \{\epsilon\} \cup \bigcup_{i \in \{1,\dots,n\}} \{i.p \mid p \in \mathcal{O}(e_i)\}$ otherwise, where ϵ denotes the empty string and $\_.\_$ is used for concatenation.*
  We say that two positions are parallel if none of them is prefix of the other.
- *For any $e \in Exp_\perp$, $p \in \mathcal{O}(()e)$, the subexpression of e at position p denoted by $s|_p$, is defined as $e|_\epsilon = e$; $h(e_1, \dots, e_n)|_{i.q} = e_i|_q$.*
- *For any $e, e' \in Exp_\perp$, $p \in \mathcal{O}(e)$, by $e[e']_p$ we denote the expression obtained from e by replacing the subexpression at position p by e', defined as $e[e']_\epsilon = e'$; $f(e_1, \dots, e_n)[e']_{i.q} = f(e_1, \dots, e_i[e']_q, \dots, e_n)$.*
- *As one-hole context can be understood as funcions $\mathcal{C} : Exp_\perp \to Exp_\perp$, for any $\mathcal{C} \in Cntx$ we may assume some $e \in Exp_\perp, p \in \mathcal{O}(e)$ such that $\mathcal{C} = \lambda e'.e[e']_p$. With this in mind Theorem 1 can be recasted as $[\![e[e']_p]\!]^{pl} = \bigcup_{\overline{t} \subseteq [\![e']\!]^{pl}} [\![e[?\overline{t}]_p]\!]^{pl}$, where $?\{t_1, \dots, t_n\}$ denotes $t_1 \ ? \ \dots \ ? \ t_n$ for some arrangement of the elements of $\{t_1, \dots, t_n\}$ in $t_1 \ ? \ \dots \ ? \ t_n$.*

A.1 For Section 3

The following auxiliary lemma will be used in the proofs:

Lemma 3. *For any πCRWL-program, $t, t' \in CTerm_\perp$, $e \in Exp_\perp$, $\sigma, \sigma' \in Subst_\perp$*

1. *$\mathcal{P} \vdash_{\pi CRWL} t \to t$*
2. *$\mathcal{P} \vdash_{\pi CRWL} t \to t'$ iff $t' \sqsubseteq t$.*
3. *If $\sigma \sqsubseteq \sigma'$ then $e\sigma \sqsubseteq e\sigma'$.*

Proof.

1. By a simple induction on the structure of t
2. Assume $\mathcal{P} \vdash_{\pi CRWL} t \to t'$, we can prove $t' \sqsubseteq t$ by a simple induction on the structure of t. For the converse implication assume $t' \sqsubseteq t$, then $t \to t$ by the previuos item, hence $t \to t'$ by Lemma 1.
3. A simple induction on the structure of e.

Proof (For Lemma 1). By induction on the structure of $e \to t$.

Base cases

B $e \to \perp \equiv t$. Then $t' \sqsubseteq t$ implies $t' \equiv \perp$, so $e' \to \perp \equiv t'$, by B.

RR $e \equiv X \to X \equiv t$. Then $t' \sqsubseteq t$ implies $t' \equiv \perp$ or $t' \equiv X$. In the first case we proceed like in the case for B, in the latter as $X \equiv e \sqsubseteq e'$ implies $e' \equiv X$ then $e' \equiv X \to X \equiv t'$ by RR.

DC $e \equiv c \to c \equiv t$. We can proceed in similar way we did in the previous case.

Inductive steps

DC Then we have

$$\frac{e_1 \to t_1 \ \dots \ e_n \to t_n}{e \equiv c(e_1, \dots, e_n) \to c(t_1, \dots, t_n) \equiv t} \ DC$$

Then $t' \sqsubseteq t$ implies $t' \equiv \perp$ or $t' \equiv c(t'_1, \dots, t'_n)$ with $t'_i \sqsubseteq t_i$ for every $i \in \{1, \dots, n\}$. In the first case we proceed like in the case for B, in the latter as $c(e_1, \dots, e_n) \equiv e \sqsubseteq e'$ implies $e' \equiv c(e'_1, \dots, e'_n)$ with $e_i \sqsubseteq e'_i$ for every $i \in \{1, \dots, n\}$ then by IH $e'_i \to t'_i$ for every $i \in \{1, \dots, n\}$, and we can build the following proof:

$$\frac{e'_1 \to t_1 \ \dots \ e'_n \to t_n}{e' \equiv c(e'_1, \dots, e'_n) \to c(t'_1, \dots, t'_n) \equiv t'} \ DC$$

POR Then we have

$$\dfrac{\begin{array}{ccccc} e_1 \twoheadrightarrow p_1\theta_{11} & & e_n \twoheadrightarrow p_n\theta_{n1} & \\ \dots & \dots & \dots & \\ e_1 \twoheadrightarrow p_1\theta_{1m_1} & & e_n \twoheadrightarrow p_n\theta_{nm_n} & r\theta \twoheadrightarrow t \end{array}}{e \equiv f(e_1,\dots,e_n) \twoheadrightarrow t}\ POR$$

with $\theta = ?(\theta_{11}\dots\theta_{1m_1}) \uplus \dots \uplus ?(\theta_{n1}\dots\theta_{nm_n})$, for some $(f(p_1,\dots,p_n) \to r) \in \mathcal{P}$. Then as $f(e_1,\dots,e_n) \equiv e \sqsubseteq e'$ implies $e' \equiv f(e'_1,\dots,e'_n)$ with $e_i \sqsubseteq e'_i$ for every $i \in \{1,\dots,n\}$ then by IH $\forall i \in \{1,\dots,n\}, \forall j \in \{1,\dots,m_i\}$ $e'_i \twoheadrightarrow p_i\theta_{ij}$. We can also apply the IH to get $r\theta \twoheadrightarrow t'$, as $t' \sqsubseteq t$ by hypothesis, and build the following proof:

$$\dfrac{\begin{array}{ccccc} e'_1 \twoheadrightarrow p_1\theta_{11} & & e'_n \twoheadrightarrow p_n\theta_{n1} & \\ \dots & \dots & \dots & \\ e'_1 \twoheadrightarrow p_1\theta_{1m_1} & & e'_n \twoheadrightarrow p_n\theta_{nm_n} & r\theta \twoheadrightarrow t' \end{array}}{e' \equiv f(e'_1,\dots,e'_n) \twoheadrightarrow t'}\ POR$$

Lemma 4. *For any CRWL-program, $\mathcal{C} \in Contx$ and $e_1, e_2, e_3 \in Exp_\perp$:*

1. $[\![\mathcal{C}[e1\ ?\ e_2]]\!]^{pl} = [\![\mathcal{C}[e_2\ ?\ e_1]]\!]^{pl}$
2. $[\![\mathcal{C}[(e1\ ?\ e_2)\ ?\ e_3]]\!]^{pl} = [\![\mathcal{C}[e_1\ ?\ (e_2\ ?\ e_3)]]\!]^{pl}$
3. $[\![\mathcal{C}[e_1\ ?\ e_1]]\!]^{pl} = [\![\mathcal{C}[e_1]]\!]^{pl}$
4. $[\![\mathcal{C}[e_1]]\!]^{pl} \subseteq [\![\mathcal{C}[e_1\ ?\ e_2]]\!]^{pl}$. *As a consequence, for any pair of finite chains $a_1 \dots a_n \in Exp^*_\perp$, $b_1 \dots b_m \in Exp^*_\perp$ if $\{a_1,\dots,a_n\} \subseteq \{b_1,\dots,b_m\}$ then for any context $\mathcal{C}$, $[\![\mathcal{C}[a_1\ ?\ \dots\ ?\ a_n]]\!]^{pl} \subseteq [\![\mathcal{C}[b_1\ ?\ \dots\ ?\ b_m]]\!]^{pl}$ holds.*

Proof (For Lemma 4 (Sketch)).

1. We have to prove that for any $t \in CTerm_\perp$ if $\mathcal{C}[e_1\ ?\ e_2] \twoheadrightarrow t$ then $\mathcal{C}[e_2\ ?\ e_1] \twoheadrightarrow t$ and vice versa. This can be easily done with a simple induction on the size of the proof which acts as hypothesis.
2. Similar to the previous item.
3. Similar to the previous item.
4. Assume $\mathcal{C}[e_1] \twoheadrightarrow t$, we can prove that then $\mathcal{C}[e_1\ ?\ e_2] \twoheadrightarrow t$ with a simple induction on the size of the proof for $\mathcal{C}[e_1] \twoheadrightarrow t$. Regarding the second part of this item, assume $\mathcal{C}[a_1\ ?\ \dots\ ?\ a_n] \twoheadrightarrow t$, then by the previuos items we may eliminate repeated elements in the chains and arrange them in way such that $b_1 \dots b_m \equiv a_1 \dots a_n b'_1 \dots b'_k$ con $n + k = m$. Then by the first part of this item $[\![\mathcal{C}[a_1\ ?\ \dots\ ?\ a_n]]\!]^{pl} \subseteq [\![\mathcal{C}[a_1\ ?\ \dots\ ?\ a_n\ ?\ b'_1]]\!]^{pl} \subseteq \dots \subseteq [\![\mathcal{C}[a_1\ ?\ \dots\ ?\ a_n\ ?\ b'_1\ ?\ \dots\ ?\ b'_k]]\!]^{pl} = [\![\mathcal{C}[b_1\ ?\ \dots\ ?\ b_m]]\!]^{pl}$, and this process ends because both chains are finite.

Proof (For Theorem 1). First we will prove that for any $t \in CTerm_\perp$, if $\mathcal{C}[e] \twoheadrightarrow t$ then $\exists\{s_1,\dots,s_n\} \subseteq [\![e]\!]^{pl}$ such that $\mathcal{C}[s_1\ ?\ \dots\ ?\ s_n] \twoheadrightarrow t$, we proceed by induction on the size K of the proof for $\mathcal{C}[e] \twoheadrightarrow t$, measured as the number of rules of the calculus applied.

Base cases $K = 1$:

B Then we can take $\{s_1,\dots,s_n\} = \{\perp\}$ to do $\mathcal{C}[\perp] \twoheadrightarrow \perp$, by **B**.

RR, DC These cases correspond to $X \twoheadrightarrow X$ by **RR** and $c \twoheadrightarrow c$ by **DC**. Then $\mathcal{C} = []$ and so the hypothesis was $e \equiv \mathcal{C}[e] \twoheadrightarrow t$. Hence we can take $\{s_1,\dots,s_n\} = \{t\}$ to do $\mathcal{C}[s_1\ ?\ \dots\ ?\ s_n] \equiv [t] \equiv t \twoheadrightarrow t$, by Lemma 3.

Inductive steps $K > 1$:

DC If $\mathcal{C} = []$ then we are done like in the previous step. Otherwise we have:

$$\dfrac{e_1 \twoheadrightarrow t_1\ \dots\ \mathcal{C}'[e] \twoheadrightarrow t'\ \dots\ e_l \twoheadrightarrow t_l}{\mathcal{C}[e] \equiv c(e_1,\dots,\mathcal{C}'[e],\dots,e_l) \twoheadrightarrow c(t_1,\dots,t',\dots,t_l) \equiv t}\ DC$$

Then by IH $\exists\{s_1,\ldots,s_n\} \subseteq [\![e]\!]^{pl}$ such that $\mathcal{C}'[s_1 \ ? \ \ldots \ ? \ s_n] \twoheadrightarrow t'$, therefore we can build the following proof:

$$\dfrac{\dfrac{hypothesis}{e_1 \twoheadrightarrow t_1} \quad \ldots \quad \dfrac{IH}{\mathcal{C}'[s_1 \ ? \ \ldots \ ? \ s_n] \twoheadrightarrow t'} \quad \ldots \quad \dfrac{hypothesis}{e_l \twoheadrightarrow t_l}}{\mathcal{C}[s_1 \ ? \ \ldots \ ? \ s_n] \equiv c(e_1,\ldots,\mathcal{C}'[s_1 \ ? \ \ldots \ ? \ s_n],\ldots,e_l) \twoheadrightarrow c(t_1,\ldots,t',\ldots,t_l) \equiv t} \ DC$$

POR If $\mathcal{C} = []$ then we are done like in the previous step. Otherwise we have:

$$\dfrac{\begin{array}{ccccccc} e_1 \twoheadrightarrow p_1\theta_{11} & & \mathcal{C}'[e] \twoheadrightarrow p'\theta'_1 & & e_l \twoheadrightarrow p_l\theta_{l1} & \\ \ldots & \ldots & \ldots & \ldots & \ldots & \\ e_1 \twoheadrightarrow p_1\theta_{1m_1} & & \mathcal{C}'[e] \twoheadrightarrow p'\theta'_{m'} & & e_l \twoheadrightarrow p_l\theta_{lm_l} & r\theta \twoheadrightarrow t \end{array}}{\mathcal{C}[e] \equiv f(e_1,\ldots,\mathcal{C}'[e],\ldots,e_l) \twoheadrightarrow t} \ POR$$

with $\theta = ?(\theta_{11}\ldots\theta_{1m_1}) \uplus \ldots \uplus ?(\theta'_1\ldots\theta'_{m'}) \uplus \ldots \uplus ?(\theta_{l1}\ldots\theta_{lm_l})$ for some $(f(p_1,\ldots,p',\ldots,p_l) \to r) \in \mathcal{P}$. Then by IH for each $\theta'_i \in \{\theta'_1,\ldots,\theta'_{m'}\}$ $\exists\{s_{i1},\ldots,s_{in_i}\} \subseteq [\![e]\!]^{pl}$ such that $\mathcal{C}'[s_{i1} \ ? \ \ldots \ ? \ s_{in_i}] \twoheadrightarrow p'\theta'_i$, hence $\mathcal{C}'[s_{11} \ ? \ \ldots \ ? \ s_{1n_1} \ ? \ \ldots \ ? \ s_{m'1} \ ? \ \ldots \ ? \ s_{m'n_{m'}}] \twoheadrightarrow p'\theta'_i$ by Lemma 4. Therefore we can take:

$$\begin{array}{c} \{s_1,\ldots,s_n\} = \{s_{11} \ ? \ \ldots \ ? \ s_{1n_1} \ ? \ \ldots \ ? \ s_{m'1} \ ? \ \ldots \ ? \ s_{m'n_{m'}}\} \\ a \equiv \mathcal{C}'[s_{11} \ ? \ \ldots \ ? \ s_{1n_1} \ ? \ \ldots \ ? \ s_{m'1} \ ? \ \ldots \ ? \ s_{m'n_{m'}}] \end{array}$$

to build the following proof:

$$\dfrac{\begin{array}{ccccccc} e_1 \twoheadrightarrow p_1\theta_{11} & & a \twoheadrightarrow p'\theta'_1 & & e_l \twoheadrightarrow p_l\theta_{l1} & \\ \ldots & \ldots & \ldots & \ldots & \ldots & \\ e_1 \twoheadrightarrow p_1\theta_{1m_1} & & a \twoheadrightarrow p'\theta'_{m'} & & e_l \twoheadrightarrow p_l\theta_{lm_l} & r\theta \twoheadrightarrow t \end{array}}{\mathcal{C}[s_1 \ ? \ \ldots \ ? \ s_n] \equiv f(e_1,\ldots,a,\ldots,e_l) \twoheadrightarrow t} \ POR$$

Now we have to prove the other implication, that is, given $\{s_1,\ldots,s_n\} \subseteq [\![e]\!]^{pl}$ such that $\mathcal{C}[s_1 \ ? \ \ldots \ ? \ s_n] \twoheadrightarrow t$ then $\mathcal{C}[e] \twoheadrightarrow t$. If $\mathcal{C}[]$ then the hypothesis is $s_1 \ ? \ \ldots \ ? \ s_n \twoheadrightarrow t$, so it must exists some $s_i \in \{s_1,\ldots,s_n\}$ such that $s_i \twoheadrightarrow t$. But then $t \sqsubseteq s_i$ by Lemma 3, as $s_i \in CTerm_\perp$, as it is a value for e. But then the hypothesis $e \twoheadrightarrow s_i$ and $t \sqsubseteq s_i$ implies $e \twoheadrightarrow t$ by the monotonicity of Lemma 1.
To prove the case when $\mathcal{C} \neq []$ we need to do a simple induction on the size of $\mathcal{C}[s_1 \ ? \ \ldots \ ? \ s_n] \twoheadrightarrow t$, in which we will not assume $\mathcal{C} \neq []$.

The following auxiliary lemma will be used in the proof for Theorem 2:

Lemma 5. *For any CRWL-program,* $e_1, e_2 \in Exp_\perp$, $[\![e_1 \ ? \ e_2]\!]^{pl} = [\![e_1]\!]^{pl} \cup [\![e_2]\!]^{pl}$.

Proof. Assume $e_1 \ ? \ e_2 \twoheadrightarrow t$ for some $t \in CTerm_\perp$. If the proof has B in its root then $t \equiv \perp$ and $\perp \in [\![e_1]\!]^{pl}$ trivially. Otherwise the proof is:

$$\dfrac{e_i \twoheadrightarrow t_i \quad e_{3-i} \twoheadrightarrow t_{3-i} \quad t_i \twoheadrightarrow t}{e_1 \ ? \ e_2 \twoheadrightarrow t} \ POR$$

with $i \in \{1,2\}$. Then $t \sqsubseteq t_i$ by Lemma 3 and so $e_i \twoheadrightarrow t$ by Lemma 1, hence $t \in [\![e_i]\!]^{pl} \subseteq [\![e_1]\!]^{pl} \cup [\![e_2]\!]^{pl}$.
On the other hand if $t \in [\![e_1]\!]^{pl} \cup [\![e_2]\!]^{pl}$ then $t \in [\![e_i]\!]^{pl}$ for some $i \in \{1,2\}$, hence we can apply POR to get t from $e_1 \ ? \ e_2$.

Proof (For Theorem 2). By Lemma 5 we already have $[\![e_1 \ ? \ e_2]\!]^{pl} = [\![e_1]\!]^{pl} \cup [\![e_2]\!]^{pl}$. Then we can apply this fact, Theorem 1 and basic set reasoning to get

$$\begin{array}{l} [\![\mathcal{C}[e_1 \ ? \ e_2]]\!]^{pl} = \bigcup_{\{t_1,\ldots,t_n\} \subseteq [\![e_1 \ ? \ e_2]\!]^{pl}} [\![\mathcal{C}[t_1 \ ? \ \ldots \ ? \ t_n]]\!]^{pl} = \bigcup_{\{t_1,\ldots,t_n\} \subseteq [\![e_1]\!]^{pl} \ \cup \ [\![e_2]\!]^{pl}} [\![\mathcal{C}[t_1 \ ? \ \ldots \ ? \ t_n]]\!]^{pl} \\ = \bigcup_{\{t_1,\ldots,t_n\} \subseteq [\![e_1]\!]^{pl}} [\![\mathcal{C}[t_1 \ ? \ \ldots \ ? \ t_n]]\!]^{pl} \ \cup \ \bigcup_{\{t_1,\ldots,t_n\} \subseteq [\![e_2]\!]^{pl}} [\![\mathcal{C}[t_1 \ ? \ \ldots \ ? \ t_n]]\!]^{pl} \\ = [\![\mathcal{C}[e_1]]\!]^{pl} \cup [\![\mathcal{C}[e_1]]\!]^{pl} = [\![\mathcal{C}[e_1] \ ? \ \mathcal{C}[e_2]]\!]^{pl} \end{array}$$

Some facts about the preorders $\sqsubseteq_\pi$ and $\trianglelefteq$:

Lemma 6.

1. *$\forall \theta, \theta' \in CSusbt_\perp$, $\theta \sqsubseteq \theta'$ iff $\theta \sqsubseteq_\pi \theta'$.*
2. *$\sqsubseteq_\pi$: $CSubst^?_\perp \times CSubst^?_\perp$ is a preorder but not a partial order.*
3. *Given $\theta, \theta' \in CSubst^?_\perp$ if $\theta \sqsubseteq_\pi \theta'$ then $\theta \trianglelefteq \theta'$*
4. *$\trianglelefteq : Subst_\perp \times Subst_\perp$ is a preorder but not a partial order*

Proof.

1. By definition.
2. It is very easy to check that it is reflexive and transitive, but it is not antisymetric, as $[X/0] \sqsubseteq_\pi [X/0 \ ?\ 0]$, $[X/0 \ ?\ 0] \sqsubseteq_\pi [X/0]$ but $[X/0] \neq [X/0 \ ?\ 0]$.
3. For any $X \in \mathcal{V}$, if $\theta(X) = t_1 \ ?\ \dots\ ?\ t_n$ and $\theta'(X) = t'_1 \ ?\ \dots\ ?\ t'_m$ (note that $\theta(X)$ has that shape even when $X \notin dom(\theta)$, as X has that shape) and $\theta(X) \rightarrowtriangle t$ then it must exist some $t_i \in \{t_1, \dots, t_n\}$ such that $t_i \rightarrowtriangle t$, and so $t \sqsubseteq t_i$ by Lemma 3. But then $t \sqsubseteq t_i \sqsubseteq t'_j$ for some $t_j \in \{t'_1, \dots, t'_m\}$, because $\theta \sqsubseteq_\pi \theta'$, hence $t'_j \rightarrowtriangle t$ by Lemma 3 and so $\theta'(X) \rightarrowtriangle t$.
4. It is very easy to check that it is reflexive and transitive, but it is not antisymetric, because given $\mathcal{P} = \{f \rightarrow 1, g \rightarrow 1\}$ we have $[X/f] \trianglelefteq [X/g]$ and $[X/g] \trianglelefteq [X/f]$ while $[X/f] \neq [X/g]$.

Proof (For Lemma 2).

1. If $e \equiv X \in \mathcal{V}$, assume $e\sigma \equiv \sigma(X) \rightarrowtriangle t$, then $e\sigma' \equiv \sigma'(X) \rightarrowtriangle t$ with a proof of the same size or smaller, by hypothesis. Otherwise we proceed by induction on the structure of $e\sigma \rightarrowtriangle t$.

   Base cases

   B Then $t \equiv \perp$ and $e\sigma' \rightarrowtriangle \perp$ with a proof of size 1 just applying rule B.

   RR Then $e \in \mathcal{V}$ and we are in the previous case.

   DC Then $e \equiv c \in CS^0$, as $e \notin \mathcal{V}$, hence $e\sigma \equiv c \equiv e\sigma'$ and every proof for $e\sigma \rightarrowtriangle t$ is a proof for $e\sigma' \rightarrowtriangle t$.

   Inductive steps

   DC Then $e \equiv c(e_1, \dots, e_n)$, as $e \notin \mathcal{V}$, and we have:

   $$\frac{e_1\sigma \rightarrowtriangle t_1 \ \dots\ e_n\sigma \rightarrowtriangle t_n}{e\sigma \equiv c(e_1\sigma, \dots, e_n\sigma) \rightarrowtriangle c(t_1, \dots, t_n) \equiv t} \ DC$$

   By IH or the proof of the other cases $\forall i \in \{1, \dots, n\}$ we have $e_i\sigma' \rightarrowtriangle t_i$ with a proof of the same size or smaller, so we can built a proof for $e\sigma' \equiv c(e_1\sigma', \dots, e_n\sigma') \rightarrowtriangle c(t_1, \dots, t_n) \equiv t$ using DC, with a size equal or smaller than the size of the starting proof.

   OR Very similar to the proof of the previuos case. We also have $e \equiv f(e_1, \dots, e_n)$ (as $e \notin \mathcal{V}$) and given a proof for $e\sigma \equiv f(e_1, \dots, e_n)\sigma \rightarrowtriangle t$, we apply the IH to every $e_i\sigma \rightarrowtriangle p_i\theta_{ij}$ to get that $e_i\sigma' \rightarrowtriangle p_i\theta_{ij}$ with a proof of the same size or smaller. But then we can use this proofs in a POR step from $e\sigma' \equiv f(e_1\sigma', \dots, e_n\sigma')$ and use the same substitution $\theta \in CSubst^?_\perp$ for parameter passing, constructing a proof with a size equal or smaller than the size of the starting one.
2. If $\theta \sqsubseteq \theta'$ then for any $X \in \mathcal{V}$ we have $\theta(X) \sqsubseteq \theta'(X)$, hence if $\theta(X) \rightarrowtriangle t$ then $\theta'(X) \rightarrowtriangle t$ by the mononicity of $\pi CRWL$. But then we can apply the strong monotonicity of $Subst_\perp$ to get the desired result.
3. Using the notations of Definition 3, given $X_i \in \overline{X} = var(e)$ if the set of positions of the occurrences of X_i in e is $\{p_{i1}, \dots, p_{im_i}\}$ then $e \equiv e[X_i]_{p_{i1}} \equiv (e[X_i]_{p_{i1}})[X_i]_{p_{i2}} \equiv \dots e[X_i]_{p_{i1}} \dots [X_i]_{p_{im_i}}$. As the positions of any pair of different occurrences of (possibly different) variables are parallel, we can do this for every variable in $\overline{X}$ to get $e \equiv e[Y_1]_{o_1} \dots [Y_m]_{o_m}$, where $\{o_1, \dots, o_m\}$ is the set of positions of every occurrence in e of any variable in $var(e)$ and $\{Y_1, \dots, Y_m\} = \overline{X}$. Note how each position in $\{o_1, \dots, o_m\}$ is parallel

to each other. But then we can apply the recasted version of Theorem 1 that appears in Definition 3, to get:

$$
\begin{array}{ll}
[\![e\sigma]\!]^{pl} = [\![e[Y_1\sigma]_{o_1} \dots [Y_m\sigma]_{o_m}]\!]^{pl} & \\
= \bigcup_{\overline{t_1} \subseteq [\![Y_1\sigma]\!]^{pl}} [\![e[?\overline{t_1}]_{o_1} \dots [Y_m\sigma]_{o_m}]\!]^{pl} & \text{by Theorem 1} \\
= \bigcup_{\overline{\overline{t} \subseteq [\![Y\sigma]\!]^{pl}}} [\![e[\overline{?\overline{t}}]_{\overline{o}}]\!]^{pl} & \text{by Theorem 1 (many times)} \\
= \bigcup_{\overline{\overline{t} \subseteq [\![Y\sigma']\!]^{pl}}} [\![e[\overline{?\overline{t}}]_{\overline{o}}]\!]^{pl} & \text{as } \sigma \trianglelefteq \sigma' \\
= [\![e[Y_1\sigma']_{o_1} \dots [Y_m\sigma']_{o_m}]\!]^{pl} = [\![e\sigma']\!]^{pl} & \text{by Theorem 1}
\end{array}
$$

Note that we cannot claim $e\sigma' \rightarrow t$ with proof of the same size of smaller, as we can see for example with $\sigma = [X/0] \trianglelefteq [X/0 \ ? \ 0] = \sigma', e \equiv X, t \equiv 0$ for which $e\sigma \equiv 0 \rightarrow 0$ with size one but $e\sigma' \equiv 0 \ ? \ 0 \rightarrow 0$ with size greater or equal to four.

4. If $\theta \sqsubseteq_\pi \theta'$ then $\theta \trianglelefteq \theta'$ by Lemma 6, hence this item holds by the previous item.

A.2 For Section 4

Lemma 7. *For any CRWL-program,* $e \in Exp_\perp, t \in CTerm_\perp, \theta \in CSubst_\perp$ *if* $\mathcal{P} \vdash_{\pi CRWL} e \rightarrow t$ *then* $\mathcal{P} \vdash_{\pi CRWL} e\theta \rightarrow t\theta$.

Proof. If $t \equiv \perp$ then $e\theta \rightarrow t\theta \equiv \perp$ by B, otherwise the proof is just a simple induction over the size of $e \rightarrow t$.

In order to prove the soundness of a rewriting step wrt. $\pi CRWL$ we will need the following auxiliary but revealing results.

Definition 4 (Denotation of substitutions). *Given* $\sigma \in Subst_\perp$ *we define* $[\![\sigma]\!]^{pl} = \{\theta \in CSubst_\perp \mid dom(\theta) = dom(\sigma) \wedge \forall X \in dom(\theta) \ \vdash_{\pi CRWL} \sigma(X) \rightarrow \theta(X)\}$.

Lemma 8. *For any* $\sigma \in Subst_\perp, e \in Exp_\perp, t \in CTerm_\perp$ *if* $\vdash_{\pi CRWL} e\sigma \rightarrow t$ *then* $\exists \Theta \subseteq [\![\sigma]\!]^{pl}$ *finite such that* $\vdash_{\pi CRWL} e(?\Theta) \rightarrow t$

Proof. By a case distintion over e:

- If $e \equiv X \in dom(\sigma)$: Then $e\sigma \equiv \sigma(X) \rightarrow t$, so we can define:

$$
\theta(Y) = \begin{cases} t & \text{if } Y \equiv X \\ \perp & \text{if } Y \not\equiv X \wedge Y \in dom(\sigma) \\ Y & \text{otherwise} \end{cases}
$$

  Then is trivial to check that $\theta \in [\![\sigma]\!]^{pl}$, so we can take $\Theta = \{\theta\}$ for which $e(?\Theta) \equiv \theta(X) \equiv t \rightarrow t$, by Lemma 3.
- If $e \equiv X \notin dom(\sigma)$: Then given $\overline{Y} = dom(\sigma)$ we define $[\overline{Y/\perp}]$ for which $[\overline{Y/\perp}] \in [\![\sigma]\!]^{pl}$ obviously, so we can take $\Theta = \{[\overline{Y/\perp}]\}$ for which $[\![e\sigma]\!]^{pl} = [\![X]\!]^{pl} = [\![X(?\Theta)]\!]^{pl}$.
- If $e \notin \mathcal{V}$ then we proceed by induction over the structure of $e\sigma \rightarrow t$:

  Base cases

  B Then $t \equiv \perp$, so given $\overline{Y} = dom(\sigma)$ we can take $\Theta = \{[\overline{Y/\perp}]\}$ for which $e(?\Theta) \rightarrow \perp$ by B.

  RR Then $e \in \mathcal{V}$ and we are in the previous case.

  DC Similar to the case for $e \equiv X \notin dom(\sigma)$.

  Inductive steps

  DC Then $e \equiv c(e_1, \dots, e_n)$, as $e \notin \mathcal{V}$, and we have:

$$
\frac{e_1\sigma \rightarrow t_1 \ \dots \ e_n\sigma \rightarrow t_n}{e\sigma \equiv c(e_1\sigma, \dots, e_n\sigma) \rightarrow c(t_1, \dots, t_n) \equiv t} \ DC
$$

  By IH or the proof of the other cases $\forall i \in \{1, \dots, n\} \ \exists \Theta_i \subseteq [\![\sigma]\!]^{pl}$ such that $e_i\Theta_i \rightarrow t_i$. Hence we can define $\Theta = \bigcup_{i \in \{1,\dots,n\}} \Theta_i$, for which is trivial to prove that $\forall i \in \{1, \dots, n\} \ ?\Theta_i \sqsubseteq_\pi ?\Theta$, and so $\forall i \in \{1, \dots, n\} \ e_i\Theta_i \rightarrow t_i$ implies $e_i\Theta \rightarrow t_i$, by Lemma 2. Therefore $e(?\Theta) \rightarrow c(t_1, \dots, t_n)$ by DC.

POR Very similar to the proof of the previuos case. We also have $e \equiv f(e_1, \ldots, e_n)$ (as $e \notin \mathcal{V}$) and given a proof for $e\sigma \equiv f(e_1, \ldots, e_n)\sigma \rightarrowtriangle t$, we apply the IH or the proof of the other cases to every $e_i\sigma \rightarrowtriangle p_i\theta_{ij}$ to get some $\Theta_{ij} \subseteq [\![e_i]\!]^{pl}$ such that $e_i(?\Theta_{ij}) \rightarrowtriangle p_i\theta_{ij}$. Then we define $\Theta = \bigcup_{i\in\{1,\ldots,n\}} \bigcup_{j\in\{1,\ldots,m_i\}} \Theta_{ij}$ for which $\forall i, j,\ ?\Theta_{ij} \sqsubseteq_\pi ?\Theta$ obviously holds, and as a consequence $\forall i, j,\ e_i(?\Theta) \rightarrowtriangle p_i\theta_{ij}$. Hence with $e(?\Theta) \equiv f(e_1(?\Theta), \ldots, e_n(?\Theta))$ we can compute the same value for its arguments and thus use the same substitution $\theta \in CSubst^?_\perp$ for parameter passing in POR.

Lemma 9. *For any finite $\Theta \subseteq [\![\sigma]\!]^{pl}$ we have $?\Theta \trianglelefteq \sigma$.*

Proof. Note that Θ is required to be finite because otherwise $?\Theta$ is not defined. Given $X \in \mathcal{V}$ if $X \notin dom(?\Theta) = dom(\sigma)$ (as $\Theta \subseteq [\![\sigma]\!]^{pl}$) then $[\![?\Theta(X)]\!]^{pl} = [\![X]\!]^{pl} = [\![\sigma(X)]\!]^{pl}$. Otherwise if $(?\Theta)(X) \rightarrowtriangle t$ then $\exists\theta_i \in \Theta$ such that $\theta_i(X) \rightarrowtriangle t$, hence $t \sqsubseteq \theta_i(X)$ by Lemma 3. But as $\Theta \subseteq [\![\sigma]\!]^{pl}$ then $\theta_i \in [\![\sigma]\!]^{pl}$, therefore $\sigma(X) \rightarrowtriangle \theta_i(X)$, and so $\sigma(X) \rightarrowtriangle t$ by the monotonicity Lemma 1, as $t \sqsubseteq \theta_i(X)$.

Lemma 10. *For any $t \in CTerm_\perp, \Theta \subseteq CSusbt_\perp$ finite, given $\theta_i \in \Theta$ then $\vdash_{\pi CRWL} t(?\Theta) \rightarrowtriangle t\theta_i$*

Proof. A simple induction on the structure of t.

Lemma 11 (One step soundness of $\to$ wrt $\pi CRWL$). *For any CRWL-program $\mathcal{P}$, $e, e' \in Exp$ if $e \to e'$ then $[\![e']\!]^{pl} \subseteq [\![e]\!]^{pl}$.*

Proof (For Lemma 11). Assume the step has been performed at the top of the expression, that is, $e \equiv f(p_1, \ldots, p_n)\sigma \to r\sigma \equiv e'$ for some $(f(p_1, \ldots, p_n) \to r) \in \mathcal{P}$. Given some $t \in CTerm_\perp$ such that $r\sigma \rightarrowtriangle t$ then by Lemma 8 there must exists some $\Theta \subseteq [\![\sigma]\!]^{pl}$ finite for which $r(?\Theta) \rightarrowtriangle t$. If $\Theta = \{\theta_1, \ldots, \theta_m\}$ then we can do:

$$\dfrac{\begin{array}{c}\dfrac{\text{Lemma 10}}{p_1(?\Theta) \rightarrowtriangle p_1\theta_1 \equiv p_1\theta_1|_{var(p_1)}} \\ \ldots \\ \dfrac{\text{Lemma 10}}{p_1(?\Theta) \rightarrowtriangle p_1\theta_m \equiv p_1\theta_m|_{var(p_1)}}\end{array} \quad \ldots \quad \begin{array}{c}\dfrac{\text{Lemma 10}}{p_n(?\Theta) \rightarrowtriangle p_n\theta_1 \equiv p_n\theta_1|_{var(p_n)}} \\ \ldots \\ \dfrac{\text{Lemma 10}}{p_n(?\Theta) \rightarrowtriangle p_n\theta_m \equiv p_n\theta_m|_{var(p_n)}}\end{array} \quad r\theta' \equiv_{(*)} r(?\Theta) \rightarrowtriangle t}{f(p_1, \ldots, p_n)(?\Theta) \rightarrowtriangle t}\ POR$$

with $\theta' = ?\{\theta_1|_{var(p_1)} \ldots \theta_m|_{var(p_1)}\} \uplus \ldots \uplus\ ?\{\theta_1|_{var(p_n)} \ldots \theta_m|_{var(p_n)}\}$, using the same program rule. The equivalence $(*)$ holds because for any $X \in var(r) \subseteq var(f(p_1, \ldots, p_n))$ there must exist some p_i such that $X \in var(p_i)$, hence

$$\begin{array}{l} X\theta' \equiv X(?\{\theta_1|_{var(p_i)} \ldots \theta_m|_{var(p_i)}\}) \\ \theta_1(X)\ ?\ \ldots\ ?\ \theta_m(X) \equiv (?\Theta)(X) \end{array}$$

Now we can apply Lemma 9 to get $?\Theta \trianglelefteq \sigma$, which combined with Lemma 2 implies $f(p_1, \ldots, p_n)\sigma \rightarrowtriangle t$.
If the step was not performed at the root of the expression then we have $e \equiv \mathcal{C}[f(\overline{p})\sigma] \to \mathcal{C}[r\sigma] \equiv e'$ for which $f(\overline{p})\sigma \to r\sigma$ is performed at the top. But then $[\![r\sigma]\!]^{pl} \subseteq [\![f(\overline{p})\sigma]\!]^{pl}$ by the proof of the previous case, and we can chain:

$$\begin{array}{llll} [\![\mathcal{C}[r\sigma]]\!]^{pl} = & \bigcup\limits_{\{t_1,\ldots,t_n\}\subseteq[\![r\sigma]\!]^{pl}} & & \text{by Theorem 1} \\ \subseteq & \bigcup\limits_{\{t_1,\ldots,t_n\}\subseteq[\![f(\overline{p})\sigma]\!]^{pl}} & = [\![\mathcal{C}[f(\overline{p})\sigma]]\!]^{pl} & \text{by Theorem 1} \end{array}$$

Now we have the tools to prove Theorem 5 and Corollary 1.

Proof (For Theorem 5). Given some $t \in [\![e]\!]^{rw}$, by definition $\exists e' \in Exp$ such that $t \sqsubseteq |e'|$ and $e \to^* e'$. We can extend Lemma 11 to $\to^*$ by a simple induction on the length of $e \to^* e'$, hence $[\![e']\!]^{pl} \subseteq [\![e]\!]^{pl}$. As $\forall e \in Exp_\perp, |e| \in [\![e]\!]^{pl}$ (by a simple induction on the structure of e), then $t \sqsubseteq |e'| \in [\![e']\!]^{pl} \subseteq [\![e]\!]^{pl}$, hence $t \in [\![e]\!]^{pl}$ by Lemma 1. Example 4 shows that the converse inclusion does not hold in general.

Proof (For Corollary 1). The first part holds just combining Theorem 4 with Theorem 5. Concerning the second part, assume $\vdash_{CRWL} e \rightarrowtriangle t$, in other words, $t \in [\![e]\!]^{sg}$. Then by the first part $t \in [\![e]\!]^{rw}$, hence $e \rightarrow^* e'$ such that $t \sqsubseteq |e'|$. But as $t \in CTerm$ then t is maximal wrt $\sqsubseteq$ (a know property of $\sqsubseteq$), and so $t \sqsubseteq |e'|$ implies $t \equiv |e'|$, which implies $t \equiv e'$, as t is total (very easy to check by induction on the structure of t). Therefore $e \rightarrow^* e' \equiv t$. Concerning the last fact holds by defition, if $e \rightarrow^* t \in CTerm$ then $t \in [\![e]\!]^{rw}$ by definition, as $t \sqsubseteq t \equiv |t|$ (an old property of shells easy to check by induction on the structure of t), but then $t \in [\![e]\!]^{pl}$ by the first part, in other words, $e \rightarrowtriangle t$.

A.3 For Section 5

The following auxiliary results will be needed to prove Theorem 6.

Lemma 12. *For any CRWL-program $\mathcal{P}$, πCRWL-statement $if\ e_1\ then\ e_2 \rightarrowtriangle t$ there is a πCRWL-proof for that statement of the shape:*

$$\frac{e_1 \rightarrowtriangle true \quad e_2 \rightarrowtriangle t \quad t \rightarrowtriangle t}{if\ e_1\ then\ e_2 \rightarrowtriangle t}\ POR$$

Proof. As the only rule for $if\_then\_$ is $if\ true\ then\ X \rightarrow X$ then every proof must be of the shape:

$$\frac{\begin{array}{lll} e_1 \rightarrowtriangle true\theta_1 & e_2 \rightarrowtriangle t_1 & \\ \ldots & \ldots & t_1 ? \ldots ? t_l \rightarrowtriangle t \\ e_1 \rightarrowtriangle true\theta_m & e_2 \rightarrowtriangle t_l & \end{array}}{if\ e_1\ then\ e_2 \rightarrowtriangle t}\ POR$$

As $t_1 ? \ldots ? t_l \rightarrowtriangle t$ there must exist some $t_i \in \{t_1, \ldots, t_l\}$ such that $t_i \rightarrowtriangle t$, but then $t \sqsubseteq t_i$ by Lemma 3 and so $e_2 \rightarrowtriangle t_i$ implies $e_2 \rightarrowtriangle t$ by Lemma 1, and we can apply Lemma 3 again to get $t \rightarrowtriangle t$ and construct the proof of the formulation.

Proof (For Theorem 6). Assume $pST(\mathcal{P}) \vdash_{\pi CRWL} e \rightarrowtriangle t$ for some $t \in CTerm_\perp$, we will see that then $\mathcal{P} \vdash_{\pi CRWL} e \rightarrowtriangle t$ by induction on the size of $pST(\mathcal{P}) \vdash_{\pi CRWL} e \rightarrowtriangle t$. The base cases are trivial because no program rule is involved, and so it is the case for DC, in which we only have to apply the IH over the hypothesis. The case for POR when $e \equiv f(\overline{e})$ and $f \in \{\_?\_, if\_then\_\}$ can be resolved applying the IH too, so the difficult case is that in which $f \in \{\_?\_, if\_then\_\}$. For the sake of sake of simplicity we will consider $f \in FS^1$, the proof can be easily extended to functions with zero or more than one arguments. Assume the rule used was $f(Y) \rightarrow if\ match(Y)\ then\ r[\overline{X_j/project_j(Y)}]$, corresponding to the original rule $f(p) \rightarrow r$ and with the auxiliary functions defined by $match(p) \rightarrow true$, $project_j(p) \rightarrow X_j$, for $\overline{X_j} = var(p) \cap var(r)$. Then the proof was of the shape:

$$\frac{\begin{array}{l} e \rightarrowtriangle t_1 \\ \ldots \\ e \rightarrowtriangle t_m \quad if\ match(t_1 ? \ldots ? t_m)\ then\ r[\overline{X_j/project_j(t_1 ? \ldots ? t_m)}] \rightarrowtriangle t \end{array}}{pST(\mathcal{P}) \vdash_{\pi CRWL} f(e) \rightarrowtriangle t}\ POR$$

where

$$\frac{\dfrac{t_1 ? \ldots ? t_m \rightarrowtriangle p\mu \quad true \rightarrowtriangle true}{match(t_1 ? \ldots ? t_m) \rightarrowtriangle true}\ POR \quad r[\overline{X_j/project_j(t_1 ? \ldots ? t_m)}] \rightarrowtriangle t \quad t \rightarrowtriangle t}{pST(\mathcal{P}) \vdash_{\pi CRWL} if\ match(t_1 ? \ldots ? t_m)\ then\ r[\overline{X_j/project_j(t_1 ? \ldots ? t_m)}] \rightarrowtriangle t}\ POR$$

by Lemma 12. Let $s_1 \ldots s_l = t_1 \ldots t_m \mid \lambda t.(t \equiv p\theta)$ with $dom(\theta) = var(p)$, for some $\theta \in CSubst_\perp$, one θ for each s in $s_1 \ldots s_n$. Then $pST(\mathcal{P}) \vdash_{\pi CRWL} t_1 ? \ldots ? t_m \rightarrowtriangle p\mu$ implies $\exists j \in \{1, \ldots, m\}$ such that $pST(\mathcal{P}) \vdash_{\pi CRWL} t_j \rightarrowtriangle p\mu$, and so $p\mu \sqsubseteq t_j$ by Lemma 3. But then it is very easy to prove by induction on the structure of t, taking advantage of its linearity and totality, that there must exist some $\theta_j \in CSubst_\perp$ such that $t_j \equiv p\theta_j$. Hence $s_1 \ldots s_l$ is not empty and then it is very easy to prove that

$[\overline{X_j/project_j(t_1? \ldots ?t_m)}]$ and $[\overline{X_j/project_j(s_1? \ldots ?s_l)}]$ verify the conditions to apply the strong monotonicity of Lemma 2 in order to get that $pST(\mathcal{P}) \vdash_{\pi CRWL} r[\overline{X_j/project_j(s_1? \ldots ?s_l)}] \rightarrowtriangle t$ with a proof of the same size or smaller. By definition $s_1 \ldots s_l \equiv p\theta_1 \ldots p\theta_l$, now we will see that the substitutions $[\overline{X_j/project_j(s_1? \ldots ?s_l)}]$ and $?\{\theta_1, \ldots, \theta_l\}|_{var(r)}$ also verify the conditions of to apply the strong monotonicity of Lemma 2. As $\forall \theta \in \{\theta_1, \ldots, \theta_l\}\ dom(\theta) = var(p)$ then $dom(?\{\theta_1, \ldots, \theta_l\}|_{var(r)}) = \overline{X_j}$, so given $X \notin \overline{X_j}$ both substitutions leave it untouched. On the other hand if $X \equiv X_j \in \overline{X_j}$, given

$$\dfrac{\begin{array}{l} s_1? \ldots ?s_l \rightarrowtriangle p\mu_1 \\ \ldots \\ s_1? \ldots ?s_l \rightarrowtriangle p\mu_h \quad X_j(?\{\mu_1, \ldots, \mu_h\}) \rightarrowtriangle t \end{array}}{pST(\mathcal{P}) \vdash_{\pi CRWL} X_j[\overline{X_j/project_j(s_1? \ldots ?s_l)}] \equiv project_j(s_1? \ldots ?s_l) \rightarrowtriangle t}\ POR$$

Then $pST(\mathcal{P}) \vdash_{\pi CRWL} X_j(?\{\mu_1, \ldots, \mu_h\}) \rightarrowtriangle t$ implies $pST(\mathcal{P}) \vdash_{\pi CRWL} \mu(X_j) \rightarrowtriangle t$ for some $\mu \in \{\mu_1, \ldots, \mu_h\}$. But then $pST(\mathcal{P}) \vdash_{\pi CRWL} s_1? \ldots ?s_l \rightarrowtriangle p\mu$, and so $pST(\mathcal{P}) \vdash_{\pi CRWL} s \rightarrowtriangle p\mu$ for some $s \in \{s_1, \ldots, s_l\}$. Hence, by Lemma 3 $t \sqsubseteq \mu(X_j)$ and $p\mu \sqsubseteq s$, and as $s_1 \ldots s_l \equiv p\theta_1 \ldots p\theta_l$ by definition then $p\mu \sqsubseteq s \equiv p\theta$ for some $\theta \in \{\theta_1, \ldots, \theta_l\}$. But as $X_j \in \overline{X_j} \subseteq var(p)$ then $p\mu \sqsubseteq p\theta$ implies $\mu(X_j) \sqsubseteq \theta(X_j)$, hence $t \sqsubseteq \mu(X_j) \sqsubseteq \theta(X_j)$ and $pST(\mathcal{P}) \vdash_{\pi CRWL} \theta(X_j) \rightarrowtriangle t$ with a proof of the same size or smaller, by Lemma 1, and so $pST(\mathcal{P}) \vdash_{\pi CRWL} X_j(?\{\theta_1, \ldots, \theta_l\}|_{var(r)}) \rightarrowtriangle t$ with a proof of the same size or smaller than the proof for $pST(\mathcal{P}) \vdash_{\pi CRWL} X_j[\overline{X_j/project_j(s_1? \ldots ?s_l)}] \rightarrowtriangle t$. But then we can apply Lemma 2 to get $pST(\mathcal{P}) \vdash_{\pi CRWL} r(?\{\theta_1, \ldots, \theta_l\}|_{var(r)}) \rightarrowtriangle t$ to which we can apply the IH to get $\mathcal{P} \vdash_{\pi CRWL} r(?\{\theta_1, \ldots, \theta_l\}|_{var(r)}) \rightarrowtriangle t$. We can also apply the IH to each $pST(\mathcal{P}) \vdash_{\pi CRWL} e_i \rightarrowtriangle t_l \equiv s_l \equiv p\theta$ and build the following proof:

$$\dfrac{\begin{array}{l} e \rightarrowtriangle s_1 \equiv p\theta_1 \\ \ldots \\ e \rightarrowtriangle s_l \equiv p\theta_l \quad r(?\{\theta_1, \ldots, \theta_l\}) \equiv r(?\{\theta_1, \ldots, \theta_l\}|_{var(r)}) \rightarrowtriangle t \end{array}}{\mathcal{P} \vdash_{\pi CRWL} f(e) \rightarrowtriangle t}\ POR$$

Concerning the proof for Theorem 7, we will use the following auxiliary results.

Lemma 13. *For every $e \in Exp$ and $p \in CTerm$ linear, given $\theta \in CSubst_\perp$ such that $dom(\theta) \subseteq FV(p)$, if $p\theta \sqsubseteq |e|$ then $\exists \sigma \in Subst$ such that $dom(\sigma) = dom(\theta)$, $p\sigma \equiv e$ and $\theta \sqsubseteq \sigma$.*

Proof. See [16].

Definition 5. *Given a signature $\Sigma = FS \uplus CS$ and a CRWL-program $\mathcal{P}$ the set $FS^{\mathcal{P}} \subseteq FS$ is defined as $FS^{\mathcal{P}} = \{f \in FS \mid \exists (f(\overline{p}) \rightarrow r) \in \mathcal{P}\}$*

Definition 6. *Given a πCRWL-proof Δ for a πCRWL-statement $e \rightarrowtriangle t$ by Π^Δ we denote the multiset of πCRWL-statements that compose Δ, including $e \rightarrowtriangle t$. Sometimes we will use $\Pi^{e \rightarrowtriangle t}$ when Δ is implicit. We will also use Δ_π to refer to the subproof for some premise π of Δ, when it is implicit.*

Lemma 14. *For any πCRWL-statement $e \rightarrowtriangle t$ which holds exists some πCRWL-proof Δ such that $\forall e' \rightarrowtriangle t' \in \Pi^\Delta$, $e' \rightarrowtriangle t'$ is not a premise, neither directly nor indirectly, of itself in Δ.*

Proof. As $e \rightarrowtriangle t$ holds we may assume some π*CRWL*-proof Δ for it. If no $e' \rightarrowtriangle t' \in \Pi^\Delta$ is premise of itself then we are done. Otherwise as any π*CRWL*-proof is finite then taking the subproof corresponding to some $e' \rightarrowtriangle t'$ which is premise of itself there must be some $e' \rightarrowtriangle t'$ which does not have $e' \rightarrowtriangle t'$ as its premise in its corresponding proof. Then we can use that proof to replace the subproof for $e' \rightarrowtriangle t'$, as the proof is finite this process ends, because each time the number of sentences premises of itself decreases.

Lemma 15. *Given a CRWL-program $\mathcal{P}$ let $\hat{\mathcal{P}} \uplus \mathcal{M} = pST(\mathcal{P})$, where $\mathcal{M}$ is the set containing the rules for $\_?\_$, $if\_then\_$ and the new functions match and project, and $\hat{\mathcal{P}}$ contains the new versions of the original rules of $\mathcal{P}$. Then for any $e \in Exp_\perp, t \in CTerm_\perp$ constructed using just symbols in the signature of $\mathcal{P} \uplus \mathcal{M}$ we have $\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \rightarrowtriangle t$ implies $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e \rightarrow^* e'$ such that $t \sqsubseteq |e'|$.*

Proof. For any proof for $\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \twoheadrightarrow t$ we define $sa\mathcal{P}(e \twoheadrightarrow t)$ by

$$sa\mathcal{P}(e \twoheadrightarrow t) = \{e' \twoheadrightarrow t' \mid (e' \twoheadrightarrow t') \in \Pi^{e \twoheadrightarrow t} \wedge e' \equiv f(\overline{a}) \text{ for some } f \in FS^{\mathcal{P}}\}$$

Note that $sa\mathcal{P}(e \twoheadrightarrow t)$ is a set, not a multiset. Besides for any $\pi \in \Pi^{e \twoheadrightarrow t}$ we have $sa\mathcal{P}(\pi) \subseteq sa\mathcal{P}(e \twoheadrightarrow t)$ by definition. For any $\pi CRWL$-proof Δ by $size(\Delta)$ we denote the number of rules of the calculus used.
We assume we start with a proof for $\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \twoheadrightarrow t$ which fulfils the conditions granted by Lemma 14. We define the relation $\lessdot$ over pairs of $\pi CRWL$-proofs Δ_1, Δ_2, by $\Delta_1 \lessdot \Delta_2$ iff $sa\mathcal{P}(\Delta_1) \subset sa\mathcal{P}(\Delta_2)$ or $sa\mathcal{P}(\Delta_1) = sa\mathcal{P}(\Delta_2)$ and $size(\Delta_1) < size(\Delta_2)$. Then for any $\pi \in \Pi^{e \twoheadrightarrow t}$ if $\pi \not\equiv e \twoheadrightarrow t$ then $\Pi^{\pi} \lessdot \Pi^{e \twoheadrightarrow t}$. We proceed by induction over $\lessdot$ applied over $\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \twoheadrightarrow t$, let us do a case distinction over the rule applied at the root of the proof:

B Then $t \equiv \perp$ and $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e \rightarrow^0 e$ for which $\perp \sqsubseteq |e|$ holds.
RR Then $e \equiv X \equiv t$ and $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e \rightarrow^0 e$ for which $X \sqsubseteq X \equiv |e|$ holds.
DC If $e \equiv c \in CS^0$ then $t \equiv c$ and so $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e \rightarrow^0 e$ for which $c \sqsubseteq c \equiv |e|$ holds. Otherwise $\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \equiv c(e_1, \ldots, e_n) \twoheadrightarrow c(t_1, \ldots, t_n) \equiv t$. As we saw before for any $\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e_i \twoheadrightarrow t_i$ we have $\Delta_{e_i \twoheadrightarrow t_i} \lessdot \Delta_{e \twoheadrightarrow t}$, hence by IH $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e_i \rightarrow^* e'_i$ such that $t_i \sqsubseteq |e'_i|$. But then $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e \equiv c(e_1, \ldots, e_n) \twoheadrightarrow c(e'_1, \ldots, e'_n)$ and $c(t_1, \ldots, t_n) \sqsubseteq c(e'_1, \ldots, e'_n) \equiv |c(e'_1, \ldots, e'_n)|$.
POR Then we have $f(\overline{e}) \twoheadrightarrow t$. In case $t \equiv \perp$ lemma holds trivially for $f(\overline{e}) \rightarrow^0 f(\overline{e})$ with $t \equiv \perp \sqsubseteq \perp \equiv |f(\overline{e})|$. Otherwise we proceed by a case distinct over f.
If $f \in FS^{\mathcal{P}}$ first we will see that

$$r(?\{\theta_{11}, \ldots, \theta_{1m_1}\} \uplus \ldots \uplus ?\{\theta_{n1}, \ldots, \theta_{nm_n}\}) \equiv r[\overline{X_{ij}/\theta_{i1}(X_{ij}) \ ? \ \ldots \ ? \ \theta_{im_i}(X_{ij})}]$$

with $i \in \{1, \ldots, n\}, j \in \{1, \ldots, k_i\}$ (remember that in the transformation of Definition 1 we had $\forall p_i \in \{p_1, \ldots, p_n\}$, $var(p_i) \cap var(r) = \{X_{i1}, \ldots, X_{ik_i}\}$). This is true because for any $Y \in var(r)$ as $var(r) \subseteq var(\overline{p})$ there must exists some $p_i \in \overline{p}$ such that $Y \in var(p_i)$, hence:

$$\begin{array}{ll}
Y(?\{\theta_{11}, \ldots, \theta_{1m_1}\} \uplus \ldots \uplus ?\{\theta_{n1}, \ldots, \theta_{nm_n}\}) & \\
\equiv Y(?\{\theta_{i1}, \ldots, \theta_{im_i}\}\}) & (1) \\
\equiv \theta_{i1}(Y) \ ? \ \ldots \ ? \ \theta_{im_i}(Y) & (2) \\
\equiv Y[Y/\theta_{i1}(Y) \ ? \ \ldots \ ? \ \theta_{im_i}(Y)] & \\
\equiv Y[\overline{X_{ij}/\theta_{i1}(X_{ij}) \ ? \ \ldots \ ? \ \theta_{im_i}(X_{ij})}] & (3)
\end{array}$$

(1) because $\forall i, j \ dom(\theta_{ij}) = var(p_i)$ and $\overline{p}$ is linear; (2) because $Y \in var(p_i) = dom(\theta_{ij})$ for any $\theta_{ij} \in \{\theta_{i1}, \ldots, \theta_{im_i}\}$; because $Y \in var(r)$ and $Y \in var(p_i)$, therefore $Y \in \{X_{i1}, \ldots, X_{ik_i}\}$ by definition.

Now let us do a preliminary version for $f \in FS^1$. Assume the rule used was $(f(p) \rightarrow r) \in \mathcal{P}$, then its transformation was $f(Y) \rightarrow if \ match(Y) \ then \ r[\overline{X_i/project_i(Y)}]$, where $\overline{X_i} = var(p) \cap var(r)$ and the rules for $match$ and each $project_i$ are $match(p) \rightarrow true$, $project_i(p) \rightarrow X_i$, and the proof must be of the following shape:

$$\dfrac{\begin{array}{l} e \twoheadrightarrow p\theta_1 \\ \quad \ldots \\ e \twoheadrightarrow p\theta_m \quad r?\{\theta_1, \ldots, \theta_m\} \equiv r[\overline{X_i/\theta_1(X_i) \ ? \ \ldots \ ? \ \theta_m(X_i)}] \twoheadrightarrow t \end{array}}{\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} f(e) \twoheadrightarrow t} \ POR$$

Then for any $\theta_j \in \{\theta_1, \ldots, \theta_m\}$ we have $\Delta_{e \twoheadrightarrow p\theta_j} \lessdot \Delta_{f(e) \twoheadrightarrow t}$, so we choose arbitrary one of these θ_j and apply the IH to $\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \twoheadrightarrow p\theta_j$ to get $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e \rightarrow^* e'_j$ for some $e'_j \in Exp$ such that $p\theta_j \sqsubseteq |e'_j|$. But p is linear because it is in the left hand side of a rule, hence by Lemma 13 there must exist some $\sigma_j \in Subst$ such that $p\sigma_j \equiv e'_j$, and we can do:

$$\begin{array}{l}
\hat{\mathcal{P}} \uplus \mathcal{M} \vdash f(e) \rightarrow if \ match(e) \ then \ r[\overline{X_i/project_i(e)}] \\
\rightarrow^* if \ match(e'_j) \ then \ r[\overline{X_i/project_i(e)}] \\
\equiv if \ match(p\sigma_j) \ then \ r[\overline{X_i/project_i(e)}] \\
\rightarrow if \ true \ then \ r[\overline{X_i/project_i(e)}] \rightarrow r[\overline{X_i/project_i(e)}]
\end{array}$$

By Lemma 14 we know that $f(e) \to t$ is not a premise of itself, but then $saP(f(e) \to t) = (f(e) \to t) \uplus \mathcal{S}$, where $\mathcal{S} = \{\bigcup_{j \in \{1,\ldots,m\}} saP(e \to p\theta_j)\} \cup saP(r[\overline{X_i/\theta_1(X_i)\ ?\ \ldots\ ?\ \theta_m(X_i)}] \to t)$. Now we will see that there is a proof Δ for $\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} r[\overline{X_i/project_i(e)}] \to t$ such that for any $\pi \in \Pi^{\Delta}$, $saP(\pi) \subseteq \mathcal{S}$. If $t \equiv \perp$ the proof is trivial using no program rule, thus $saP(\pi) = \emptyset$. Otherwise we proceed by induction on the structure of r. If $r \equiv Y \in \mathcal{V}$ such that $Y \notin \overline{X_i}$ then the proof is trivial because then $r[\overline{X_i/\theta_1(X_i)\ ?\ \ldots\ ?\ \theta_m(X_i)}] \equiv Y \equiv r[\overline{X_i/project_i(e)}]$, and all the proofs starting from Y use no program rule and thus $saP(\pi) = \emptyset$. If $r \equiv X_i \in \overline{X_i}$ then as $r[\overline{X_i/\theta_1(X_i)\ ?\ \ldots\ ?\ \theta_m(X_i)}] \to t$ then there must exist some $\theta_j \in \{\theta_1, \ldots, \theta_m\}$ such that $\theta_j(X_i) \to t$. But then we can do:

$$\frac{e \to p\theta_j \quad X_i\theta_j \to t}{\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} r[\overline{X_i/project_i(e)}] \equiv project_i(e) \to t}\ POR$$

where $saP(X_i\theta_j \to s) = \emptyset$, as no program rule was used because $X_i\theta \in CTerm_{\perp}$, and $saP(e \to p\theta_j) \subseteq \mathcal{S}$: but then $saP(r[\overline{X_i/project_i(e)}] \to t) = saP(X_i\theta_j \to s) \cup saP(e \to p\theta_j) \subseteq \mathcal{S}$. If $r \in CS^0$ the proof is trivial as no program rule is involved. If $r \equiv c(a_1, \ldots, a_l)$ then $t \equiv c(t_1, \ldots, t_l)$ (as $t \not\equiv \perp$) and we can apply the HI over each $a_k[\overline{X_i/\theta_1(X_i)\ ?\ \ldots\ ?\ \theta_m(X_i)}] \to t_k$ to get $a_k[\overline{X_i/project_i(e)}] \to t_k$ with $saP(a_k[\overline{X_i/project_i(e)}] \to t_k) \subseteq \mathcal{S}$, hence $r[\overline{X_i/project_i(e)}]$ by DC with

$$saP(r[\overline{X_i/project_i(e)}]) = \bigcup_k saP(a_k[\overline{X_i/project_i(e)}] \to t_k) \subseteq \mathcal{S}$$

If $r \equiv g(a_1, \ldots, a_l)$ with $g \in FS$, we can apply the IH to every $a_k[\overline{X_i/\theta_1(X_i)\ ?\ \ldots\ ?\ \theta_m(X_i)}] \to s$ to get $a_k[\overline{X_i/project_i(e)}] \to s$ with $saP(a_k[\overline{X_i/project_i(e)}] \to s) \subseteq \mathcal{S}$. If the instance of the right hand side used in $g(a_1, \ldots, a_l)[\overline{X_i/\theta_1(X_i)\ ?\ \ldots\ ?\ \theta_m(X_i)}] \to t$ was $r'\mu \to t$, then we can use the same instance for $g(a_1, \ldots, a_l)[\overline{X_i/project_i(e)}] \to t$, as we have reduced the arguments to the same values. Besides by definition $saP(r'\mu \to t) \subseteq saP(g(a_1, \ldots, a_l)[\overline{X_i/\theta_1(X_i)\ ?\ \ldots\ ?\ \theta_m(X_i)}] \to t) \subseteq \mathcal{S}$, hence $saP(g(a_1, \ldots, a_l)[\overline{X_i/project_i(e)}] \to t) \subseteq \mathcal{S}$.
But then as $saP(f(e) \to t) = (f(e) \to t) \uplus \mathcal{S}$ then $saP(r[\overline{X_i/project_i(e)}] \to t) \subseteq \mathcal{S} \subset saP(f(e) \to t)$ and we can apply the HI to get $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash r[\overline{X_i/project_i(e)}] \to^* e'$ such that $t \sqsubseteq |e'|$.

If $e \equiv match(e_1, \ldots, e_n)$ for some of these auxiliary functions, with rule $match(p_1, \ldots, p_n) \to true$, then as $t \not\equiv \perp$ we have $t \equiv true$ with:

$$\frac{e_1 \to p_1\theta_1 \quad \ldots\ e_n \to p_n\theta_n \quad true \to true}{\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \equiv match(e_1, \ldots, e_n) \to true}\ POR$$

There could be more evaluations for each e_i but those are useless as $true$ is ground. Then we have $\Delta_{e_i \to p_i\theta_i} \lessdot \Delta_{e \to true}$ as we saw before, hence by IH $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e_i \to^* e'_i$ such that $p_i\theta_i \sqsubseteq |e'_i|$. But then by Lemma 13 there must exist some $\sigma_i \in Subst$ such that $p_i\sigma_i \equiv e'_i$, and we can do $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash match(e_1, \ldots, e_n) \to^* match(e'_1, \ldots, e'_n) \equiv match(p_1\sigma_1, \ldots, p_n\sigma_n) \to true$, and $true \sqsubseteq true \equiv |true|$.

If $e \equiv project(e_1)$ for some of these auxiliary functions, with rule $project(p) \to X$, then as $t \not\equiv \perp$ we have:

$$\frac{\begin{array}{l} e_1 \to p\theta_1 \\ \ldots \\ e_1 \to p\theta_m \quad X(?\{\theta_1, \ldots, \theta_m\}) \to t \end{array}}{\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \equiv project(e_1) \to t}\ POR$$

Then there must exist some $\theta_j \in \{\theta_1, \ldots, \theta_m\}$ such that $X\theta_j \to t$, hence $t \sqsubseteq \theta_j(X)$. Besides we have $\Delta_{e_1 \to p\theta_j} \lessdot \Delta_{e \to t}$ as we saw before, hence by IH $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e_1 \to^* e'_j$ such that $p\theta_j \sqsubseteq |e'_j|$, to which we can apply Lemma 13 to get some $\sigma_j \in Subst$ such that $p\sigma_j \equiv e'_j$ and $\theta_j \sqsubseteq \sigma_j$. Therefore $t \sqsubseteq \theta_j(X) \sqsubseteq \sigma_j(X)$, so it is trivial to check that then $t \sqsubseteq |\sigma_j(X)|$ (by induction on the structure of t), and we can do

$\hat{\mathcal{P}} \uplus \mathcal{M} \vdash match(e_1) \to^* match(e'_j) \equiv match(p\sigma_j) \to \sigma_j(X)$.

If $e \equiv if\ e_1\ then\ e_2$ then as $t \not\equiv \perp$ we have:

$$\frac{e_1 \rightarrowtriangle true \quad \begin{array}{c} e_2 \rightarrowtriangle t_1 \\ \dots \\ e_2 \rightarrowtriangle t_m \end{array} \quad t_1 ? \dots ? t_m \rightarrowtriangle t}{\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \equiv if\ e_1\ then\ e_2 \rightarrowtriangle t}\ POR$$

There could be more evaluations for e_1 but those are useless as $true$ is ground. We have $\Delta_{e_1 \rightarrowtriangle true} \lessdot \Delta_{e \rightarrowtriangle t}$ as we saw before, hence by IH $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e_1 \to^* e'_1$ such that $true \sqsubseteq |e'_1|$. Then we can apply Lemma 13 to get some $\sigma_1 \in Subst$ such that $true \equiv true\sigma_1 \equiv e'_1$, so we can do $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash if\ e_1\ then\ e_2 \to^* if\ e'_1\ then\ e_2 \equiv if\ true\ then\ e_2 \to e_2$. Besides $t_1 ? \dots ? t_m \rightarrowtriangle t$ implies $t_j \rightarrowtriangle t$ for some $t_j \in \{t_1, \dots, t_m\}$ such that $t \sqsubseteq t_j$ and $\Delta_{e_2 \rightarrowtriangle t_j} \lessdot \Delta_{e \rightarrowtriangle t}$ as we saw before. We can apply the IH to get $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e_2 \to^* e'$ such that $t \sqsubseteq t_j \sqsubseteq |e'|$.

If $e \equiv e_1\ ?\ e_2$ then as $t \not\equiv \perp$ we have $e_i \rightarrowtriangle t$ for some $i \in \{1, 2\}$ with a proof to which we can apply the IH to get $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e_1\ ?\ e_2 \to e_i \to^* e'$ such that $t \sqsubseteq |e'|$.

Finally we are ready to prove Theorem 7 and Corollary 2.

Proof (For Theorem 7). Let $\hat{\mathcal{P}} \uplus \mathcal{M} = pST(\mathcal{P})$ be , where $\mathcal{M}$ is the set containing the rules for $\_?\_$, $if\_then\_$ and the new functions $match$ and $project$, and $\hat{\mathcal{P}}$ contains the new versions of the original rules of $\mathcal{P}$. If $e \in Exp, t \in CTerm_\perp$ are built using symbols on the signature of $\mathcal{P}$, then $\mathcal{P} \vdash_{\pi CRWL} e \rightarrowtriangle t$ implies $\mathcal{P} \uplus \mathcal{M} \vdash_{\pi CRWL} e \rightarrowtriangle t$, which implies $\hat{\mathcal{P}} \uplus \mathcal{M} \vdash e \to^* e'$ such that $t \sqsubseteq |e'|$ by Lemma 15, that is, $pST(\mathcal{P}) \vdash e \to^* e'$.

Proof (For Corollary 2). Given some $t \in [\![e]\!]^{pl}_{\mathcal{P}}$ then by Theorem 7 exists some $e' \in Exp$ such that $pST(\mathcal{P}) \vdash e \to^* e'$ and $t \sqsubseteq |e'|$, hence $t \in [\![e]\!]^{rw}_{pST(\mathcal{P})}$ by definition. On the other hand if $t \in [\![e]\!]^{rw}_{pST(\mathcal{P})}$ then $t \in [\![e]\!]^{pl}_{pST(\mathcal{P})}$ by Corollary 1, but then $t \in [\![e]\!]^{pl}_{\mathcal{P}}$ by Theorem 6. For the second part, if $\mathcal{P} \vdash_{\pi CRWL} e \rightarrowtriangle t$ then $t \in [\![e]\!]^{pl}_{\mathcal{P}} = [\![e]\!]^{rw}_{pST(\mathcal{P})}$ by the first part, hence $\exists e' \in Exp$ such that $e \to^* e'$ and $t \sqsubseteq |e'|$. But as $t \in CTerm$ t is maximal wrt. $\sqsubseteq$ and so $t \equiv |e'|$ which implies $t \equiv e'$ (these are known properties of shells and $\sqsubseteq$). But then $pST(\mathcal{P}) \vdash e \to^* e' \equiv t$. If $pST(\mathcal{P}) \vdash e \to^* t$ then as $t \sqsubseteq t \equiv |t|$ then $t \in [\![e]\!]^{rw}_{pST(\mathcal{P})} = [\![e]\!]^{pl}_{\mathcal{P}}$: $\mathcal{P} \vdash_{\pi CRWL} e \rightarrowtriangle t$.