A Fully Abstract Semantics for Constructor Systems (Extended version) *

F.J. López-Fraguas, J. Rodríguez-Hortalá, and J. Sánchez-Hernández

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain
fraguas@sip.ucm.es, jrodrigu@fdi.ucm.es, jaime@sip.ucm.es

Abstract. Constructor-based term rewriting systems are a useful subclass of TRS, in particular for programming purposes. In this kind of systems constructors determine a universe of values, which are the expected output of the computations. Then it would be natural to think of a semantics associating each expression to the set of its reachable values. Somehow surprisingly, the resulting semantics has poor properties, for it is not compositional nor fully abstract when non-clonfluent systems are considered. In this paper we propose a novel semantics for expressions in constructor systems, which is compositional and fully abstract (with respect to sensible observation functions, in particular the set of reachable values for an expression), and therefore can serve as appropriate basis for semantic based analysis or manipulation of such kind of rewrite systems.

1 Introduction

Constructor based term rewriting systems (CS) are an important subclass of TRS. The use of CS for programming has been frequently connected to the requirement of confluence. By these days this is not necessarily so, and many proposals (see e.g.[12, 3, 9, 8, 10]) drop the requirement of confluence and/or termination.

On the other hand, it is widely accepted that an adequate semantics constitutes an excellent companion to any programming language. In the case of CS, an 'obvious' notion of semantics comes from defining the denotation of an expression e as the set of values reachable from e by rewriting. The notion of 'values' could be made concrete in different manners: constructor terms, outer constructor part of expressions or normal forms. Two questions arise:
– Is the semantics compositional? In our case: is the semantics of an expression determined by the semantics of its subexpressions?
– Does it capture observational equivalence? That is: for two semantically equivalent expressions e, e', is it ensured that we will *observe* the same behavior when e, e' are put in the same context? This depends on a criterion of what can be observed from an expression. In the constructor discipline point of view, one is mostly interested again in observing which constructor terms (or outer stable constructor part) can be reached by rewriting.

Somehow surprisingly, the answers is negative for the 'obvious' semantics:

Example 1. Consider the constructors a, b, c, d and the non-confluent program

$$f(c(X)) \to d(X, X) \quad amb(X, Y) \to X \quad amb(X, Y) \to Y$$

The expressions $e \equiv c(amb(a, b))$ and $e' \equiv amb(c(a), c(b))$ reach by rewriting exactly the same constructor values, namely $c(a)$ and $c(b)$. However, this does not ensure that e, e' behave the same when put in the same context. For instance, $f(e)$ can be rewritten to the constructor values $d(a, a), d(a, b), d(b, a), d(b, b)$ while $f(e')$ only to $d(a, a)$ and $d(b, b)$. More in general, this works starts by remarking that *knowing the constructor values of an expression e is not enough information to know the constructor values of $\mathcal{C}[e]$ for any given context $\mathcal{C}$*. The same example shows that the remark remains true if we replace 'constructor value' by 'normal form' or 'outer constructor part'. Using standard terminology (see Sect. 4 for definitions) all those semantics are not compositional, sound nor fully abstract.

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

The aim of our work can be made clear now: to define a semantics for CS that is fully abstract (compositionality and soundness will come along the way) wrt the observability criterion of reachable constructor terms.

Our starting insight is that, to recover compositionality, the semantics must not collect a *flat* set of reachable values, like is $\{c(a), c(b)\}$ for $c(amb(a,b))$, but rather a more structured and 'packaged' representation, where constructors can be applied to sets, as to reflect more appropriately the matching capabilities of expressions. In our example, and disregarding for the moment some technical details, the denotation of $c(amb(a,b))$ will be the singleton 'package' $\{c(\{a,b\})\}$, reflecting the fact that $c(amb(a,b))$ can match $c(X)$ without reducing $amb(a,b)$, while the denotation of $amb(c(a), c(b))$ will be the two-element package $\{c(a), c(b)\}$. Technically, things will be a bit more complicated (see Sect. 3), in particular due to the possibility of non-termination, that yields possibly infinite sets, and will require expressing some kind of *partial* values in the semantics.

Related work Not too much attention has been paid to the issue of semantics of TRS, at least when compared to the huge amount of research in the fields of TRS and of semantics of programming languages in general. There are nevertheless some works to be mentioned.

In [6], Boudol develops a deep theory of the space of computations of left-linear TRS and provides a computational semantics based on continuous algebras. However, his semantics still associates an expression with a flat set of (possibly infinite) values, thus presenting the problems of our Ex. 1. Moreover, [5, 15] demonstrate that there are problems with achieving full abstraction for non-terminating non-deterministic systems, if the semantics is based on fixpoints and infinite (limit) values (our semantics will avoid them). In [2] a compositional semantics for conditional TRS is presented. Compositionality is understood in a different sense, related to the issue of joining programs. In addition, the considered programs are canonical (confluent and terminating). In [1], an abstract diagnosis scheme for functional programs modeled as TRS is developed, based on some notions of semantics that again collect results of individual computations. The semantics characterization of narrowing given in [11] includes a semantics for TRS, but most of the interesting results are for confluent ones. On the other hand, the cited papers give a more general treatment of variables, which have a passive role in our paper, behaving almost as constants.

With respect to the nesting of sets inside constructor symbols, a similar idea appears in [4], to improve the efficiency of functional logic computations, in [7] as part of the design of a functional programming implementation of functional logic languages, and in [13] as a mean for improving the programming of non-determinism in a Haskell-like ambient. All these works are much more oriented to practice, far from the aims and results of our present work. Moreover, the setting is not the same: functional logic programming for the two first (with a *call-time choice* semantics [9], having essential differences with standard rewriting) and functional programming for the last one.

The rest of the paper is organized as follows. Sect. 2 contains some preliminaries about TRS. Sect. 3 is the technical core of the paper, where our semantics is technically defined and many strong properties are proved: polarity, compositionality, adequacy wrt rewriting. In Sect. 4 we discuss in detail the question of full abstraction. Finally Sect. 5 presents some conclusions. Detailed proofs can be found in Appendix A.

2 Preliminaries

We assume a first order signature $\Sigma = DC \cup FS$, where DC and FS are two disjoint sets of *constructor* and *function* symbols resp., all them with associated arity. We write DC^n (FS^n resp.) for the set of constructor (function) symbols of arity n, and also Σ^n for any symbol of the signature of arity n. We also assume a numerable set $\mathcal{V}$ of variables. As usual notations we write $c, d, \ldots$ for constructors, $f, g, \ldots$ for functions and $X, Y, \ldots$ for variables. The set Exp of *expressions* is defined as $Exp \ni e ::= X \mid h(e_1, \ldots, e_n)$, where $X \in \mathcal{V}$, $h \in \Sigma^n$ 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 DC^n (so $CTerm \subseteq Exp$).[1] 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 $var(e)$. The notation $\overline{o}$ stands for tuples of any kind of syntactic objects along the paper.

We consider also the extended signature $\Sigma_\perp = \Sigma \cup \{\perp\}$, where $\perp$ is a new 0-arity constructor symbol that stands for the undefined value. Over this signature we define the sets $Exp_\perp$ and $CTerm_\perp$ of *partial* expressions and c-terms resp. The intended meaning is that Exp and $Exp_\perp$ stand for evaluable expressions, i.e., expressions that can contain function symbols, while $CTerm$ and $CTerm_\perp$ stand for data terms representing total and partial values resp.

[1] We use the terminology Exp (for general expressions) instead of the more usual $Term$ in order to highlight the syntactic (and semantic) difference with $CTerm$ (data values).

The *shell* $|e|$ of an expression e represents the outer constructed part of e and is defined as: $|X| = X$; $|c(e_1,\ldots,e_n) = c(|e_1|,\ldots,|e_n|)$; $|f(e_1,\ldots,e_n)| = \bot$.

Substitutions $\theta \in Subst$ are mappings $\theta : \mathcal{V} \to Exp$, extending naturally to $\theta : Exp \to Exp$.

One-hole contexts are defined as $Cntxt \ni \mathcal{C} ::= [\,] \mid h(e_1,\ldots,\mathcal{C},\ldots,e_n)$, with $h \in \Sigma^n$. The application of a context $\mathcal{C}$ to an expression e, written by $\mathcal{C}[e]$, is defined inductively as $[\,][e] = e$ and $h(e_1,\ldots,\mathcal{C},\ldots,e_n)[e] = h(e_1,\ldots,\mathcal{C}[e],\ldots,e_n)$.

The approximation ordering $\sqsubseteq$ is defined on expressions as the least partial ordering satisfying: *i)* $\bot \sqsubseteq e$ for all $e \in Exp_\bot$, and *ii)* $e \sqsubseteq e' \Rightarrow \mathcal{C}[e] \sqsubseteq \mathcal{C}[e']$ for all $e, e' \in Exp_\bot, \mathcal{C} \in Cntxt$.

A *constructor-based term rewriting system* $\mathcal{P}$ (*CS*, also called *program* along this paper) is a set of rewrite rules of the form $f(\bar{t}) \to e$ where $f \in FS^n$, $e \in Exp$, $var(e) \subseteq var(\bar{t})$, and $\bar{t}$ is a linear n-tuple of c-terms, where linearity means that variables occur only once in $\bar{t}$. Given a program $\mathcal{P}$, its associated rewrite relation $\to_\mathcal{P}$ is defined as: $\mathcal{C}[l\theta] \to_\mathcal{P} \mathcal{C}[r\theta]$ for any context $\mathcal{C}$, rule $l \to r \in \mathcal{P}$ and $\theta \in Subst$. We write $\overset{*}{\to}_\mathcal{P}$ for the reflexive and transitive closure of the relation $\to_\mathcal{P}$. In the following, we will usually omit the reference to $\mathcal{P}$.

3 A semantics for CS

In this section we present our proposed semantics, which has a logic flavor as it is based on a proof calculus. The use of proof calculi to specify the semantics of rewriting formalisms is not unfrequent. Two well-known cases correspond to the frameworks of rewriting logic [14] and CRWL [9]. We have been inspired by the philosophy of the latter, according to the following roadmap:

– We first identify the 'finite pieces' of which the denotation of expressions should be made of. In our case these will be the *s-cterms* introduced in Sect. 3.1, capturing technically the idea of 'packaging sets below constructor' mentioned in Sect. 1.

– Then, we devise a proof calculus able to prove statements of the form $e \twoheadrightarrow st$ expressing that st is a finite approximation to the denotation of e. This will be done in Sect. 3.2, although technically expressions will be generalized to the more general *s-expressions*.

– The proof calculus induces a natural notion of denotation of expression: it is simply the set of its provable approximations. The fact of working with finite approximations makes unnecessary to use a background of cpo's and powerdomains. This was found greatly convenient in the CRWL framework, and it is even more so in our case, where recursive nestings of constructors and sets occur. Moreover, it is known ([5, 15]) that an approach based on semantic domains with infinite (limit) elements and using fixpoint techniques has technical limitations ([5, 15]).

– If the proof calculus is designed to have a 'compositional' aspect, then one can expect compositionality of the resulting semantics, and the proof calculus is in itself a great aid to prove it. We have pursued this design principle in our proof calculus; as a result, and we have been able to prove compositionality and other relevant properties of the semantics (Sect. 3.2).

– Now, since or aim is to develop a new semantics for standard rewriting, not to give a new notion of rewriting, it is essential to show that our semantics is indeed related to rewriting: this is done in Sect. 3.3 by correctness and completeness results.

– Finally, with all the previous results and an extra little effort, we are able to prove full abstraction of our semantics (Sect. 4).

3.1 SCTerms: the pieces of the semantics

In this section we define new syntactic notions (of expressions, cterms, etc) in order to pack different values coming from non deterministic reductions at the syntactic level, by introducing sets in the corresponding syntax. Values become *s-cterms* that must be defined in mutual recursion with *elemental s-cterms*:

$$\begin{aligned} &ESCTerm \ni est ::= X \mid c(st_1,\ldots,st_n) \\ &\qquad \text{for } X \in \mathcal{V}, c \in DC^n, st_1,\ldots,st_n \in SCTerm \\ &SCTerm \ni st ::= \emptyset \mid \{est_1,\ldots,est_n\} \\ &\qquad \text{for } n > 0, est_1,\ldots,est_n \in ESCTerm \end{aligned}$$

A s-cterm is a *finite* set of elemental s-cterms, that are variables or constructors applied to s-cterms. The aim of these values is to capture the reduction of a non deterministic expression like $c(amb(a,b))$ into the single value

$\{c(\{a,b\})\}$. With the same idea, but allowing function symbols we define *elemental s-expressions* and *s-expressions* as:

$$\begin{array}{l} ESExp \ni ese ::= X \mid h(se_1,\ldots,se_n) \\ \qquad \text{for } X \in \mathcal{V}, h \in \Sigma^n, se_1,\ldots,se_n \in SExp \\ SExp \ni se ::= \emptyset \mid \{ese_1,\ldots,ese_n\} \\ \qquad \text{for } n>0, ese_1,\ldots,ese_n \in ESExp \end{array}$$

In the definition of $SCTerm$ (and also in $SExp$), the base case $\emptyset$ could be hidden in the brace notation $\{est_1,\ldots,est_n\}$ just permitting $n=0$ (in fact, we will do it sometimes). We have preferred to emphasize the presence of $\emptyset$, which plays the role of the undefined value (similar to $\bot$ for Exp in Sect. 2). Therefore s-cterms and s-expressions should be understood as *partial*. *Total* s-expressions and s-cterms would not use $\emptyset$, but they do not play any significant role in the following.

We can flatten a s-expression se to obtain the set $flat(e)$ of expressions "contained" in it: $flat(\emptyset)=\{\bot\}$ and $flat(se)=\bigcup_{ese\in se} flat(ese)$ if $se\neq\emptyset$, where $flat$ for elemental s-expressions is defined as $flat(X)=\{X\}$; $flat(h(se_1,\ldots,se_n))=\{h(e_1,\ldots,e_n)|e_i\in flat(se_i) \text{ for } i=1..n\}$.

The set $SSubst$ of *s-substitutions* consists of mappings $\sigma:\mathcal{V}\to SExp$. The *domain* of a s-substitution σ is defined as $dom(\sigma)=\{X \mid \sigma(X)\neq\{X\}\}\}$. Notice that s-substitutions replace variables by s-expressions (which are sets), and some care must be taken when extending s-substitutions to $eSExp$ and $SExp$:

$$\begin{array}{ll} \sigma : eSExp \to SExp & \sigma : SExp \to SExp \\ X\sigma = \sigma(X) & \{ese_1,\ldots,ese_n\}\sigma = \bigcup_{i\in\{1..n\}} ese_i\sigma \\ h(\overline{se})\sigma = \{h(\overline{se\sigma})\} & \end{array}$$

The set $SCSubst$ of *s-csubstitutions* consists of of mappings $\sigma:\mathcal{V}\to SCTerm$ and the extensions to the domains of $eSCTerm$ and $SCTerm$ are defined analogously to the previous extensions.

One hole (elemental) *s-contexts* are defined as:

$$sCntxt \ni s\mathcal{C} ::= [\,] \mid \{\ldots,h(\ldots,s\mathcal{C},\ldots),\ldots\} \quad \text{with } h\in\Sigma \text{ and } s\mathcal{C}\in sCntxt$$

The application of a context to a s-expression is defined in the natural way. Notice that s-contexts allow only the hole to be in the place of a sub-s-expression. For example, the possible s-contexts of $\{Y,c(\{X\})\}$ are $[\,]$ and $\{Y,c([\,])\}$, but not $\{[],c(\{X\})\}$ neither $\{Y,[]\}$.

The preorder $\sqsubseteq$ is defined for s-expressions as the least preorder satisfying: $se\sqsubseteq se'$ if $\forall ese\in se.\exists ese'\in se'$ such that $ese\sqsubseteq ese'$, where for elemental s-expressions $\sqsubseteq$ is defined as the least preorder such that: $X\sqsubseteq X$ for any $X\in\mathcal{V}$ and $h(se_1,\ldots,se_n)\sqsubseteq h(se'_1,\ldots,se'_n)$ iff $se_i\sqsubseteq se'_i$ for $i=1..n$. For s-substitutions, the preorder is defined as $\sigma\sqsubseteq\sigma'$ if $\sigma(X)\sqsubseteq\sigma'(X)$ for $X\in\mathcal{V}$.

Programs are exactly those defined in Sect. 3. The proof calculus of the next section needs to use function rules transformed into the new syntactical framework of s-expressions. For this purpose we define the transformation of $e\in Exp$ into a s-expression $\tilde{e}\in SExp$ as: $\tilde{\bot}=\emptyset$; $\tilde{X}=\{X\}$ for any $X\in\mathcal{V}$; $\widetilde{h(e_1,\ldots,e_n)}=\{h(\tilde{e_1},\ldots,\tilde{e_n})\}$, with $h\in\Sigma^n$. The transformation $\tilde{\mathcal{C}}$ of a context $\mathcal{C}$ is defined in the natural way and its application to a s-expression is defined as $\widetilde{\mathcal{C}[e]}=\tilde{\mathcal{C}}[\tilde{e}]$. On the other hand, $\tilde{\sigma}$ is defined as $\tilde{\sigma}(X)=\widetilde{\sigma(X)}$, for $\sigma\in Subst$.

3.2 A Proof Calculus

Our goal in this section is to devise a proof calculus to specify which $SCTerm$'s correspond to a given expression under a given CS. To do that we will inspire in the *CRWL* proof calculus [9], adapting it to serve our purposes. Therefore the expressions will be evaluated in an innermost way and we will avoid the use of any transitivity rule, in order for the calculus to be compositional in the values it computes. But as we will use partial s-cterms as values then this innermost evaluation will not induce the strictness of functions, hence enabling the completeness of our semantics wrt term rewriting even for non-terminating CS.
Besides, during parameter passing the variables in the rewrite rules will be instanciated with partial s-cterms. As a consequence it is possible to end up evaluating expressions with some $SCTerm$ "inside" (as a subexpression), even when starting the computation from an ordinary $e\in Exp$. So, instead of dealing only with expressions from Exp, our calculus will compute the partial s-cterms corresponding to any given partial s-expression. Finally, the mapping $\tilde{}$ will be used in combination with our logic to get the $SCTerm$'s corresponding to a given Exp.

E	$se \twoheadrightarrow \emptyset$	
RR	$\{X\} \twoheadrightarrow \{X\}$	if $X \in \mathcal{V}$
DC	$\dfrac{se_1 \twoheadrightarrow st_1 \ \ldots \ se_n \twoheadrightarrow st_n}{\{c(se_1,\ldots,se_n)\} \twoheadrightarrow \{c(st_1,\ldots,st_n)\}}$	if $c \in CS$
More	$\dfrac{se \twoheadrightarrow st_1 \ \ldots se \twoheadrightarrow st_n}{se \twoheadrightarrow st_1 \cup \ldots \cup st_n}$	
Less	$\dfrac{\{esa_1\} \twoheadrightarrow st_1 \ \ldots \ \{esa_m\} \twoheadrightarrow st_m}{\{ese_1,\ldots,ese_n\} \twoheadrightarrow st_1 \cup \ldots \cup st_m}$	if $n \geq 2, m > 0$, for any $\{esa_1,\ldots,esa_m\} \subseteq \{ese_1,\ldots,ese_n\}$
ROR	$\dfrac{se_1 \twoheadrightarrow \tilde{p_1}\theta \ \ldots \ se_n \twoheadrightarrow \tilde{p_n}\theta \ \ \tilde{r}\theta \twoheadrightarrow st}{\{f(se_1,\ldots,se_n)\} \twoheadrightarrow st}$	if $\begin{array}{l}(f(p_1,\ldots,p_n) \to r) \in \mathcal{P} \\ \theta \in SCSubst\end{array}$

Fig. 1. A proof calculus for constructor systems

To be precise, our proof calculus will prove reduction statements of the form $se \twoheadrightarrow st$ with $se \in SExp$ and $st \in SCTerm$, expressing that st represents an approximation to one of the possible structured sets of values for se. This calculus is presented in Fig. 1. Rule E (empty) allows us to avoid the evaluation of any expression, in order to get a non-strict semantics. Rules RR (restricted reflexivity) and DC (decomposition) work with singleton sets and allow us to reduce any variable to itself, and to decompose the evaluation of a constructor-rooted elemental s-expression. Rule More allows us to compute more than one value for an s-expression, and to collect these values together. Rule Less allows us to discard some elemental s-expressions from the s-expression under evaluation. Finally rule ROR (run-time[2] outer reduction) expresses that to evaluate a function call we must first evaluate its arguments to get an instance of a program rule, perform parameter passing (by means of a $SCSubst\ \theta$) and then reduce the instantiated right-hand side. The use of $SCSubst$'s is fundamental to get the exact behaviour of term rewriting, because then the branching information associated to the computation of each $\tilde{p_i}\theta$ is not lost in some kind of flattening to a set of c-terms, but kept into the structured representation of $SCTerm$'s.

We write $\mathcal{P} \vdash se \twoheadrightarrow st$ to express that $se \twoheadrightarrow st$ is derivable in our calculus under the CS $\mathcal{P}$. The *denotation* of a s-expression se under a CS $\mathcal{P}$ is defined as $[\![se]\!]^{\mathcal{P}} = \{st \in SCTerm \mid \mathcal{P} \vdash se \twoheadrightarrow st\}$. In the following we will usually omit the reference to $\mathcal{P}$.

Example 2. Consider the CS of Ex. 1, we can use our calculus to prove the statement $f(c(\widetilde{amb(a,b)})) \twoheadrightarrow \widetilde{d(a,b)}$ (some steps have been omitted for the sake of conciseness):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\{a\} \twoheadrightarrow \{a\}}\ \text{DC} \quad \dfrac{}{\{b\} \twoheadrightarrow \emptyset}\ \text{E} \quad \dfrac{\ldots}{\{a\} \twoheadrightarrow \{a\}}}{\{amb(\{a\},\{b\})\} \twoheadrightarrow \{a\}}\ \text{ROR} \quad \dfrac{\ldots}{\{amb(\{a\},\{b\})\} \twoheadrightarrow \{b\}}\ \text{ROR}}{\{amb(\{a\},\{b\})\} \twoheadrightarrow \{a,b\}}\ \textsc{More}}{\dfrac{\{c(\{amb(\{a\},\{b\})\})\} \twoheadrightarrow \{c(\{a,b\})\}}\ \text{DC} \quad \dfrac{(*)}{\{d(\{a,b\},\{a,b\})\} \twoheadrightarrow \{d(\{a\},\{b\})\}}\ \text{DC}}\ \text{ROR}}{f(c(\widetilde{amb(a,b)})) \equiv \{f(\{c(\{amb(\{a\},\{b\})\})\})\} \twoheadrightarrow \{d(\{a\},\{b\})\} \equiv \widetilde{d(a,b)}}$$

where $(*)$ is the derivation:

$$\dfrac{\dfrac{\dfrac{}{\{a\} \twoheadrightarrow \{a\}}\ \text{DC}}{\{a,b\} \twoheadrightarrow \{a\}}\ \textsc{Less} \quad \dfrac{\ldots}{\{a,b\} \twoheadrightarrow \{b\}}}{\{d(\{a,b\},\{a,b\})\} \twoheadrightarrow \{d(\{a\},\{b\})\}}\ \text{DC}$$

On the other hand, $\widetilde{d(a,b)}$ is not a correct value for $f(amb(\widetilde{c(a)}, c(b)))$, because in that expression the evaluation of $amb(\widetilde{c(a)}, c(b))$ has to be performed in order to get a value matching the argument of the left-hand side of the only rule for f, and the only matching values for it are $\widetilde{c(a)}$, $\widetilde{c(b)}$ and $\{c(\emptyset)\}$, as for example $\{c(\{a\}), c(\{b\})\}$ does not match $\widetilde{c(X)}$.

Notice that, structurally, a denotation $[\![se]\!]$ is a possibly infinite set of s-cterms, each one being a finite set of elemental s-cterms. Infinite denotations might appear with non-terminating programs. Notice, however, that the

[2] The prefix 'run-time' comes from 'run-time choice', which is often applied ([12, 9]) to the parameter passing mechanism of term rewriting.

elements are s-cterms that are, by construction, finite objects. Thus, we avoid the presence of infinite values as elements of denotations.

As we anticipated above, even when proving a reduction $\tilde{e} \twoheadrightarrow \tilde{t}$ for $e \in Exp$ and $t \in CTerm$, we may find premises of the shape $se \twoheadrightarrow st$ for $se \in SExp, st \in SCTerm$, because the substitutions used for parameter passing in ROR may introduce sets in $\tilde{r}\theta$, as we can see in the second premise of the first application of ROR, in Ex. 2. But in fact the kind of s-expressions that we may find in the proof for some reduction for an expression is more restricted. It is easy to prove that in any proof for any statement $\tilde{e} \twoheadrightarrow st$ with $e \in Exp$ we have that in any premise $se' \twoheadrightarrow st'$ for it, $se \in trSExp$, a set defined as $trSExp \ni tr ::= st \mid \{h(tr_1, \ldots, tr_n)\}$, with $st \in SCTerm, h \in \Sigma, tr_1, \ldots, tr_n \in trSExp$.
This suggests that we could have defined our logic to prove reductions $se \twoheadrightarrow st$ with $se \in trSExp$ and $st \in SCTerm$ only, but we think that it is more profitable to define it to deal with the more general case of $se \in SExp$. First of all, then we get a logic that can handle a more general kind of syntactic objects, and therefore that could be used to express other formalism apart from term rewriting. We could slightly modify the rule ROR to accept not only CS's but in general *"s-expression rewriting systems" (sCS's)*, consisting of rules $\{f(st_1, \ldots, st_n)\} \twoheadrightarrow se$. This way the original formulation of ROR becomes a particular case of the new version, that works with CS's adapted to sCS's by means of $\tilde{\ }$. This would be similar to what is done in [16] to express term rewriting, term graph rewriting and noncopying rewriting by means of the more general framework of marked term rewriting. We consider this an interesting possible subject of future work.

On the other hand, working with reductions of s-expressions allows us to formulate more general and powerful results about the semantics, which become easier to prove because of its generality (which gives us stronger induction hypotheses), and that can be then easily applied to the more restricted case. These are nice properties like the following *polarity* property of our semantics. In all the results of this and next section we assume a given CS and omit mentioning it.

Proposition 1 (Polarity). *Let $se, se' \in SExp$, $st, st' \in SCTerm$. If $se \sqsubseteq se'$ and $st' \sqsubseteq st$ then $st \in [\![se]\!]$ implies $st' \in [\![se']\!]$.*

Our semantics also enjoys the following monotonicity property related to substitutions. It is formulated for the preorder $\sqsubseteq$ and for the preorder $\trianglelefteq$ over $SSubst$, defined by $\sigma \trianglelefteq \sigma'$ iff $\forall X \in \mathcal{V}, [\![\sigma(X)]\!] \subseteq [\![\sigma'(X)]\!]$.

Proposition 2 (Monotonicity of substitutions). *Let $se \in SExp$, $\sigma, \sigma' \in SSubst$. If $\sigma \trianglelefteq \sigma'$ or $\sigma \sqsubseteq \sigma'$ then $[\![se\sigma]\!] \subseteq [\![se\sigma']\!]$.*

One of the most important properties of our logic is its *compositionality*, a property very close to the DET-additivity property for algebraic specifications of [12], which will be one of the keys for full abstraction.

Theorem 1 (Compositionality). *For all $s\mathcal{C} \in sCntxt, se \in SExp$,*

$$[\![s\mathcal{C}[se]]\!] = \bigcup_{st \in [\![se]\!]} [\![s\mathcal{C}[st]]\!]$$

As a consequence: $[\![se]\!] = [\![se']\!] \Leftrightarrow \forall s\mathcal{C}.\ [\![s\mathcal{C}[se]]\!] = [\![s\mathcal{C}[se']]\!]$.

Regarding *closedness* under susbtitutions, as we use $SCSubst$ for parameter passing it is natural to have closedness of reductions under this type of substitutions. Besides, as rewriting is closed under $Subst$ it is expected to have some kind of closedness for $Subst$ too. But in general it is not true that for any $st \in SCTerm, \sigma \in SSubst$ we have $st\sigma \in SCTerm$, therefore it makes no sense to expect that $se \twoheadrightarrow st$ implies $se\sigma \twoheadrightarrow st\sigma$, as the reductions in our logic are from $SExp$ to $SCTerm$. Nevertheless we still can say something about that, as we can see in the following property.

Proposition 3 (Closedness under substitutions). *Let $se \in SExp$, $st \in SCTerm$. If $st \in [\![se]\!]$ then: a) $\forall \theta \in SCSubst, st\theta \in [\![se\theta]\!]$ b) $\forall \sigma \in SSubst, [\![st\sigma]\!] \subseteq [\![se\sigma]\!]$*

All these properties are powerful tools to reason about the denotations of s-expression. And this reasoning power is transfered to the term rewriting universe through the adequacy results that we will see in the next section, thus opening paths for the development of new reasoning techniques for constructor systems.

$\_ \vdash \_ \lessdot \_ \subseteq CS \times SCTerm \times Exp$	$\_ \vdash \_ \lessdot \_ \subseteq CS \times ESCTerm \times Exp$
$\mathcal{P} \vdash st \lessdot e$ if $\forall est \in st, \mathcal{P} \vdash est \lessdot e$	$\mathcal{P} \vdash X \lessdot e$ if $\mathcal{P} \vdash e \to^* X$
	$\mathcal{P} \vdash c(\overline{st}) \lessdot e$ if $\mathcal{P} \vdash e \to^* c(\overline{e})$ for some $\overline{e}$ such that $\forall e_i \in \overline{e}, \mathcal{P} \vdash st_i \lessdot e_i$

Fig. 2. Domination relation

3.3 Relation with rewriting

The nice properties of our logic could reveal pretty useless if not accompanied by strong adequacy results that relate this logic to the term rewriting relation. In the present section we will formally state and prove these results.

The keys to prove this adequacy are the following lemmas, whose meaning will be clarified later on.

Lemma 1. *Let $\sigma \in SSubst, se \in SExp, st \in SCTerm$. If $se\sigma \twoheadrightarrow st$ then there exists $\theta \in [\![\sigma]\!]$ such that $se\theta \twoheadrightarrow st$.*

Lemma 2. *Let $e \in Exp, st \in SCTerm, \theta \in SCSubst$. If $\tilde{e}\theta \twoheadrightarrow st$ then $st \lessdot e\sigma$ for any $\sigma \in Subst$ such that $\theta \lessdot \sigma$.*

First of all we want our logic to be *complete*, that for any expression our semantics could capture any c-term reachable from it by rewriting. This is the first result we get about that:

Proposition 4. *For all $e, e' \in Exp$, if $e \to^* e'$ then $[\![\tilde{e'}]\!] \subseteq [\![\tilde{e}]\!]$.*

We can prove it for one step (for $e \to e'$) and then just generalize it to many steps by induction on the length of $e \to^* e'$. The keys are the compositionality of Theorem 1, and Lemma 1, which expresses that in any reduction $se\sigma \twoheadrightarrow st$ only a finite amount of the information contained in σ is needed. We formalize it through the notion of denotation of a $SSubst$, defined as $[\![\sigma]\!] = \{\theta \in SCSubst \mid \forall X \in \mathcal{V}, \sigma(X) \twoheadrightarrow \theta(X)\}$. We can use these tools as follows:

Proof (Sketch). Assume $e \to e'$, if the step was performed at the root then we have $e \equiv f(\overline{p})\sigma \to r\sigma \equiv e'$ for some rule $(f(\overline{p}) \to r) \in \mathcal{P}$. Now assume $\widetilde{r\sigma} \twoheadrightarrow st$, then as $\widetilde{r\sigma} \equiv \tilde{r}\tilde{\sigma}$, by Lemma 1 $\exists \theta \in [\![\tilde{\sigma}]\!]$ such that $\tilde{r}\theta \twoheadrightarrow st$, but then it is easy to prove that $\widetilde{f(\overline{p})}\theta \twoheadrightarrow st$. Besides $\theta \in [\![\tilde{\sigma}]\!]$ implies $\theta \trianglelefteq \tilde{\sigma}$, and so we can apply Prop. 2 to get $\widetilde{f(\overline{p})\sigma} \equiv \widetilde{f(\overline{p})}\tilde{\sigma} \twoheadrightarrow st$.
If the step was not performed at the root then we have $e \equiv \mathcal{C}[f(\overline{p})\sigma] \to \mathcal{C}[r\sigma] \equiv e'$, and so we can combine the result for the previous case with the compositionality of Theorem 1 to get the desired result.

Now we can apply Prop. 4 to get the following strong completeness result.

Theorem 2 (Completeness). *For all $e, e' \in Exp, t \in CTerm$,*
a) $e \to^* e'$ implies $\widetilde{|e'|} \in [\![\tilde{e}]\!]$ *b) $e \to^* t$ implies $\widetilde{t} \in [\![\tilde{e}]\!]$*

Proof. Concerning *a)*, it is easy to prove that $\forall e \in Exp, \tilde{e} \twoheadrightarrow \widetilde{|e|}$, by induction on the structure of Exp. But then $\widetilde{|e'|} \in [\![\tilde{e'}]\!] \subseteq [\![\tilde{e}]\!]$ by Prop. 4. Concerning *b)*, we can prove $\forall t \in CTerm, |t| \equiv t$ by induction on $CTerm$, and so $\tilde{e} \twoheadrightarrow \widetilde{|t|} \equiv \widetilde{t}$, by *a)*.

We also want our logic to be *correct*, that for any expression our semantics could not compute more c-terms than those reachable by rewriting. One key ingredient will be the *domination relation* $\_ \lessdot \_$ defined in Fig. 2 (we will omit the prefix "$\vdash \mathcal{P}$" when it is implied by the context). With this relation we try to transfer to the rewriting world the finer distinction between sets of values that the structured representation of $SCTerm$ allows us to perform. This way under the CS of Ex. 1 we have $\{c(\{0,1\})\} \lessdot c(amb(0,1))$ but not $\{c(\{0,1\})\} \lessdot amb(c(0), c(1))$. The domination relation $\_ \lessdot \_$ has a strong relation with our semantics, as stated in the following result:

Lemma 3 (Domination). *For all $e \in Exp, st \in SCTerm$: $st \in [\![\tilde{e}]\!]$ iff $st \lessdot e$.*

But notice that $\_ \lessdot \_$ only talks about reductions for $\tilde{e}$ with $e \in Exp$, and so it cannot be used to formulate properties like those seen in Sect. 3.2, although it inherits them through Lemma 3.
The key to prove Lemma 3 is Lemma 2, in which we extend the relation $\_ \lessdot \_$ to $\_ \vdash \_ \lessdot \_ \subseteq CS \times SCSubst \times Subst$ by $\theta \lessdot \sigma$ iff $\forall X \in \mathcal{V}, \theta(X) \lessdot \sigma(X)$. Lemma 2 expresses that given a reduction $\tilde{e}\theta \twoheadrightarrow st$ then any substitution σ that contains at least the same information as θ can be used to dominate st. Note we have also used the domination to encode this "containment of at least the same information", hence, as $\_ \lessdot \_$ is a rewriting-based notion, we have been able to change from the universe of our logic to the universe of rewriting along the way.

Proof (For Lemma 3, sketch). The left to right direction is a trival corollary of Lemma 2, just taking $\theta = \epsilon = \sigma$. The converse implication follows from a simple induction on the proof for $st \lessdot e$, using the completeness of our logic as stated in Prop. 4.

The good thing about $\_ \lessdot \_$ is that it already has a strong conection with rewriting, as it is defined by means of rewriting derivations. Hence we can perform a simple induction on the structure of $SCTerm$ and $ESCTerm$ to prove the following result, which uses the notion of flattening defined in Sect. 3.1.

Lemma 4. *Let $st \in SCTerm$, $est \in ESCTerm$, $e \in Exp$, and assume $t \in flat(st)$. If $st \lessdot e$ then $e \to^* e'$ for some $e' \in Exp$ such that $t \sqsubseteq |e'|$.*

And now we are ready to state and prove our main correctness result.

Theorem 3 (Correctness). *Let $e \in Exp, st \in SCTerm, t \in CTerm_\perp$:*

a) If $st \in [\![\tilde{e}]\!]$ and $t \in flat(st)$, then $e \to^* e'$ for some $e' \in Exp$ such that $t \sqsubseteq |e'|$.
b) If $\tilde{t} \in [\![\tilde{e}]\!]$, then $e \to^* e'$ for some $e' \in Exp$ such that $t \sqsubseteq |e'|$.
c) Besides, in a) or b), if $t \in CTerm$, then $e \to^* t$.

Proof. We get *a)* just chaining Lemma 3 and Lemma 4. Concerning *b)*, we can prove that $\forall t \in CTerm_\perp, flat(\tilde{t}) = \{t\}$ by induction on $CTerm_\perp$, and chain it with *a)*. Finally *c)* is a consequence of *a)* and *b)*, because if $t \in CTerm$ then it is maximal wrt $\sqsubseteq$, hence $t \sqsubseteq |e'|$ implies $t \equiv |e'|$. But that implies there is no $\perp$ in $|e'|$, therefore $e' \in CTerm$ and $e' \equiv |e'| \equiv t$, and so $e \to^* e' \equiv t$.

4 Full abstraction

The semantics $[\![se]\!]^{\mathcal{P}}$ of Sect. 3 is defined for s-expressions, but induces naturally a notion of semantics for ordinary expressions $e \in Exp$:

$$[\![e]\!]^{\mathcal{P}}_S = [\![\tilde{e}]\!]^{\mathcal{P}}(= \{st \in SCTerm | \tilde{e} \twoheadrightarrow st\})$$

In this section we discuss full abstraction in the context of CS and show that $[\![\_]\!]_S$ achieves it, in contrast to semantics directly based on sets of results, informally described in Sect. 1.

The property of *full abstraction* expresses a perfect capture of observational behavior: in a fully abstract semantics, two expressions have the same semantics if and only if they are observationally indistinguishable. For this to be meaningful, one must choose a criterion of observability. The problem of full abstraction was first investigated by Plotkin [17] in connection to PCF (a simple functional language), and since then is a standard topic when dealing with program semantics (see e.g. [18]). It is common to adjust its definition to the characteristics of the language under consideration. In the context of functional-like programming languages like PCF the condition for full abstraction is usually stated as:

$$[\![e]\!] = [\![e']\!] \Leftrightarrow \mathcal{O}(\mathcal{C}[e]) = \mathcal{O}(\mathcal{C}[e']), \text{for any context } \mathcal{C} \tag{1}$$

where $\mathcal{O}$ is the observation function of interest. Programs do not need to be mentioned, because programs and expressions can be identified by contemplating the evaluation of e under $\mathcal{P}$ as the evaluation of a big λ-expression or big *let*-expression embodying $\mathcal{P}$ and e. Contexts pose no problems either. In our case, since programs (CS) are kept different from expressions, some care must be taken. It might happen that $\mathcal{P}$ has not enough syntactical elements and rules to built interesting distinguishing contexts. For instance, if in Ex. 1 we drop the definition of f, then we cannot built a context that distinguishes $c(a?b)$ from $c(a)?c(b)$. This would imply that soundness or full abstraction would not be intrinsic to the semantics, but would greatly depend on the program. What we need is requiring the right part of (1) to hold for all contexts that might be obtained by extending $\mathcal{P}$ with new auxiliary functions. To be more precise, we say that $\mathcal{P}'$ is a *safe extension* of $(\mathcal{P}, e)$ if $\mathcal{P}' = \mathcal{P} \cup \mathcal{P}''$, where $\mathcal{P}''$ does not include defining rules for any function symbol occurring in $\mathcal{P}$ or e. Any sensible notion of semantics should verify $[\![e]\!]^{\mathcal{P}} = [\![e]\!]^{\mathcal{P}'}$ when $\mathcal{P}'$ safely extends (P, e). This happens indeed for all the semantics considered below.

Things are now prepared to give the following definition:

Definition 1 (Observations, full abstraction).
(a) A semantic function (a semantics*, in short) is a function* $[\![\_]\!]$ *associating a semantic value (taken from a set* $\mathcal{D}$*) to each expression* e *under a given program* $\mathcal{P}$*. We write* $[\![e]\!]^{\mathcal{P}}$ *for such value.*
(b) An observation function *is a function* $\mathcal{O}$ *associating a set of observation values (or* observables*, taken from a set Obs) to each expression* e *under a given program* $\mathcal{P}$*. We write* $\mathcal{O}^{\mathcal{P}}(e)$ *for it.*
(c) A semantics is fully abstract *wrt* $\mathcal{O}$ *iff for any* $\mathcal{P}$ *and* $e, e' \in Expr$*, the following two conditions are equivalent:*
(i) $[\![e]\!]^{\mathcal{P}} = [\![e']\!]^{\mathcal{P}}$ *(ii)* $\mathcal{O}^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}^{\mathcal{P}'}(\mathcal{C}[e'])$ *for any* $\mathcal{P}'$ *safely extending* $(\mathcal{P}, e)$*,* $(\mathcal{P}, e')$ *and any* $\mathcal{C}$ *built with the signature of* $\mathcal{P}'$*.*
In words: semantic equality is equivalent to observational indistinguishability.
(d) A notion weaker than full abstraction is: a semantics is sound *wrt* $\mathcal{O}$ *iff the condition* (i) *above implies the condition* (ii)*.*
In words: semantic equality implies observational indistinguishability.
(e) A semantics is compositional *iff for any* $\mathcal{P}$ *and* $e, e' \in Expr$*, the following two conditions are equivalent:*
(i) $[\![e]\!]^{\mathcal{P}} = [\![e']\!]^{\mathcal{P}}$ *(ii)* $[\![\mathcal{C}[e]]\!]^{\mathcal{P}} = [\![\mathcal{C}[e']]\!]^{\mathcal{P}}$ *for any* $\mathcal{C}$ *built with the signature of* $\mathcal{P}$*.*
In words: the semantics of an expression depends only on the semantics of its subexpressions. Notice that $(ii) \Rightarrow (i)$ *holds trivially (take* $\mathcal{C} = [\ ]$*).*

In the next definition we collect some notions of semantics and observables for the case of CS. $[\![\_]\!]_S$ and $[\![\_]\!]_{S'}$ are our new contributed semantics; the rest are the 'obvious' semantics of Sect. 1. As usual, we omit the program $\mathcal{P}$ in notations.

Definition 2 (Semantics and observations for CSs).
We consider the following semantics for expressions $e \in Expr$*:*

$[\![e]\!]_{rw} = \{e' \mid e \rightarrow^* e'\}$ $\quad$ $[\![e]\!]_{nf} = \{e' \mid e \rightarrow^* e', e' \text{ in normal form}\}$
$[\![e]\!]_{t} = \{t \in CTerm \mid e \rightarrow^* t\}$ $\quad$ $[\![e]\!]_{t_\perp} = \{t \in CTerm_\perp \mid \exists e'.(e \rightarrow^* e' \wedge t \sqsubseteq |e'|)\}$
$[\![e]\!]_{S} = [\![\tilde{e}]\!]$ $\quad$ $[\![e]\!]_{S'} = \bigcup_{st \in [\![e]\!]_S} st.$

We consider the following observation functions for expressions $e \in Expr$*:*

$\mathcal{O}_t(e) = [\![e]\!]_t$ $\quad$ $\mathcal{O}_{t_\perp}(e) = [\![e]\!]_{t_\perp}$ $\quad$ $\mathcal{O}_{true}(e) = [\![e]\!]_t \cap \{true\}$

Some remarks:
• It is clear that $[\![e]\!]_t \subseteq [\![e]\!]_{nf} \subseteq [\![e]\!]_{rw}$, and also $[\![e]\!]_t \subseteq [\![e]\!]_{t_\perp}$.
• Notice that some of the sets above can play at the same time the role of semantic values and of observation values.
• $[\![e]\!]_S$ was introduced at the beginning of the this section. $[\![e]\!]_{S'}$ is a simplified variant, making more readable the semantics of particular expressions, because $[\![e]\!]_S$ is a set of finite sets of $est's$, while $[\![e]\!]_{S'}$ is simply a set of $est's$. However $[\![\_]\!]_S$ has been technically more convenient for proving properties of the semantics, due to its more direct connection to the proof calculus. Both semantics are essentially the same, as evidenced by:

Proposition 5. *For any* $e, e' \in Exp$*,* $[\![e]\!]_S = [\![e']\!]_S \Leftrightarrow [\![e]\!]_{S'} = [\![e']\!]_{S'}$

The next result shows that, although $\mathcal{O}_t, \mathcal{O}_{t_\perp}, \mathcal{O}_{true}$ define different observations, it is irrelevant which is chosen, as far as full abstraction is concerned.

Proposition 6. *Assume a given semantics* $[\![\_]\!]$*. Then:*
(a) $[\![\_]\!]$ *is fully abstract wrt* $\mathcal{O}_t \Leftrightarrow [\![\_]\!]$ *is fully abstract wrt* $\mathcal{O}_{t_\perp}$*.*
(b) If expressions to be observed are restricted to be ground, then:
$[\![\_]\!]$ *is fully abstract wrt* $\mathcal{O}_t \Leftrightarrow$ *is fully abstract wrt* $\mathcal{O}_{t_\perp} \Leftrightarrow$ *is fully abstract wrt* $\mathcal{O}_{true}$

The groundness condition is necessary in *(b)*, as sketchly discussed here: in Th. 4 below we prove that $[\![\_]\!]_S$ is fully abstract wrt $\mathcal{O}_t$. However, it is not fully abstract wrt $\mathcal{O}_{true}$ if non-ground expressions are considered: for instance, $[\![X]\!]_S = \{\emptyset, \{X\}\} \neq [\![Y]\!]_S$. However, it can be shown (left linearity is essential here) that for any $\mathcal{C}$, $\mathcal{C}[X] \rightarrow^* true \Rightarrow \mathcal{C}[Y] \rightarrow^* true$.

Now we show that the first four semantics, $[\![]\!]_{rw}$-, $[\![\_]\!]_{nf}, [\![\_]\!]_t, [\![\_]\!]_{t_\perp}$ do not have good properties, as was informally discussed in Sect. 1. We use $\mathcal{O}_t$ in the result but, according to the previous result, $\mathcal{O}_{t_\perp}$ and $\mathcal{O}_{true}$ could be used instead (for ground expressions, in the latter case). This remark extends also to Th. 4 below.

Proposition 7.
(a) $[\![\_]\!]_{rw}, [\![\_]\!]_{nf}, [\![\_]\!]_t, [\![\_]\!]_{t_\perp}$ *are not fully abstract wrt* $\mathcal{O}_t$.
(b) Moreover, $[\![\_]\!]_{nf}, [\![\_]\!]_t, [\![\_]\!]_{t_\perp}$ *are not compositional, nor sound wrt* $\mathcal{O}_t$.
(c) $[\![\_]\!]_{nf}$ ($[\![\_]\!]_t$ *resp.) remains not compositional nor sound wrt* $\mathcal{O}_t$ *even if programs are restricted to be confluent (confluent and terminating, resp.).*

Proof. (a) For $[\![\_]\!]_{nf}, [\![\_]\!]_t, [\![\_]\!]_{t_\perp}$, (a) is implied by (b). For $[\![\_]\!]_{rw}$, just add a new function $g(X) \to f(X)$ to Ex. 1. It is easy to see that $f(a)$ and $g(a)$ are contextually indistinguishable wrt $\mathcal{O}_t$, but $[\![f(a)]\!]_{rw} \neq [\![g(a)]\!]_{rw}$.
(b) Example 1 serves for all the three semantics.
(c) For $[\![\_]\!]_{nf}$, consider the program $\{f \to f, g \to c(f), h(c(X)) \to a\}$. We have $[\![f]\!]_{nf} = [\![g]\!]_{nf} = \emptyset$, but $[\![h(f)]\!]_{nf} = \emptyset \neq \{a\} = [\![h(g)]\!]_{nf}$, proving at the same time not compositionality and unsoundness. For $[\![\_]\!]_t$, replace the above program by $\{f \to h(a), g \to c(f), h(c(X)) \to a\}$.

Finally, we show that our semantics do not present those problems.

Theorem 4 (Compositionality and full abstraction of $[\![\_]\!]_S$).
$[\![\_]\!]_S$ *and* $[\![\_]\!]_{S'}$ *are compositional, and fully abstract wrt* $\mathcal{O}_t$.

Proof. We prove the results for $[\![\_]\!]_S$. For $[\![\_]\!]_{S'}$, just use Prop. 5. Compositionality follows easily from definition of $[\![\_]\!]_S$ and compositionality of $[\![\_]\!]$ (Th. 1).
For full abstraction, let $\mathcal{P}$ be any CS, and $e, e' \in Expr$. We must prove:

$$[\![e]\!]^{\mathcal{P}}_S = [\![e']\!]^{\mathcal{P}}_S \Leftrightarrow \forall \mathcal{P}', \mathcal{C}\ .\mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e'])$$

where $\mathcal{P}'$ ranges over safe extensions of $(\mathcal{P}, e)$ and $(\mathcal{P}, e')$, and $\mathcal{C}$ over contexts built with the signature of $\mathcal{P}'$.
$\boxed{\Rightarrow}$ Assume $[\![e]\!]^{\mathcal{P}}_S = [\![e']\!]^{\mathcal{P}}_S$. Since $\mathcal{P}'$ is a safe extension, we know that $[\![e]\!]^{\mathcal{P}'}_S = [\![e']\!]^{\mathcal{P}'}_S$. We prove $\mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e]) \subseteq \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e'])$ (the other inclusion is similar). Let $t \in \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e])$, which means $\mathcal{C}[e] \to^*_{\mathcal{P}'} t$. By Th. 2 we know $\tilde{t} \in [\![\mathcal{C}[e]]\!]^{\mathcal{P}'}_S = [\![\mathcal{C}[e']]\!]^{\mathcal{P}'}_S$, where the last equality is justified by compositionality. But then, since $t \in flat(\tilde{t})$, we have (by Th. 3) that $\mathcal{C}[e'] \to^*_{\mathcal{P}'} t$, that is, $t \in \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e'])$, as desired.
For the other implication we need two auxiliary constructions enabling to build a context that distinguishes two expressions having different semantics:
• Given $st \in SCTerm$, we define a c-term $\widehat{st}$ 'mirroring' st as follows:

$$\widehat{\emptyset} = \langle\ \rangle_0 \qquad \widehat{\{X\}} = X\ (X \in \mathcal{V}) \qquad \widehat{\{c(\overline{st_i})\}} = c(\overline{\widehat{st_i}})$$
$$\widehat{\{est_1, \ldots, est_n\}} = \langle \widehat{\{est_1\}}, \ldots, \widehat{\{est_n\}} \rangle_n\ (n > 1)$$

where $\langle\_\rangle_n$ $(n \geq 0)$ are new tuple-forming constructor symbols. It is assumed here that $SCTerm, ESCTerm$ are equipped with any standard ordering.
• Given $st \in SCTerm$, the program $\mathcal{P}_{st}$ defines new functions $f_{st}, \ldots$ as follows:

$$f_\emptyset(X) \to \langle\ \rangle_0 \qquad f_{\{X\}}(U) \to U\ (X \in \mathcal{V}) \qquad f_{\{c(\overline{st_i})\}}(c(\overline{U_i})) \to c(\overline{f_{st_i}(U_i)})$$
$$f_{\{est_1, \ldots, est_n\}}(U) \to \langle f_{\{est_1\}}(U), \ldots, f_{\{est_n\}}(U) \rangle_n\ (n > 1)$$

The roles of $\widehat{st}, \mathcal{P}_{st}$ are made clear by the following lemma:

Lemma 5. *For any* $\mathcal{P}, st, e$: $st \in [\![e]\!]^{\mathcal{P}}_S \Leftrightarrow f_{st}(e) \to^*_{\mathcal{P}'} \widehat{st}$, *where* $\mathcal{P}' \equiv \mathcal{P} \cup \mathcal{P}_{st}$.

We can now proceed with the proof of the pending implication.
$\boxed{\Leftarrow}$ Assume that $\mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e]) \subseteq \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e'])$ for any safe extension $\mathcal{P}'$ and context $\mathcal{C}$. We prove $[\![e]\!]^{\mathcal{P}}_S \subseteq [\![e']\!]^{\mathcal{P}}_S$ (the other inclusion is similar). Let $st \in [\![e]\!]^{\mathcal{P}}_S$. Let $\mathcal{P}' \equiv \mathcal{P} \cup \mathcal{P}_{st}$, which is a safe extension of $(P, e), (P, e')$. Lemma 5 ensures $f_{st}(e) \to^*_{\mathcal{P}'} \widehat{st}$, which means $\widehat{st} \in \mathcal{O}_t^{\mathcal{P}'}(f_{st}(e))$. Now, since $\mathcal{P}'$ is a safe extension, observational equivalence of e, e' implies $\widehat{st} \in \mathcal{O}_t^{\mathcal{P}'}(f_{st}(e'))$, which means $f_{st}(e') \to^*_{\mathcal{P}'} \widehat{st}$. Again by Lemma 5, we conclude that $st \in [\![e']\!]^{\mathcal{P}}_S$, as desired.

5 Conclusions

In this paper we have provided a semantics for constructor based rewriting systems that is fully abstract with respect to natural notions of observation that extract the outer constructor part of outcomes as relevant information of computations. To the best of our knowledge, this is the first time that full abstraction has been achieved

for this class of programs and observations. Along the way to this result we have made some contributions: After noticing that 'obvious' semantics based directly on rewrite sequences lack compositionality, our main insight has been that it can be recovered by recursively packaging set of results below constructor symbols. That insight has been realized at the technical level by introducing s-cterms as suitable semantic values, and giving a proof calculus able to derive reachable s-cterms from a given expression. Previous to full abstraction, we have proved a bunch of good properties of the semantics: polarity, compositionality, closedness under substitutions, correctness and completeness with respect to rewriting.

We expect our semantics to be a useful tool for CS-based program manipulation. We remark that, for instance, to justify the correctness of a CS-transformation by proving preservation of reachable values could be incorrect if transformations are to be used locally. Our semantics could be a better option, and we plan to explore this.

There are other aspects not yet accomplished that can be subject of future work. In the paper, 'compositionality' refers to expressions wrt its subexpressions, and not to programs obtained by joining others, an interesting topic related to modularity (see e.g. [2]). We plan also to extend our approach to consider semantics and notions of observations that give a more active role to variables (as happens in [2, 1, 11]) taking into account that, for instance, in narrowing-based operational procedures, variables are subject of narrowing substitutions. Dropping the constructor restriction is also interesting, replacing the role of constructor values by appropriate alternatives. Finally, incorporating s-expressions to the syntax of programs, as discussed in Sect. 3.2, could lead to more expressive programs.

References

1. M. Alpuente, M. Comini, S. Escobar, M. Falaschi, and S. Lucas. Abstract diagnosis of functional programs. In *Proc. LOPSTR'02*, p. 1–16, Springer LNCS 2664, 2003.
2. M. Alpuente, M. Falaschi, M. Ramis, and G. Vidal. Narrowing approximations as an optimization for equational logic programs. In *Proc. PLILP'93*, p. 391–409. Springer LNCS 714, 1993.
3. S. Antoy. Optimal non-deterministic functional logic computations. In *Proc. ALP'97*, p. 16–30. Springer LNCS 1298, 1997.
4. S. Antoy, P. J. Iranzo, and B. Massey. Improving the efficiency of non-deterministic computations. *ENTCS, 64*, 2002.
5. G. Boudol. Une semantique pour les arbres non deterministes. In *Proc. CAAP'81*, p. 147–161, Springer LNCS 1981.
6. G. Boudol. Computational semantics of term rewriting systems. In *Algebraic methods in semantics*, p. 169–236, Camb. Univ. Press, 1986.
7. B. Braßel and F. Huch. On a tighter integration of functional and logic programming. In *Proc. APLAS*, p. 122–138, 2007.
8. M. Clavel et al. (eds). *All About Maude* , Springer LNCS 4350, 2007.
9. 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. of Logic Programming*, 40(1):47–87, 1999.
10. M. Hanus. Multi-paradigm declarative languages. In *Proc.ICLP 2007*, p. 45–75. Springer LNCS 4670, 2007.
11. M. Hanus and S. Lucas. An evaluation semantics for narrowing-based functional logic languages. *J. Funct. and Logic Prog.*, 2001(2), 2001.
12. H. Hussmann. *Non-Determinism in Algebraic Specifications and Algebraic Programs.* Birkhäuser Verlag, 1993.
13. F. López-Fraguas, J. Rodríguez-Hortalá, and J. Sánchez-Hernández. Bundles pack tighter than lists. In *Draft Proc. TFP'07*, 2007.
14. N. Martí-Oliet and J. Meseguer. Rewriting logic: roadmap and bibliography. *Theor. Comput. Sci.*, 285(2):121–154, 2002.
15. S.-O. Nyström. There is no fully abstract fixpoint semantics for non-deterministic languages with infinite computations. *Inf. Process. Lett.*, 60(6):289–293, 1996.
16. E. Ohlebusch. *Advanced topics in term rewriting.* Springer-Verlag, 2002.
17. G. D. Plotkin. LCF considered as a programming language. *Theor. Comput. Sci.*, 5(3):225–255, 1977.
18. J. Reynolds. *Theories of Programing Languages.* Camb. Univ. Press, 1998.

A Proofs of the results

During this section we will use the symbol $\equiv$ to refer to syntactic equality between two elements, that is, occurence of the same symbols in the same positions.

Lemma 6. *For any $st \in SCTerm$ we have that $st \twoheadrightarrow st$.*

Proof. We proceed by induction on the structure of st. If $st \equiv \emptyset$ then $st \equiv \emptyset \twoheadrightarrow \emptyset \equiv st$, by E. Otherwise $st \equiv \{est_1, \ldots, est_n\}$ and we have two possibilities:

a) $n = 1$: Then if $st \equiv \{X\}$ for some $X \in \mathcal{V}$ we are done as $\{X\} \twoheadrightarrow \{X\}$ by RR. Otherwise $st \equiv \{c(st_1, \ldots, st_m)\}$ and we can apply the IH to each of them to get $st_i \twoheadrightarrow st_i$ and prove $\{c(st_1, \ldots, st_m)\} \twoheadrightarrow \{c(st_1, \ldots, st_m)\}$ by DC.
b) $n > 1$: Then each est_i must have one of the following shapes:
   i) $est_i \equiv X$ for some $X \in \mathcal{V}$: Then $\{est_i\} \equiv \{X\} \twoheadrightarrow \{X\} \equiv \{est_i\}$ by RR.
   ii) $est_i \equiv c(st'_1, \ldots, st'_m)$ and we can apply the IH to each of them to get $st'_i \twoheadrightarrow st'_i$ and prove $\{est_i\} \equiv \{c(st'_1, \ldots, st'_m)\} \twoheadrightarrow \{c(st'_1, \ldots, st'_m)\} \equiv \{est_i\}$ by DC.

   Therefore we have proved $\{est_i\} \twoheadrightarrow \{est_i\}$ for each est_i, hence we can apply Less to get $st \equiv \{est_1, \ldots, est_n\} \twoheadrightarrow \{est_1\} \cup \ldots \cup \{est_n\} \equiv \{est_1, \ldots, est_n\} \equiv st$.

Lemma 7 (Basic properties of $\sqsubseteq$).

- *If* $\forall i \in \{1, \ldots, n\} \exists j \in \{1, \ldots, m\}$ *such that* $se_i \sqsubseteq se'_j$ *then* $se_1 \cup \ldots \cup se_n \sqsubseteq se'_1 \cup \ldots \cup se'_m$.
- *If* $se \sqsubseteq se'$ *and* $se' \subseteq se''$ *then* $se \sqsubseteq se''$.
- *If* $se \subseteq se'$ *and* $se' \sqsubseteq se''$ *then* $se \sqsubseteq se''$. *As a result* $se \subseteq se'$ *implies* $se \sqsubseteq se'$.

Proof. By definition of $\sqsubseteq$ and basic set reasoning.

Lemma 8. *Under any program* $\mathcal{P}$, $\forall st, st' \in SCTerm$ $st' \sqsubseteq st$ *iff* $st \twoheadrightarrow st'$.

Proof. Assume $st' \sqsubseteq st$, then by Lemma 6 we have $st \twoheadrightarrow st$, hence $st \twoheadrightarrow st'$ by Prop. 1.
Concerning the other implication, assume $st \twoheadrightarrow st'$, we proceed by induction on the structure of $st \twoheadrightarrow st'$. The base cases for E, RR and DC are trivial. In the inductive case for DC we have $st \equiv \{c(st_1, \ldots, st_n)\} \twoheadrightarrow \{c(st'_1, \ldots, st'_n)\} \equiv st'$, and we may apply the IH to each $st_i \twoheadrightarrow st'_i$ to get $st'_i \sqsubseteq st_i$, hence $st \sqsubseteq st'$ by definition of $\sqsubseteq$. In the case for More we have $st \twoheadrightarrow st_1 \cup \ldots \cup st_n \equiv st'$ and we can apply the IH to each $st \twoheadrightarrow st_i$ to get $st_i \sqsubseteq st$, hence $st' \equiv st_1 \cup \ldots \cup st_n \sqsubseteq st$ by Lemma 7. Finally, in the case for Less we have $st \twoheadrightarrow st_1 \cup \ldots \cup st_m \equiv st'$ using $\{esa_1, \ldots, esa_m\} \subseteq st$. Then we can apply the IH to each $\{esa_i\} \twoheadrightarrow st_i$ to get $st_i \sqsubseteq \{esa_i\}$, hence $st_1 \cup \ldots \cup st_m \sqsubseteq \{esa_1\} \cup \ldots \{esa_m\} \equiv \{esa_1, \ldots, esa_m\} \subseteq st$ by Lemma 7, and so $st' \equiv st_1 \cup \ldots \cup st_m \sqsubseteq st$ by Lemma 7 again.

A.1 For Subsection 3.1

Given a set se we use the notation $\sharp se$ to express its cardinality.

We use the depth of an expression that is the number of symbols of the signature contained in such expresion. Formally:

Definition 3 (Depth of expresions). *The depth of an expression* e *is defined as:*

- $depth(\bot) = depth(X) = depth(h) = 0$, *for* $X \in \mathcal{V}$, $h \in \Sigma^0$
- $depth(h(e_1, \ldots, e_n)) = 1 + \sum_{i=1}^{n} depth(e_i)$, *for* $h \in \Sigma^n$

Proposition 8. *For any* $e \in Exp$ *it holds* $\tilde{e} \in SExp$ *and* $var(\tilde{e}) = var(e)$.

Proof. These properties are quit clear by construction. A formal proof can be done by induction on the depth of the expression e. The base cases are $\bot$, $X \in \mathcal{V}$ and $h \in \Sigma^0$ for which both properties clearly hold. For the case $e = h(e_1, \ldots, e_n)$ with $n > 0$ we have $\tilde{e} = \{h(\tilde{e}_1, \ldots, \tilde{e}_n)\}$ and we can apply IH to each e_i.

Proposition 9. *For any* $e \in Exp$ *it holds* $flat(\tilde{e}) = \{e\}$.

Proof. Again we proceed by induction on the depth of the expression e. There are three base cases and we only have to apply sequentially the definitions of $\tilde{}$ and $flat$:

- $e = \bot$: we have $flat(\tilde{\bot}) = flat(\emptyset) = \{\bot\}$.
- $e = X \in \mathcal{V}$: we have $flat(\tilde{X}) = flat(\{X\}) = \{X\}$.
- $e = h \in \Sigma^0$: we have $flat(\tilde{h}) = flat(\{h\}) = \{h\}$.

For the a depth greater than 0 consider $e = h(e_1, \ldots, e_n)$. We have the following chain:

$$\begin{array}{lll} flat(\widetilde{h(e_1, \ldots, e_n)}) = flat(\{h(\tilde{e}_1, \ldots, \tilde{e}_n)\}) & & \text{(def. of } \tilde{\ }) \\ = flat(h(\tilde{e}_i, \ldots, \tilde{e}_n)) & & \text{(def. of } flat \text{ for } eSExp) \\ = \{h(e'_1, \ldots, e'_n) \mid e'_i \in flat(\tilde{e}_i)\} & & \text{(def. of } flat \text{ for } SExp) \\ = \{h(e'_1, \ldots, e'_n) \mid e'_i \in \{e_i\}\} & & \text{(IH)} \\ = \{h(e_1, \ldots, e_n)\} & & \text{sets manipulation} \end{array}$$

Proposition 10. *Given $\sigma \in SSubst$, $se \in SExp$ and $ese \in eSExp$ we have both $se\sigma \in SExp$ and $ese\sigma \in SExp$.*

Proof. It is a direct application of the definition of application of substitutions.

Lemma 9. *$\forall e \in Exp_\perp, \sigma \in Subst_\perp, \mathcal{C} \in Cntxt$:*

i) $\widetilde{e\sigma} \equiv \tilde{e}\tilde{\sigma}$.
ii) $\widetilde{\mathcal{C}[e]} \equiv \tilde{\mathcal{C}}[\tilde{e}]$.

Proof.

i) We proceed by induction on the structure of e. Concerning the base cases:
   - $e \equiv \perp$: Then $\widetilde{e\sigma} \equiv \widetilde{\perp\sigma} \equiv \tilde{\perp} \equiv \emptyset \equiv \emptyset\tilde{\sigma} \equiv \tilde{\perp}\tilde{\sigma} \equiv \tilde{e}\tilde{\sigma}$
   - $e \equiv X \in \mathcal{V}$: Then $\widetilde{e\sigma} \equiv \widetilde{X\sigma} \equiv \widetilde{\sigma(X)} \equiv \tilde{\sigma}(X) \equiv X\tilde{\sigma} \equiv \{X\}\tilde{\sigma} \equiv \tilde{X}\tilde{\sigma} \equiv \tilde{e}\tilde{\sigma}$
   - $e \equiv h$: Then $\widetilde{e\sigma} \equiv \widetilde{h\sigma} \equiv \tilde{h} \equiv \{h\} \equiv \{h\}\tilde{\sigma} \equiv \tilde{e}\tilde{\sigma}$

   Concening the inductive step, this happens when $e \equiv h(e_1, \ldots, e_n)$, then

$$\begin{array}{l} \widetilde{e\sigma} \equiv \widetilde{h(e_1, \ldots, e_n)\sigma} \equiv \widetilde{h(e_1\sigma, \ldots, e_n\sigma)} \equiv \{h(\widetilde{e_1\sigma}, \ldots, \widetilde{e_n\sigma})\} \\ \equiv_{IH} \{h(\tilde{e_1}\tilde{\sigma}, \ldots, \tilde{e_n}\tilde{\sigma})\} \equiv \{h(\tilde{e_1}, \ldots, \tilde{e_n})\}\tilde{\sigma} \equiv \widetilde{h(e_1, \ldots, e_n)}\tilde{\sigma} \equiv \tilde{e}\tilde{\sigma} \end{array}$$

ii) We proceed by induction on the structure of $\mathcal{C}$. The base case happens when $\mathcal{C} \equiv []$, then $\widetilde{\mathcal{C}[e]} \equiv \widetilde{[e]} \equiv \tilde{e} \equiv []\tilde{[e]} \equiv \tilde{[]}\tilde{[e]} \equiv \tilde{\mathcal{C}}[\tilde{e}]$. Concerning the inductive step, this happens when $\mathcal{C} \equiv h(e_1, \ldots, \mathcal{C}', \ldots, e_n)$, then

$$\begin{array}{ll} \widetilde{\mathcal{C}[e]} \equiv h(e_1, \ldots, \widetilde{\mathcal{C}'[e]}, \ldots, e_n) \equiv \{h(\widetilde{e_1}, \ldots, \widetilde{\mathcal{C}'[e]}, \ldots, \widetilde{e_n})\} & \\ \equiv \{h(\widetilde{e_1}, \ldots, \tilde{\mathcal{C}'}[\tilde{e}], \ldots, \widetilde{e_n})\} & \text{by IH} \\ \equiv \{h(\widetilde{e_1}, \ldots, \tilde{\mathcal{C}'}, \ldots, \widetilde{e_n})\}[\tilde{e}] \equiv \tilde{\mathcal{C}}[\tilde{e}] & \end{array}$$

A.2 For Subsection 3.2

Proof (For Proposition 1 (Polarity)). We proceed by induction on the structure of $se \rightarrowtriangle st$. Concerning the base cases:

E Then we have $se \rightarrowtriangle \emptyset \equiv st$, and so $st' \sqsubseteq st$ implies $st' \equiv \emptyset$ too, therefore $se \rightarrowtriangle \emptyset \equiv st'$ by E again.
RR Then we have $se \equiv \{X\} \rightarrowtriangle \{X\} \equiv st$, and so $st' \sqsubseteq st$ implies $st' \equiv \emptyset$ or $st' \equiv \{X\}$. Besides $se \equiv \{X\} \sqsubseteq se'$ implies $X \in se'$ and so $\sharp se' \geq 1$. Let us consider the different possibilities:
   a) $st' \equiv \emptyset$: Then $se \rightarrowtriangle \emptyset \equiv st'$ by E.
   b) $st' \equiv \{X\}$:
      i) $\sharp se' = 1$: Then $X \in se'$ implies $se' \equiv \{X\}$ and so $se' \equiv \{X\} \rightarrowtriangle \{X\} \equiv st'$ by RR.
      ii) $\sharp se' > 1$: Then we can use $X \in se'$ to do

$$\dfrac{\dfrac{}{\{X\} \rightarrowtriangle \{X\}}\ \text{RR}}{se' \rightarrowtriangle \{X\} \equiv st'}\ \textsc{Less}$$

DC Similar to the previous case just changing X for c and RR for DC.

Concerning the inductive steps:

DC If $st' \equiv \emptyset$ then we proceed like in the case for E. Otherwise, we have

$$\frac{se_1 \twoheadrightarrow st_1 \quad \dots \quad se_n \twoheadrightarrow st_n}{se \equiv \{c(se_1, \dots, se_n)\} \twoheadrightarrow \{c(st_1, \dots, st_n)\} \equiv st} \text{ DC}$$

Then $se \sqsubseteq se'$ implies that $\exists c(se'_1, \dots, se'_n) \in se'$ such that $\forall i \in \{1, \dots, n\}, se_i \sqsubseteq se'_i$, and so $\sharp se' \geq 1$. Besides $st' \sqsubseteq st$ implies $\forall est'_j \in st', est'_j \equiv c(st'_{1j}, \dots, st'_{nj})$ such that $\forall i \in \{1, \dots, n\}, st'_{ij} \sqsubseteq st_i$. But then, if $st' \equiv \{est'_1, \dots, est'_m\}$, we can apply the IH $\forall j \in \{1, \dots, m\}, \forall i \in \{1, \dots, n\}$ over $se_i \twoheadrightarrow st_i$ with $se_i \sqsubseteq se'_i$ and $st'_{ij} \sqsubseteq st_i$ to get $se'_i \twoheadrightarrow st'_{ij}$. And we have the following possibilities:

a) $\sharp se' = 1$: Then $c(se'_1, \dots, se'_n) \in se'$ implies $se' \equiv \{c(se'_1, \dots, se'_n)\}$.

i) $\sharp st' = m = 1$: Then we can do

$$\frac{\dfrac{IH}{se'_1 \twoheadrightarrow st'_{11}} \quad \dots \quad \dfrac{IH}{se'_n \twoheadrightarrow st'_{n1}}}{se' \equiv \{c(se'_1, \dots, se'_n)\} \twoheadrightarrow \{c(st'_{11}, \dots, st'_{n1})\} \equiv st'} \text{ DC}$$

ii) $\sharp st' = m > 1$: Then we can do

$$\frac{\dfrac{\dfrac{IH}{se'_1 \twoheadrightarrow st'_{11}} \quad \dots \quad \dfrac{IH}{se'_n \twoheadrightarrow st'_{n1}}}{\{c(se'_1, \dots, se'_n)\} \twoheadrightarrow \{c(st'_{11}, \dots, st'_{n1})\}} \text{ DC} \quad \dots \quad \dfrac{\dfrac{IH}{se'_1 \twoheadrightarrow st'_{1m}} \quad \dots \quad \dfrac{IH}{se'_n \twoheadrightarrow st'_{nm}}}{\{c(se'_1, \dots, se'_n)\} \twoheadrightarrow \{c(st'_{1m}, \dots, st'_{nm})\}} \text{ DC}}{se' \equiv \{c(se'_1, \dots, se'_n)\} \twoheadrightarrow \{c(st'_{11}, \dots, st'_{n1})\} \cup \dots \cup \{c(st'_{1m}, \dots, st'_{nm})\} \equiv st'} \text{ More}$$

b) $\sharp se' > 1$: Then we can prove $\{c(se'_1, \dots, se'_n)\} \twoheadrightarrow st'$ like in case *a)*, so as $c(se'_1, \dots, se'_n) \in se'$ we can apply Less to get $se' \twoheadrightarrow st'$.

More If $st' \equiv \emptyset$ then we proceed like in the case for E. Otherwise $st' \equiv \{est'_1, \dots, est'_m\}$ with $m > 0$ and we have

$$\frac{se \twoheadrightarrow st_1 \quad \dots \quad se \twoheadrightarrow st_n}{se \twoheadrightarrow st_1 \cup \dots \cup st_n \equiv st} \text{ More}$$

Then $st' \sqsubseteq st$ implies $\forall est'_j \in st', \exists est_j \in st \equiv st_1 \cup \dots \cup st_n$ such that $est'_j \sqsubseteq est_j$, therefore $\exists i \in \{1, \dots, n\}$ such that $est'_j \sqsubseteq est_j \in st_i$, hence $\{est'_j\} \sqsubseteq st_i$. But then we can apply the IH over $se \twoheadrightarrow st_i$ with $\{est'_j\} \sqsubseteq st_i$ and $se \sqsubseteq se'$ to get that $se' \twoheadrightarrow \{est'_j\}$. We can do it for every $est'_j \in st'$, and build the following proof

$$\frac{\dfrac{IH}{se' \twoheadrightarrow \{est'_1\}} \quad \dots \quad \dfrac{IH}{se' \twoheadrightarrow \{est'_m\}}}{se' \twoheadrightarrow \{est'_1\} \cup \dots \cup \{est'_m\} \equiv st'} \text{ More}$$

Less If $st' \equiv \emptyset$ then we proceed like in the case for E. Otherwise $st' \equiv \{est'_1, \dots, est'_m\}$ with $m > 0$ and we have

$$\frac{\{esa_1\} \twoheadrightarrow st_1 \quad \dots \quad \{esa_n\} \twoheadrightarrow st_n}{se \twoheadrightarrow st_1 \cup \dots \cup st_n \equiv st} \text{ Less}$$

Then we can reason like in the case for More to get that $st' \sqsubseteq st$ implies that $\forall est'_j \in st', \exists i \in \{1, \dots, n\}$ such that $\{est'_j\} \sqsubseteq st_i$. Besides by Lemma 7 we have that $\forall i \in \{1, \dots, n\}, \{esa_i\} \subseteq \{esa_1, \dots, esa_n\} \subseteq se \sqsubseteq se'$ implies $\{esa_i\} \sqsubseteq se'$, and so we can apply the IH to each $\{esa_i\} \twoheadrightarrow st_i$ with $\{est'_j\} \sqsubseteq st_i$ and $\{esa_i\} \sqsubseteq se'$ to get $se' \twoheadrightarrow \{est'_j\}$, and build the following proof

$$\frac{\dfrac{IH}{se' \twoheadrightarrow \{est'_1\}} \quad \dots \quad \dfrac{IH}{se' \twoheadrightarrow \{est'_m\}}}{se' \twoheadrightarrow \{est'_1\} \cup \dots \cup \{est'_m\} \equiv st'} \text{ More}$$

ROR If $st' \equiv \emptyset$ then we proceed like in the case for E. Otherwise $st' \equiv \{est'_1, \dots, est'_m\}$ with $m > 0$ and we have

$$\frac{se_1 \twoheadrightarrow \tilde{p_1}\theta \quad \dots \quad se_n \twoheadrightarrow \tilde{p_n}\theta \quad \tilde{r}\theta \twoheadrightarrow st}{se \equiv \{f(se_1, \dots, se_n)\} \twoheadrightarrow st} \text{ ROR}$$

Then $se \sqsubseteq se'$ implies that $\exists f(se'_1, \dots, se'_n) \in se'$ such that $\forall i \in \{1, \dots, n\}, se_i \sqsubseteq se'_i$, and so $\sharp se' \geq 1$. But then by IH over each $se_i \twoheadrightarrow \tilde{p_i}\theta$ with $\tilde{p_i}\theta \sqsubseteq \tilde{p_i}\theta$ and $se_i \sqsubseteq se'_i$ we get $se'_i \twoheadrightarrow \tilde{p_i}\theta$. We can also apply the IH to $\tilde{r}\theta \twoheadrightarrow st$ with $st' \sqsubseteq st$ and $\tilde{r}\theta \sqsubseteq \tilde{r}\theta$, to get $\tilde{r}\theta \twoheadrightarrow st'$. Then we have the following possibilities:

a) $\sharp se' = 1$: Then $f(se'_1, \ldots, se'_n) \in se'$ implies $se' \equiv \{f(se'_1, \ldots, se'_n)\}$, and we can do

$$\frac{\dfrac{IH}{se'_1 \twoheadrightarrow \tilde{p_1}\theta} \quad \ldots \quad \dfrac{IH}{se'_n \twoheadrightarrow \tilde{p_n}\theta} \quad \dfrac{IH}{\tilde{r}\theta \twoheadrightarrow st'}}{se' \equiv \{f(se'_1, \ldots, se'_n)\} \twoheadrightarrow st'} \text{ ROR}$$

b) $\sharp se' > 1$: Then we can prove $\{f(se'_1, \ldots, se'_n)\} \twoheadrightarrow st'$ like in case *a)*, so as $f(se'_1, \ldots, se'_n) \in se'$ we can apply LESS to get $se' \twoheadrightarrow st'$.

Lemma 10. *Under any program and* $\forall se \in SExp, ese \in ESExp, \sigma \in SSubst$*,if* $se\sigma \equiv \{ese\}$ *then* $\forall ese' \in se, ese'\sigma \equiv \{ese\} \equiv se\sigma$*.*

Proof. $\{ese\} \equiv \bigcup_{ese' \in se} ese'\sigma$, hence $\forall ese' \in se, ese'\sigma \equiv \{ese\} \equiv se\sigma$.

Lemma 11. *Under any program and* $\forall se \in SExp, st \in SCTerm, \sigma \in SSubst$*, if* $\exists ese \in se$ *such that* $ese\sigma \twoheadrightarrow st$ *then* $se\sigma \twoheadrightarrow st$*. In other words,* $\forall ese \in se, [\![ese\sigma]\!] \subseteq [\![se\sigma]\!]$*.*

Proof. If $ese \in se$ then $ese\sigma \subseteq \bigcup_{esa \in se} esa\sigma \equiv se\sigma$, hence $ese\sigma \sqsubseteq se\sigma$ by Lemma 7, and $[\![ese\sigma]\!] \subseteq [\![se\sigma]\!]$ by Prop. 1.

Proposition 11. *The relation* $\trianglelefteq$ *is a preorder but not a partial order under any CS.*

Proof. It is reflexive because obviously $\forall \sigma \in SSubst, \forall X \in \mathcal{V}, [\![\sigma(X)]\!] = [\![\sigma(X)]\!]$, it is transitive as a consecuence of the transitivity of $\subseteq$, but it is not antisymmetric because for example under $\{amb(X,Y) \to X, amb(X,Y) \to Y\}$ we have $[X/amb(a,b)] \trianglelefteq [X/amb(a, amb(a,b))]$ and $[X/amb(a, amb(a,b))] \trianglelefteq [X/amb(a,b)]$ but $[X/amb(a,b)] \neq [X/amb(a, amb(a,b))]$.

Proof (For Proposition 2 (Monotonicity of substitutions)). First of all, if $\sigma \sqsubseteq \sigma'$ then $\sigma \trianglelefteq \sigma'$, because if $\sigma \sqsubseteq \sigma'$ then $\forall X \in \mathcal{V}$ if $\sigma(X) \twoheadrightarrow st$ then as $\sigma(X) \sqsubseteq \sigma'(X)$ then we can apply Prop. 1 to get $\sigma'(X) \twoheadrightarrow st$.

All that is left is proving that $\sigma \trianglelefteq \sigma'$ implies that forall $st \in SCTerm$ such that $se\sigma \twoheadrightarrow st$ then $se\sigma' \twoheadrightarrow st$. We proceed by induction on the structure of $se\sigma \twoheadrightarrow st$, the base cases are the following:

E Then $st \equiv \emptyset$, hence $se\sigma' \twoheadrightarrow \emptyset \equiv st$ by E.

RR Then $se\sigma \equiv \{X\} \twoheadrightarrow \{X\} \equiv st$, and so by Lemma 10 $\forall ese \in se, ese\sigma \equiv \{X\}$, hence $\forall ese \in se, ese \in \mathcal{V}$. Besides $se\sigma \equiv \{X\}$ implies $se \not\equiv \emptyset$, therefore $\exists ese \in se, ese \equiv Y \in \mathcal{V} \wedge Y\sigma \equiv ese\sigma \equiv \{X\}$, and so $\sigma(Y) \equiv Y\sigma \equiv \{X\} \twoheadrightarrow st$ by hypothesis. But then $\sigma \trianglelefteq \sigma'$ implies $st \in [\![\sigma(Y)]\!] \subseteq [\![\sigma'(Y)]\!] \subseteq [\![se\sigma']\!]$ by Lemma 11.

DC Then $se\sigma \equiv \{c\} \twoheadrightarrow \{c\} \equiv st$, and so by Lemma 10 $\forall ese \in se, ese\sigma \equiv \{c\}$. We have two possibilities:

a) $se \cap \mathcal{V} \neq \emptyset$: Then given some $Y \in se \cap \mathcal{V}$ we have $\sigma(Y) \equiv Y\sigma \equiv \{c\} \twoheadrightarrow st$ by hyphotesis. But then $\sigma \trianglelefteq \sigma'$ implies $st \in [\![\sigma(Y)]\!] \subseteq [\![\sigma'(Y)]\!] \subseteq [\![se\sigma']\!]$ by Lemma 11.

b) $se \cap \mathcal{V} = \emptyset$: Then $\forall ese \in se, ese\sigma \equiv \{c\}$ implies $\forall ese \in se, ese \equiv \{c\}$. Besides $se\sigma \equiv \{c\}$ implies $se \not\equiv \emptyset$, therefore $\exists ese \in se, ese\sigma' \equiv \{c\}\sigma' \equiv \{c\} \twoheadrightarrow st$ by hypothesis , but then $st \in [\![ese\sigma']\!] \subseteq [\![se\sigma']\!]$ by Lemma 11.

Concerning the inductive steps:

DC Then we have

$$\frac{se_1 \twoheadrightarrow st_1 \quad \ldots \quad se_n \twoheadrightarrow st_n}{se\sigma \equiv \{c(se_1, \ldots, se_n)\} \twoheadrightarrow \{c(st_1, \ldots, st_n)\} \equiv st} \text{ DC}$$

and so by Lemma 10 $\forall ese \in se, ese\sigma \equiv \{c(se_1, \ldots, se_n)\}$. We have two possibilities:

a) $se \cap \mathcal{V} \neq \emptyset$: Then given some $Y \in se \cap \mathcal{V}$ we have $\sigma(Y) \equiv Y\sigma \equiv \{c(se_1, \ldots, se_n)\} \twoheadrightarrow st$ by hypothesis. But then $\sigma \trianglelefteq \sigma'$ implies $st \in [\![\sigma(Y)]\!] \subseteq [\![\sigma'(Y)]\!] \subseteq [\![se\sigma']\!]$ by Lemma 11.

b) $se \cap \mathcal{V} = \emptyset$: Then $se\sigma \equiv \{c(se_1, \ldots, se_n)\}$ implies $se \not\equiv \emptyset$. Therefore $\exists ese \in se = (se \setminus \mathcal{V})$ such that $ese\sigma \equiv \{c(se_1, \ldots, se_n)\}$, which implies $ese \equiv c(se'_1, \ldots, se'_n)$ such that $\forall i, se'_i\sigma \equiv se_i \twoheadrightarrow st_i$, to which we can apply the IH to get $se'_i\sigma' \twoheadrightarrow st_i$, and build the following proof:

$$\frac{se'_1\sigma' \twoheadrightarrow st_1 \quad \ldots \quad se'_n\sigma' \twoheadrightarrow st_n}{ese\sigma' \equiv \{c(se'_1\sigma', \ldots, se'_n\sigma')\} \twoheadrightarrow \{c(st_1, \ldots, st_n)\} \equiv st} \text{ DC}$$

But then $st \in [\![ese\sigma']\!] \subseteq [\![se\sigma']\!]$ by Lemma 11.

More Then we have

$$\frac{se\sigma \twoheadrightarrow st_1 \quad \dots \quad se\sigma \twoheadrightarrow st_n}{se\sigma \twoheadrightarrow st_1 \cup \dots \cup st_n \equiv st} \text{ More}$$

But then we can apply the IH to each $se\sigma \twoheadrightarrow st_i$ to get $se\sigma' \twoheadrightarrow st_i$, and build the following proof:

$$\frac{se\sigma' \twoheadrightarrow st_1 \quad \dots \quad se\sigma' \twoheadrightarrow st_n}{se\sigma' \twoheadrightarrow st_1 \cup \dots \cup st_n \equiv st} \text{ More}$$

Less Then we have

$$\frac{\{esa_1\} \twoheadrightarrow st_1 \quad \dots \quad \{esa_n\} \twoheadrightarrow st_m}{se\sigma \twoheadrightarrow st_1 \cup \dots \cup st_m \equiv st} \text{ Less}$$

for some $\{esa_1, \dots, esa_m\} \subseteq se\sigma$ and $\sharp se\sigma \geq 2$. Then for any $esa_i \in \{esa_1, \dots, esa_m\} \subseteq se\sigma$ we have two possibilities:

a) $esa_i \in \sigma(X)$ for some $X \in se$: But then

$$\begin{array}{ll} st_i \in [\![\{esa_i\}]\!] \subseteq [\![\sigma(X)]\!] & \text{by Lemma 7 and Prop. 1, as } \{esa_i\} \subseteq \sigma(X) \\ \subseteq [\![\sigma'(X)]\!] = [\![X\sigma']\!] & \text{as } \sigma \trianglelefteq \sigma' \\ \subseteq [\![se\sigma']\!] & \text{by Lemma 11, as } X \in se \end{array}$$

b) $esa_i \equiv h(se_1\sigma, \dots, se_n\sigma)$ for some $h(se_1, \dots, se_n) \in se$. But then $\{h(se_1, \dots, se_n)\}\sigma \equiv \{h(se_1\sigma, \dots, se_n\sigma)\} \equiv \{esa_i\} \twoheadrightarrow st_i$ by hypothesis, and we can apply the IH to get $\{h(se_1, \dots, se_n)\}\sigma' \twoheadrightarrow st_i$. But then $h(se_1, \dots, se_n) \in se$ implies $st_i \in [\![\{h(se_1, \dots, se_n)\}\sigma']\!] = [\![h(se_1, \dots, se_n)\sigma']\!] \subseteq [\![se\sigma']\!]$ by Lemma 11.

Therefore $\forall i \in \{1, \dots, m\}, se\sigma' \twoheadrightarrow st_i$, and we can build the following proof:

$$\frac{se\sigma' \twoheadrightarrow st_1 \quad \dots \quad se\sigma' \twoheadrightarrow st_m}{se\sigma' \twoheadrightarrow st_1 \cup \dots \cup st_m \equiv st} \text{ More}$$

ROR Then we have

$$\frac{se_1 \twoheadrightarrow \tilde{p_1}\theta \quad \dots \quad se_n \twoheadrightarrow \tilde{p_n}\theta \quad \tilde{r}\theta \twoheadrightarrow st}{se\sigma \equiv \{f(se_1, \dots, se_n)\} \twoheadrightarrow st} \text{ ROR}$$

and so by Lemma 10 $\forall ese \in se, ese\sigma \equiv \{f(se_1, \dots, se_n)\}$. We have two possibilities:

a) $se \cap \mathcal{V} \neq \emptyset$: Then given some $Y \in se \cap \mathcal{V}$ we have $\sigma(Y) \equiv Y\sigma \equiv \{f(se_1, \dots, se_n)\} \twoheadrightarrow st$ by hypothesis. But then $\sigma \trianglelefteq \sigma'$ implies $st \in [\![\sigma(Y)]\!] \subseteq [\![\sigma'(Y)]\!] \subseteq [\![se\sigma']\!]$ by Lemma 11.

b) $se \cap \mathcal{V} = \emptyset$: Then $se\sigma \equiv \{f(se_1, \dots, se_n)\}$ implies $se \not\equiv \emptyset$. Therefore $\exists ese \in se = (se \setminus \mathcal{V})$ such that $ese\sigma \equiv \{f(se_1, \dots, se_n)\}$, which implies $ese \equiv f(se'_1, \dots, se'_n)$ such that $\forall i, se'_i\sigma \equiv se_i \twoheadrightarrow \tilde{p_i}\theta$, to which we can apply the IH to get $se'_i\sigma' \twoheadrightarrow \tilde{p_i}\theta$, and build the following proof:

$$\frac{se'_1\sigma' \twoheadrightarrow \tilde{p_1}\theta \quad \dots \quad se'_n\sigma' \twoheadrightarrow \tilde{p_n}\theta \quad \dfrac{hypothesis}{\tilde{r}\theta \twoheadrightarrow st}}{ese\sigma' \equiv \{f(se'_1\sigma', \dots, se'_n\sigma')\} \twoheadrightarrow st} \text{ ROR}$$

But then $st \in [\![ese\sigma']\!] \subseteq [\![se\sigma']\!]$ by Lemma 11.

Lemma 12. *For any $\sigma \in SSubst, \theta \in [\![\sigma]\!]$ we have $\theta \trianglelefteq \sigma$.*

Proof. It is enough to check that $\forall X \in \mathcal{V}$, $[\![\theta(X)]\!] \subseteq [\![\sigma(X)]\!]$. That happens because given any $st \in SCTerm$ such that $\theta(X) \twoheadrightarrow st$, as $\theta \in [\![\sigma]\!]$ implies $\theta \in SCSubst$ which implies $\theta(X) \in SCTerm$, then $st \sqsubseteq \theta(X)$ by Lemma 8. But $\theta \in [\![\sigma]\!]$ implies $\sigma(X) \twoheadrightarrow \theta(X)$, hence we can apply Prop. 1 to get $\sigma(X) \twoheadrightarrow st$.

The following lemma will be used to prove compositionality.

Lemma 13. *$\forall se_1, se_2 \in SExp$ such that $se_1 \subseteq se_2$ we have that $\forall s\mathcal{C} \in sCntxt, [\![s\mathcal{C}[se_1]]\!] \subseteq [\![s\mathcal{C}[se_2]]\!]$.*

Proof. If $se_1 = se_2$ then $s\mathcal{C}[se_1] \equiv s\mathcal{C}[se_2]$ and so the result trivially holds.
On the other hand, for the case when $se_1 \subset se_2$, we will see that for any $s\mathcal{C} \in sCntxt$ and any $st \in SCTerm$ such that $s\mathcal{C}[se_1] \twoheadrightarrow st$ then $s\mathcal{C}[se_2] \twoheadrightarrow st$, by induction on the size K of $s\mathcal{C}[se_1] \twoheadrightarrow st$.

Base cases $K = 1$: Let us see which was the rule applied at the root of the proof for $s\mathcal{C}[se_1] \twoheadrightarrow st$.

E : Then $st \equiv \emptyset$ but then $s\mathcal{C}[se_2] \twoheadrightarrow \emptyset \equiv st$ by E.

RR : Then $s\mathcal{C}[se_1] \equiv \{X\}$ and so $s\mathcal{C} = []$ and $s\mathcal{C}[se_1] \equiv se_1 \equiv \{X\}$. Hence $\sharp se_1 = 1$, which combined with $se_1 \subseteq se_2$ implies $\sharp se_1 \geq 2$, and so

$$\dfrac{\dfrac{}{\{X\} \equiv se_1 \equiv s\mathcal{C}[se_1] \twoheadrightarrow st}\ hypothesis}{s\mathcal{C}[se_2] \equiv se_2 \twoheadrightarrow \{X\} \equiv st}\ \textsc{Less}$$

as $\{X\} \equiv se_1 \subset se_2$.

DC : Then $s\mathcal{C}[se_1] \equiv \{c\}$ and so $s\mathcal{C} = []$, and the hypothesis was

$$\dfrac{}{s\mathcal{C}[se_1] \equiv se_1 \equiv \{c\} \twoheadrightarrow \{c\} \equiv st}\ \text{DC}$$

Hence $\sharp se_1 = 1$, which combined with $se_1 \subseteq se_2$ implies $\sharp se_1 \geq 2$, and so

$$\dfrac{\dfrac{}{\{c\} \equiv se_1 \equiv s\mathcal{C}[se_1] \twoheadrightarrow st}\ hypothesis}{s\mathcal{C}[se_2] \equiv se_2 \twoheadrightarrow \{c\} \equiv st}\ \textsc{Less}$$

as $\{X\} \equiv se_1 \subset se_2$.

Inductive step $K > 1$: Let us see which was the rule applied at the root of the proof for $s\mathcal{C}[se_1] \twoheadrightarrow st$.

DC If $s\mathcal{C} = []$ then $s\mathcal{C}[se_1] \equiv se_1 \equiv \{c(se'_1, \ldots, se'_n)\}$ and so $\sharp se_1 = 1$, which combined with $se_1 \subseteq se_2$ implies $\sharp se_1 \geq 2$, and so

$$\dfrac{\dfrac{}{\{c(se'_1, \ldots, se'_n)\} \equiv se_1 \equiv s\mathcal{C}[se_1] \twoheadrightarrow st}\ hypothesis}{s\mathcal{C}[se_2] \equiv se_2 \twoheadrightarrow st}\ \textsc{Less}$$

as $\{c(se'_1, \ldots, se'_n)\} \equiv se_1 \subseteq se_2$. Otherwise if $s\mathcal{C} \neq []$ then the hypothesis was

$$\dfrac{se'_1 \twoheadrightarrow st'_1 \ \ldots\ s\mathcal{C}'[se_1] \twoheadrightarrow st' \ \ldots\ se'_n \twoheadrightarrow st'_n}{s\mathcal{C}[se_1] \equiv \{c(se'_1, \ldots, s\mathcal{C}'[se_1], \ldots, se'_n)\} \twoheadrightarrow c(st'_1, \ldots, st', \ldots, st'_n)\}}\ \text{DC}$$

but then we can apply the IH to $s\mathcal{C}'[se_1] \twoheadrightarrow st'$ to get $s\mathcal{C}'[se_2] \twoheadrightarrow st'$, so we can build

$$\dfrac{\dfrac{}{se'_1 \twoheadrightarrow st'_1}\ hypothesis \quad \ldots \quad \dfrac{}{s\mathcal{C}'[se_2] \twoheadrightarrow st'}\ IH \quad \ldots \quad \dfrac{}{se'_n \twoheadrightarrow st'_n}\ hypothesis}{s\mathcal{C}[se_2] \equiv \{c(se'_1, \ldots, s\mathcal{C}'[se_2], \ldots, se'_n)\} \twoheadrightarrow c(st'_1, \ldots, st', \ldots, st'_n)\}}\ \text{DC}$$

More Then we have

$$\dfrac{s\mathcal{C}[se_1] \twoheadrightarrow st_1 \ \ldots\ s\mathcal{C}[se_1] \twoheadrightarrow st_n}{s\mathcal{C}[se_1] \twoheadrightarrow st_1 \ \cup\ \ldots\ \cup\ st_n}\ \textsc{More}$$

but then we can apply the IH to each $s\mathcal{C}[se_1] \twoheadrightarrow st_i$ to get $s\mathcal{C}[se_2] \twoheadrightarrow st_i$ and build

$$\dfrac{\dfrac{}{s\mathcal{C}[se_2] \twoheadrightarrow st_1}\ IH \quad \ldots \quad \dfrac{}{s\mathcal{C}[se_2] \twoheadrightarrow st_n}\ IH}{s\mathcal{C}[se_2] \twoheadrightarrow st_1 \ \cup\ \ldots\ \cup\ st_n}\ \textsc{More}$$

Less If $s\mathcal{C} = []$ then the hypothesis was

$$\dfrac{\{esa_1\} \twoheadrightarrow st_1 \ \ldots\ \{esa_m\} \twoheadrightarrow st_m}{s\mathcal{C}[se_1] \equiv se_1 \twoheadrightarrow st_1 \ \cup\ \ldots\ \cup\ st_m}\ \textsc{Less}$$

with $\{esa_1, \ldots, esa_m\} \subseteq se_1 \subset se_2$, but then

$$\dfrac{\dfrac{}{\{esa_1\} \twoheadrightarrow st_1}\ hypothesis \quad \ldots \quad \dfrac{}{\{esa_m\} \twoheadrightarrow st_m}\ hypothesis}{s\mathcal{C}[se_2] \equiv se_2 \twoheadrightarrow st_1 \ \cup\ \ldots\ \cup\ st_m}\ \textsc{Less}$$

Otherwise if $s\mathcal{C} \neq []$ then the hypothesis was

$$\frac{\{esa_1\} \twoheadrightarrow st_1 \ \ldots \ \{esa_m\} \twoheadrightarrow st_m}{s\mathcal{C}[se_1] \equiv \{ese_1, \ldots, h(se'_1, \ldots, s\mathcal{C}'[se_1], \ldots, se'_k), \ldots, ese_n\} \twoheadrightarrow st_1 \ \cup \ \ldots \ \cup \ st_m} \text{ LESS}$$

Now we have two possibilities. If $h(se'_1, \ldots, s\mathcal{C}'[se_1], \ldots, se'_k) \notin \{esa_1, \ldots, esa_m\}$, then we can do

$$\frac{\dfrac{hypothesis}{\{esa_1\} \twoheadrightarrow st_1} \ \ldots \ \dfrac{hypothesis}{\{esa_m\} \twoheadrightarrow st_m}}{s\mathcal{C}[se_2] \equiv \{ese_1, \ldots, h(se'_1, \ldots, s\mathcal{C}'[se_2], \ldots, se'_k), \ldots, ese_n\} \twoheadrightarrow st_1 \ \cup \ \ldots \ \cup \ st_m} \text{ LESS}$$

because then neither $h(se'_1, \ldots, s\mathcal{C}'[se_2], \ldots, se'_k)$ nor $s\mathcal{C}'[se_2]$ in general are involved in the premises. Otherwise if $h(se'_1, \ldots, s\mathcal{C}'[se_1], \ldots, se'_k) \in \{esa_1, \ldots, esa_m\}$ then consider the corresponding premise $h(se'_1, \ldots, s\mathcal{C}'[se_1], \ldots, se'_k) \twoheadrightarrow st_j$. Then we can take $s\mathcal{C}'' = h(se'_1, \ldots, s\mathcal{C}', \ldots, se'_k)$ for which $s\mathcal{C}''[se_1] \twoheadrightarrow st_j$ has a proof of size less than K, and apply the IH to it to get $s\mathcal{C}''[se_2] \twoheadrightarrow st_j$, so we can build

$$\frac{\dfrac{hypothesis}{\{esa_1\} \twoheadrightarrow st_1} \ \ldots \ \dfrac{IH}{h(se'_1, \ldots, s\mathcal{C}'[se_1], \ldots, se'_k) \equiv s\mathcal{C}''[se_2] \twoheadrightarrow st_j} \ \ldots \ \dfrac{hypothesis}{\{esa_m\} \twoheadrightarrow st_m}}{s\mathcal{C}[se_2] \equiv \{ese_1, \ldots, h(se'_1, \ldots, s\mathcal{C}'[se_2], \ldots, se'_k), \ldots, ese_n\} \twoheadrightarrow st_1 \ \cup \ \ldots \ \cup \ st_j \ \cup \ \ldots \ \cup \ st_m} \text{ LESS}$$

ROR If $s\mathcal{C} = []$ then $s\mathcal{C}[se_1] \equiv se_1 \equiv \{f(se'_1, \ldots, se'_n)\}$ and so $\sharp se_1 = 1$, which combined with $se_1 \subseteq se_2$ implies $\sharp se_1 \geq 2$, and so

$$\frac{\dfrac{hypothesis}{\{f(se'_1, \ldots, se'_n)\} \equiv se_1 \equiv s\mathcal{C}[se_1] \twoheadrightarrow st}}{s\mathcal{C}[se_2] \equiv se_2 \twoheadrightarrow st} \text{ LESS}$$

as $\{f(se'_1, \ldots, se'_n)\} \equiv se_1 \subseteq se_2$. Otherwise if $s\mathcal{C} \neq []$ then the hypothesis was

$$\frac{se'_1 \twoheadrightarrow \tilde{p_1}\theta \ \ldots \ s\mathcal{C}'[se_1] \twoheadrightarrow \tilde{p'}\theta \ \ldots \ se'_n \twoheadrightarrow \tilde{p_n}\theta \ \tilde{r}\theta \twoheadrightarrow st}{s\mathcal{C}[se_1] \equiv \{f(se'_1, \ldots, s\mathcal{C}'[se_1], \ldots, se'_n)\} \twoheadrightarrow st} \text{ ROR}$$

but then we can apply the IH to $s\mathcal{C}'[se_1] \twoheadrightarrow \tilde{p'}\theta$ to get $s\mathcal{C}'[se_2] \twoheadrightarrow \tilde{p'}\theta$, so we can build

$$\frac{\dfrac{hypothesis}{se'_1 \twoheadrightarrow \tilde{p_1}\theta} \ \ldots \ \dfrac{IH}{s\mathcal{C}'[se_2] \twoheadrightarrow \tilde{p'}\theta} \ \ldots \ \dfrac{hypothesis}{se'_n \twoheadrightarrow \tilde{p_n}\theta} \ \dfrac{hypothesis}{\tilde{r}\theta \twoheadrightarrow st}}{s\mathcal{C}[se_2] \equiv \{f(se'_1, \ldots, s\mathcal{C}'[se_2], \ldots, se'_n)\} \twoheadrightarrow st} \text{ ROR}$$

Proof (For Theorem 1 (Compositionality)). For the part $\subseteq$, we must prove that for any context $s\mathcal{C}[se] \twoheadrightarrow st \Rightarrow \exists st' \in [\![se]\!]$ such that $s\mathcal{C}[st'] \twoheadrightarrow st$. First, we distinguish the possible contexts $s\mathcal{C}$:

a) if $s\mathcal{C} = [\,]$ we must prove: $se \twoheadrightarrow st \Rightarrow \exists st' \in [\![se]\!]$ such that $st' \twoheadrightarrow st$. This is trivial taking $st' = st$, as we have $st \twoheadrightarrow st$ for any $st \in SCTerm_\emptyset$ by Lemma 6.

b) if $s\mathcal{C} = \{\ldots, h(\ldots, s\mathcal{C}', \ldots), \ldots\}$, then we must prove: $\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st \Rightarrow \exists st' \in [\![se]\!]$ such that $\{\ldots, h(\ldots, s\mathcal{C}'[st'], \ldots), \ldots\} \twoheadrightarrow st$. We proceed by induction on the length K of the derivation for $\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st$.

- $K = 1$: if the derivation is $\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow \emptyset$ by rule **E** we can take any value st' in $[\![se]\!]$ and trivially make $\{\ldots, h(\ldots, s\mathcal{C}'[st'], \ldots), \ldots\} \twoheadrightarrow \emptyset$.
  The derivation can not be done by rule **RR** because of the syntactic forms. Neither it can be done by **DC** or some of the other rules in a single step.
- $K > 1$: the derivation can be done by the following rules:
  - by **DC**, with the form:

$$\frac{\ldots, s\mathcal{C}'[se] \twoheadrightarrow st'', \ldots}{\{c(\ldots, s\mathcal{C}'[se], \ldots)\} \twoheadrightarrow \{c(\ldots, st'', \ldots)\} \equiv st}$$

    Either by IH or by case *a*) if $s\mathcal{C}' = []$, there exists $st' \in [\![se]\!]$ such that $s\mathcal{C}'[st'] \twoheadrightarrow st''$ and then we can build the derivation for $\{c(\ldots, s\mathcal{C}'[st'], \ldots)\} \twoheadrightarrow \{c(\ldots, st'', \ldots)\} \equiv st$.

∗ by **More**, with the form:

$$\frac{\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st_1 \ldots \{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st_n}{\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st_1 \cup \ldots \cup st_n \equiv st}$$

Either by IH or by case *a*) if $s\mathcal{C}' = []$, there exists $st'_1, \ldots, st'_n \in [\![se]\!]$ such that $\{\ldots, h(\ldots, s\mathcal{C}'[st'_i], \ldots), \ldots\} \twoheadrightarrow st_i$ for each i. Now we define $st' = st'_1 \cup \ldots \cup st'_n$ so that $st'_i \subseteq st'$, and then by Lemma 13 we have $\{\ldots, h(\ldots, s\mathcal{C}'[st'], \ldots), \ldots\} \twoheadrightarrow st_i$ for each i. Then we can build the derivation

$$\frac{\{\ldots, h(\ldots, s\mathcal{C}'[st'], \ldots), \ldots\} \twoheadrightarrow st_1 \ldots \{\ldots, h(\ldots, s\mathcal{C}'[st'], \ldots), \ldots\} \twoheadrightarrow st_n}{\{\ldots, h(\ldots, s\mathcal{C}'[st'], \ldots), \ldots\} \twoheadrightarrow st_1 \cup \ldots \cup st_n \equiv st}$$

∗ by **Less**, with the form:

$$\frac{\{esa_1\} \twoheadrightarrow st_1 \ldots \{esa_m\} \twoheadrightarrow st_n}{\{ese_1, \ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots, ese_n\} \twoheadrightarrow st_1 \cup \ldots \cup st_m \equiv st}$$

taking $\{esa_1, \ldots, esa_m\} \subseteq \{ese_1, \ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots, ese_n\}$. If $h(\ldots, s\mathcal{C}'[se], \ldots) \notin \{esa_1, \ldots, esa_m\}$ the proof is trivial. In the other case, if $h(\ldots, s\mathcal{C}'[se], \ldots) = esa_i$ for some i, then either by IH or by case *a*) if $s\mathcal{C}' = []$, there exists $st' \in [\![se]\!]$ such that $s\mathcal{C}'[st'] \twoheadrightarrow st_i$ and we can build the derivation:

$$\frac{\{esa_1\} \twoheadrightarrow st_1 \ldots \{h(\ldots, s\mathcal{C}'[st'], \ldots)\} \twoheadrightarrow st_i \ldots \{esa_m\} \twoheadrightarrow st_n}{\{ese_1, \ldots, h(\ldots, s\mathcal{C}'[st'], \ldots), \ldots, ese_n\} \twoheadrightarrow st_1 \cup \ldots \cup st_i \cup \ldots \cup st_m \equiv st}$$

∗ by **ROR** with the form:

$$\frac{\ldots s\mathcal{C}'[se] \twoheadrightarrow \tilde{p}\theta \ldots \tilde{r}\theta \twoheadrightarrow st}{\{f(\ldots, s\mathcal{C}'[se], \ldots)\} \twoheadrightarrow st}$$

having $(f(\ldots, p, \ldots) \to r) \in \mathcal{P}$ and $\theta \in SCSubst_{\emptyset}$.
Either by IH or by case *a*) if $s\mathcal{C}' = []$, there exists $st' \in [\![se]\!]$ such that $s\mathcal{C}'[st'] \twoheadrightarrow \tilde{p}\theta$ and then we can build the proof for $\{f(\ldots, s\mathcal{C}'[st'], \ldots)\} \twoheadrightarrow st$.

For the part $\supseteq$ we must prove that for any context $s\mathcal{C}$ if $se \twoheadrightarrow st$ and $s\mathcal{C}[st] \twoheadrightarrow st'$ then $s\mathcal{C}[se] \twoheadrightarrow st'$. As before, we distinguish the possible contexts $s\mathcal{C}$:

a) if $s\mathcal{C} = [\,]$ we must prove: if $\exists st$ such that $se \twoheadrightarrow st$ and $st \twoheadrightarrow st'$, then $se \twoheadrightarrow st'$. By Lemma 8 $st \twoheadrightarrow st'$ implies $st' \sqsubseteq st$; on the other hand $se \sqsubseteq se$. So we can apply Lemma 1 to obtain $se \twoheadrightarrow st'$.
b) if $s\mathcal{C} = \{\ldots, h(\ldots, s\mathcal{C}', \ldots), \ldots\}$, then we must prove: if $\exists st \in [\![se]\!]$ such that $\{\ldots, h(\ldots, s\mathcal{C}'[st], \ldots), \ldots\} \twoheadrightarrow st'$, then $\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st'$. We proceed by induction on the length K of the derivation for $\{\ldots, h(\ldots, s\mathcal{C}'[st], \ldots), \ldots\} \twoheadrightarrow st'$:
   - $K = 1$: if the derivation is $\{\ldots, h(\ldots, s\mathcal{C}'[st], \ldots), \ldots\} \twoheadrightarrow \emptyset$ by rule **E**, trivially we can derive $\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots \emptyset$ by the same rule **E**. The derivation can not be done by rule **RR** because of the syntactic forms. Neither it can be done by **DC** or some of the other rules in a single step.
   - $K > 1$: the derivation can be done by the following rules:
     ∗ by **DC**, with the form:

$$\frac{\ldots, s\mathcal{C}'[st] \twoheadrightarrow st'', \ldots}{\{c(\ldots, s\mathcal{C}'[st], \ldots)\} \twoheadrightarrow \{c(\ldots, st'', \ldots)\} \equiv st'}$$

     Either by IH or by case *a*) if $s\mathcal{C}' = []$, $s\mathcal{C}'[se] \twoheadrightarrow st'$ and then we can build the derivation for $\{c(\ldots, s\mathcal{C}'[se], \ldots)\} \twoheadrightarrow \{c(\ldots, st'', \ldots)\} \equiv st'$.
     ∗ by **More**, with the form:

$$\frac{\{\ldots, h(\ldots, s\mathcal{C}'[st], \ldots), \ldots\} \twoheadrightarrow st_1 \ldots \{\ldots, h(\ldots, s\mathcal{C}'[st], \ldots), \ldots\} \twoheadrightarrow st_n}{\{\ldots, h(\ldots, s\mathcal{C}'[st], \ldots), \ldots\} \twoheadrightarrow st_1 \cup \ldots \cup st_n \equiv st'}$$

     Either by IH or by case *a*) if $s\mathcal{C}' = []$, $\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st_i$ for each i and we can build the derivation

$$\frac{\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st_1 \ldots \{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st_n}{\{\ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots\} \twoheadrightarrow st_1 \cup \ldots \cup st_n \equiv st'}$$

$*$ by **Less**, with the form:

$$\frac{\{esa_1\} \twoheadrightarrow st_1 \ldots \{esa_m\} \twoheadrightarrow st_n}{\{ese_1, \ldots, h(\ldots, s\mathcal{C}'[st], \ldots), \ldots, ese_n\} \twoheadrightarrow st_1 \cup \ldots \cup st_m \equiv st'}$$

taking $\{esa_1, \ldots, esa_m\} \subseteq \{ese_1, \ldots, h(\ldots, s\mathcal{C}'[st], \ldots), \ldots, ese_n\}$. If $h(\ldots, s\mathcal{C}'[st], \ldots) \notin \{esa_1, \ldots, esa_m\}$ the proof is trivial. In the other case, if $h(\ldots, s\mathcal{C}'[st], \ldots) = esa_i$ for some i, then either by IH or by case *a*) if $s\mathcal{C}' = []$, we have $\{h(\ldots, s\mathcal{C}'[se], \ldots) \twoheadrightarrow st_i$ and we can build the derivation:

$$\frac{\{esa_1\} \twoheadrightarrow st_1 \ldots \{h(\ldots, s\mathcal{C}'[se], \ldots)\} \twoheadrightarrow st_i \ldots \{esa_m\} \twoheadrightarrow st_n}{\{ese_1, \ldots, h(\ldots, s\mathcal{C}'[se], \ldots), \ldots, ese_n\} \twoheadrightarrow st_1 \cup \ldots \cup st_i \cup \ldots \cup st_m \equiv st'}$$

$*$ by **ROR** with the form:

$$\frac{\ldots s\mathcal{C}'[st] \twoheadrightarrow \tilde{p}\theta \ldots \tilde{r}\theta \twoheadrightarrow st'}{\{f(\ldots, s\mathcal{C}'[st], \ldots)\} \twoheadrightarrow st'}$$

having $(f(\ldots, p, \ldots) \to r) \in \mathcal{P}$ and $\theta \in SCSubst_{\emptyset}$.
Either by IH or by case *a*) if $s\mathcal{C}' = []$, we have $s\mathcal{C}'[se] \twoheadrightarrow \tilde{p}\theta$ and then we can build the proof for $\{f(\ldots, s\mathcal{C}'[se], \ldots)\} \twoheadrightarrow st'$.

Proof (For Proposition 3 (Closedness under substitutions)). First we prove *a)* by induction on the structure of $se \twoheadrightarrow st$. Concerning the base cases:

E Then we have $se \twoheadrightarrow \emptyset \equiv st$, but then $se\theta \twoheadrightarrow \emptyset \equiv \emptyset\theta \equiv st\theta$ by E too.
RR Then $se \equiv \{X\} \twoheadrightarrow \{X\} \equiv st$, but then $se\theta \equiv \theta(X) \in SCTerm$, as $\theta \in SCSubst$, and then $se\theta \equiv \theta(X) \twoheadrightarrow \theta(X) \equiv st\theta$ by Lemma 6.
DC Then $se \equiv \{c\} \twoheadrightarrow \{c\} \equiv st$, but then $se\theta \equiv \{c\} \twoheadrightarrow \{c\} \equiv \{c\}\theta \equiv st\theta$, by DC.

Concerning the inductive steps:

DC Then we can apply the IH to each $se_i \twoheadrightarrow st_i$ to get $se_i\theta \twoheadrightarrow st_i\theta$, and build

$$\frac{se_1\theta \twoheadrightarrow st_1\theta \quad \ldots \quad se_n\theta \twoheadrightarrow st_n\theta}{se\theta \equiv \{c(se_1\theta, \ldots, se_n\theta)\} \twoheadrightarrow \{c(st_1\theta, \ldots, st_n\theta)\} \equiv st\theta}\ \text{DC}$$

More Then we can apply the IH to each $se \twoheadrightarrow st_i$ to get $se\theta \twoheadrightarrow st_i\theta$, and build

$$\frac{se\theta \twoheadrightarrow st_1\theta \quad \ldots \quad se\theta \twoheadrightarrow st_n\theta}{se\theta \twoheadrightarrow st_1\theta \cup \ldots \cup st_n\theta \equiv st\theta}\ \textsc{More}$$

Less Then we can apply the IH to each $\{esa_i\} \twoheadrightarrow st_i$ to get $\{esa_i\}\theta \twoheadrightarrow st_i\theta$. But then $[\![\{esa_i\}\theta]\!] = [\![esa_i\theta]\!] \subseteq [\![se\theta]\!]$, by Lemma 11. So $\forall i, se\theta \twoheadrightarrow st_i\theta$, and we can build the following proof

$$\frac{se\theta \twoheadrightarrow st_1\theta \quad \ldots \quad se\theta \twoheadrightarrow st_m\theta}{se\theta \twoheadrightarrow st_1\theta \cup \ldots \cup st_m\theta \equiv st\theta}\ \textsc{More}$$

ROR Then we have

$$\frac{se_1 \twoheadrightarrow \tilde{p_1}\mu \quad \ldots \quad se_n \twoheadrightarrow \tilde{p_n}\mu \quad \tilde{r}\mu \twoheadrightarrow st}{se \equiv \{f(se_1, \ldots, se_n)\} \twoheadrightarrow st}\ \text{ROR}$$

Then we can apply the IH to each $se_i \twoheadrightarrow \tilde{p_i}\mu$ to get that $se_i\theta \twoheadrightarrow \tilde{p_i}\mu\theta$, and to $\tilde{r}\mu \twoheadrightarrow st$ to get that $\tilde{r}\mu\theta \twoheadrightarrow st\theta$. Besides as $\mu, \theta \in SCSubst$ then $\mu\theta \in SCSubst$ too, and so we can do

$$\frac{se_1\theta \twoheadrightarrow \tilde{p_1}\mu\theta \quad \ldots \quad se_n\theta \twoheadrightarrow \tilde{p_n}\mu\theta \quad \tilde{r}\mu\theta \twoheadrightarrow st\theta}{se\theta \equiv \{f(se_1\theta, \ldots, se_n\theta)\} \twoheadrightarrow st\theta}\ \text{ROR}$$

Now we can prove *b)* also. Assume $st\sigma \twoheadrightarrow st'$ for some $st' \in SCTerm$, then by Lemma 1 $\exists\theta \in [\![\sigma]\!]$ such that $st\theta \twoheadrightarrow st'$. But $\theta \in [\![\sigma]\!]$ implies $\theta \in SCSubst$ by definition and $\theta \trianglelefteq \sigma$ by Lemma 12. Therefore $st\theta \in SCTerm$ and so $st\theta \twoheadrightarrow st'$ implies $st' \sqsubseteq st\theta$ by Lemma 8, and as $se \twoheadrightarrow st$ implies $se\theta \twoheadrightarrow st\theta$ by part *a)*, then $se\theta \twoheadrightarrow st'$ by Prop. 1. But then we can apply Prop. 2 over $\theta \trianglelefteq \sigma$ and $se\theta \twoheadrightarrow st'$ to get $se\sigma \twoheadrightarrow st'$.

A.3 For Subsection 3.3

Definition 4. *For any non empty and finite set* $\{\theta_1, \ldots, \theta_n\} \subseteq SCSubst$ *we define* $\bigcup\{\theta_1, \ldots, \theta_n\} \in SCSubst$ *as* $\bigcup\{\theta_1, \ldots, \theta_n\}(X) = \theta_1(X) \cup \ldots \cup \theta_n(X)$.

Lemma 14. *For any non empty and finite set* $\{\theta_1, \ldots, \theta_n\} \subseteq SCSubst$.

i) $dom(\bigcup\{\theta_1, \ldots, \theta_n\}) = \bigcup_i dom(\theta_i)$.
ii) $\forall \theta_i \in \{\theta_1, \ldots, \theta_n\}$ *we have* $\theta_i \trianglelefteq \bigcup\{\theta_1, \ldots, \theta_n\}$.
iii) If $\forall \theta_i \in \{\theta_1, \ldots, \theta_n\}, \theta_i \in [\![\sigma]\!]$ *for some* $\sigma \in SSubst$ *then* $\bigcup\{\theta_1, \ldots, \theta_n\} \in [\![\sigma]\!]$ *too.*

Proof.

i) Given some $X \in \mathcal{V}$, if $X \in \bigcup_i dom(\theta_i)$ then $\exists \theta_i \in \{\theta_1, \ldots, \theta_n\}$ such that $X \in dom(\theta_i)$, hence $\theta_i(X) \not\equiv \{X\}$. But then $\bigcup\{\theta_1, \ldots, \theta_n\}(X) \equiv \theta_1(X) \cup \ldots \cup \theta_n(X) \not\equiv \{X\}$, hence $X \in dom(\bigcup\{\theta_1, \ldots, \theta_n\})$.
On the other hand if $X \notin \bigcup_i dom(\theta_i)$ then $\forall \theta_i \in \{\theta_1, \ldots, \theta_n\}$ we have $\theta_i(X) \equiv \{X\}$ and so $\bigcup\{\theta_1, \ldots, \theta_n\}(X) \equiv \theta_1(X) \cup \ldots \cup \theta_n(X) \equiv \{X\} \cup \ldots \cup \{X\} = \{X\}$, hence $X \notin dom(\bigcup\{\theta_1, \ldots, \theta_n\})$.
ii) Given any $X \in \mathcal{V}$ we have $\theta_i(X) \subseteq \bigcup\{\theta_1, \ldots, \theta_n\}(X)$ by definition, hence $[\![\theta_i(X)]\!] \subseteq [\![\bigcup\{\theta_1, \ldots, \theta_n\}(X)]\!]$ by Lemma 7 and Lemma 1.
iii) Given any $X \in \mathcal{V}$ we can build

$$\frac{\dfrac{\theta_1 \in [\![\sigma]\!]}{\sigma(X) \twoheadrightarrow \theta_1(X)} \quad \ldots \quad \dfrac{\theta_n \in [\![\sigma]\!]}{\sigma(X) \twoheadrightarrow \theta_n(X)}}{\sigma(X) \twoheadrightarrow \theta_1(X) \cup \ldots \cup \theta_n(X) \equiv \bigcup\{\theta_1, \ldots, \theta_n\}(X)} \textsc{More}$$

Proof (For Lemma 1). We proceed by a case distinction over se. If $se \equiv \{X\}$ for some $X \in dom(\sigma)$: Then the hypothesis is $\sigma(X) \twoheadrightarrow st$, and we can define $\theta \in SCSubst$ as

$$\theta(Y) = \begin{cases} st & \text{if } Y \equiv X \\ \emptyset & \text{if } Y \in dom(\sigma) \setminus \{X\} \\ \{Y\} & \text{otherwise} \end{cases}$$

But then $\theta \in [\![\sigma]\!]$, because given any $Y \in \mathcal{V}$ we have the following possibilities:

a) $Y \equiv X$: Then $\sigma(Y) \twoheadrightarrow st \equiv \theta(Y)$ by hypothesis.
b) $Y \in dom(\sigma) \setminus \{X\}$: Then $\sigma(Y) \twoheadrightarrow \emptyset \equiv \theta(Y)$, using rule E.
b) $Y \notin dom(\sigma)$: Then $\sigma(Y) \equiv \{Y\} \twoheadrightarrow \{Y\} \equiv \theta(Y)$, using rule RR.

Besides, $\theta(X) \equiv st \twoheadrightarrow st$ by Lemma 6, so we are done.

On the other hand, if $se \equiv \{X\}$ for some $X \in \mathcal{V} \setminus dom(\sigma)$: Then the hypothesis is $\sigma(X) \equiv \{X\} \twoheadrightarrow st$, so if $\overline{Y} = dom(\sigma)$ then we can take $\theta = [\overline{Y/\emptyset}]$ for which is very easy to check $\theta \in [\![\sigma]\!]$, in a similar way to the previous case. But then $\theta(X) \equiv \{X\} \twoheadrightarrow st$ by hypothesis.

Otherwise we proceed by induction on the structure of $se\sigma \twoheadrightarrow st$. Concerning the base cases:

E Then $st \equiv \emptyset$ and so if $\overline{Y} = dom(\sigma)$ then we can take $\theta = [\overline{Y/\emptyset}]$ for which is very easy to check $\theta \in [\![\sigma]\!]$. Then $se\theta \twoheadrightarrow \emptyset \equiv st$ by E, so we are done.
RR Then $se\sigma \equiv \{Y\} \twoheadrightarrow \{Y\} \equiv st$, and so by Lemma 10 $\forall ese \in se, ese\sigma \equiv \{Y\}$, hence $\forall ese \in se, ese \in \mathcal{V}$. Besides $se\sigma \equiv \{Y\}$ implies $se \not\equiv \emptyset$, therefore $\exists Z \in se \cap \mathcal{V}$ such that $\{Z\}\sigma \equiv Z\sigma \equiv \{Y\} \twoheadrightarrow st$ by hypothesis. But then by the proof of the cases when $se \equiv \{X\}$ we get some $\theta \in [\![\sigma]\!]$ such that $Z\theta \equiv \{Z\}\theta \twoheadrightarrow st$. But then $st \in [\![Z\theta]\!] \subseteq [\![se\theta]\!]$ by Lemma 11, and so $se\theta \twoheadrightarrow st$.
DC Then $se\sigma \equiv \{c\} \twoheadrightarrow \{c\} \equiv st$, and so by Lemma 10 $\forall ese \in se, ese\sigma \equiv \{c\}$. We have two possibilities:
a) $se \cap \mathcal{V} \neq \emptyset$: Then given some $Y \in se \cap \mathcal{V}$ we have $\{Y\}\sigma \equiv Y\sigma \equiv \{c\} \twoheadrightarrow st$ by hypothesis. But then by the proof of the cases when $se \equiv \{X\}$ we get some $\theta \in [\![\sigma]\!]$ such that $Y\theta \equiv \{Y\}\theta \twoheadrightarrow st$. But then $st \in [\![Y\theta]\!] \subseteq [\![se\theta]\!]$ by Lemma 11, and so $se\theta \twoheadrightarrow st$.

b) $se \cap \mathcal{V} = \emptyset$: Then $\forall ese \in se, ese\sigma \equiv \{c\}$ implies $\forall ese \in se, ese \equiv \{c\}$. Now if $\overline{Y} = dom(\sigma)$ we can take $\theta = [\overline{Y/\emptyset}]$ for which is very easy to check $\theta \in [\![\sigma]\!]$. Besides $se\sigma \equiv \{c\}$ implies $se \not\equiv \emptyset$, therefore $\exists ese \in se, ese\theta \equiv \{c\}\theta \equiv \{c\} \twoheadrightarrow st$ by hypothesis. But then $st \in [\![ese\theta]\!] \subseteq [\![se\theta]\!]$ by Lemma 11, and so $se\theta \twoheadrightarrow st$.

Concerning the inductive steps:

DC Then we have

$$\frac{se_1 \twoheadrightarrow st_1 \quad \ldots \quad se_n \twoheadrightarrow st_n}{se\sigma \equiv \{c(se_1, \ldots, se_n)\} \twoheadrightarrow \{c(st_1, \ldots, st_n)\} \equiv st} \text{ DC}$$

and so by Lemma 10 $\forall ese \in se, ese\sigma \equiv \{c(se_1, \ldots, se_n)\}$. We have two possibilities:

a) $se \cap \mathcal{V} \neq \emptyset$: Then given some $Y \in se \cap \mathcal{V}$ we have $\{Y\}\sigma \equiv Y\sigma \equiv \{c(se_1, \ldots, se_n)\} \twoheadrightarrow st$ by hypothesis. But then by the proof of the cases when $se \equiv \{X\}$ we get some $\theta \in [\![\sigma]\!]$ such that $Y\theta \equiv \{Y\}\theta \twoheadrightarrow st$. But then $st \in [\![Y\theta]\!] \subseteq [\![se\theta]\!]$ by Lemma 11, and so $se\theta \twoheadrightarrow st$.

b) $se \cap \mathcal{V} = \emptyset$: Then $se\sigma \equiv \{c(se_1, \ldots, se_n)\}$ implies $se \not\equiv \emptyset$. Therefore $\exists ese \in se = (se \setminus \mathcal{V})$ such that $ese\sigma \equiv \{c(se_1, \ldots, se_n)\}$, which implies $ese \equiv c(se'_1, \ldots, se'_n)$ such that $\forall i, se'_i\sigma \equiv se_i \twoheadrightarrow st_i$, to which we can apply the IH or the proof for the cases when $se \equiv \{X\}$ to get some $\theta_i \in [\![\sigma]\!]$ such that $se'_i\theta_i \twoheadrightarrow st_i$. But then we can take $\theta = \bigcup\{\theta_1, \ldots, \theta_n\} \in [\![\sigma]\!]$ by Lemma 14, to get $\forall i, \theta_i \trianglelefteq \theta$ by Lemma 14, hence $\forall i, se'_i\theta \twoheadrightarrow st_i$ by Lemma 2 and we can build the following proof:

$$\frac{se'_1\theta \twoheadrightarrow st_1 \quad \ldots \quad se'_n\theta \twoheadrightarrow st_n}{ese\theta \equiv \{c(se'_1\theta, \ldots, se'_n\theta)\} \twoheadrightarrow \{c(st_1, \ldots, st_n)\} \equiv st} \text{ DC}$$

But then $st \in [\![ese\theta]\!] \subseteq [\![se\theta]\!]$ by Lemma 11, and so $se\theta \twoheadrightarrow st$.

More Then we have

$$\frac{se\sigma \twoheadrightarrow st_1 \quad \ldots \quad se\sigma \twoheadrightarrow st_n}{se\sigma \twoheadrightarrow st_1 \cup \ldots \cup st_n \equiv st} \text{ More}$$

But then we can apply the IH or the proof for the cases when $se \equiv \{X\}$ to each $se\sigma \twoheadrightarrow st_i$ to get some $\theta_i \in [\![\sigma]\!]$ such that $se\theta_i \twoheadrightarrow st_i$. But then we can take $\theta = \bigcup\{\theta_1, \ldots, \theta_n\} \in [\![\sigma]\!]$ by Lemma 14, to get $\forall i, \theta_i \trianglelefteq \theta$ by Lemma 14, hence $\forall i, se\theta \twoheadrightarrow st_i$ by Lemma 2 and we can build the following proof:

$$\frac{se\theta \twoheadrightarrow st_1 \quad \ldots \quad se\theta \twoheadrightarrow st_n}{se\theta \twoheadrightarrow st_1 \cup \ldots \cup st_n \equiv st} \text{ More}$$

Less Then we have

$$\frac{\{esa_1\} \twoheadrightarrow st_1 \quad \ldots \quad \{esa_n\} \twoheadrightarrow st_m}{se\sigma \twoheadrightarrow st_1 \cup \ldots \cup st_m \equiv st} \text{ Less}$$

for some $\{esa_1, \ldots, esa_m\} \subseteq se\sigma$ and $\sharp se\sigma \geq 2$. Then for any $esa_i \in \{esa_1, \ldots, esa_m\} \subseteq se\sigma$ we have two possibilities:

a) $esa_i \in \sigma(X)$ for some $X \in se$: Then $st_i \in [\![\{esa_i\}]\!] \subseteq [\![\sigma(X)]\!] = [\![\{X\}\sigma]\!]$ by Lemma 7 and Lemma 1, as $\{esa_i\} \subseteq \sigma(X)$. Therefore by the proof of the case when $se \equiv \{X\}$ we get some $\theta_i \in [\![\sigma]\!]$ such that $\{X\}\theta_i \twoheadrightarrow st_i$. But then $X \in se$ implies $st_i \in [\![\{X\}\theta_i]\!] = [\![X\theta_i]\!] \subseteq [\![se\theta_i]\!]$ by Lemma 11.

b) $esa_i \equiv h(se_1\sigma, \ldots, se_n\sigma)$ for some $h(se_1, \ldots, se_n) \in se$. But then $\{h(se_1, \ldots, se_n)\}\sigma \equiv \{h(se_1\sigma, \ldots, se_n\sigma)\} \equiv \{esa_i\} \twoheadrightarrow st_i$ by hypothesis, and we can apply the IH to get some $\theta_i \in [\![\sigma]\!]$ such that $\{h(se_1, \ldots, se_n)\}\theta_i \twoheadrightarrow st_i$. But then $h(se_1, \ldots, se_n) \in se$ implies $st_i \in [\![\{h(se_1, \ldots, se_n)\}\theta_i]\!] = [\![h(se_1, \ldots, se_n)\theta_i]\!] \subseteq [\![se\theta_i]\!]$ by Lemma 11.

Therefore $\forall i \in \{1, \ldots, m\}, \exists \theta_i \in [\![\sigma]\!], se\theta_i \twoheadrightarrow st_i$. But then we can take $\theta = \bigcup\{\theta_1, \ldots, \theta_m\} \in [\![\sigma]\!]$ by Lemma 14, to get $\forall i, \theta_i \trianglelefteq \theta$ by Lemma 14, hence $\forall i, se\theta \twoheadrightarrow st_i$ by Lemma 2 and we can build the following proof:

$$\frac{se\theta \twoheadrightarrow st_1 \quad \ldots \quad se\theta \twoheadrightarrow st_m}{se\theta \twoheadrightarrow st_1 \cup \ldots \cup st_m \equiv st} \text{ More}$$

ROR Then we have

$$\frac{se_1 \twoheadrightarrow \tilde{p_1}\mu \quad \ldots \quad se_n \twoheadrightarrow \tilde{p_n}\mu \quad \tilde{r}\mu \twoheadrightarrow st}{se\sigma \equiv \{f(se_1, \ldots, se_n)\} \twoheadrightarrow st} \text{ ROR}$$

and so by Lemma 10 $\forall ese \in se, ese\sigma \equiv \{f(se_1, \ldots, se_n)\}$. We have two possibilities:

a) $se \cap \mathcal{V} \neq \emptyset$: Then given some $Y \in se \cap \mathcal{V}$ we have $\{Y\}\sigma \equiv Y\sigma \equiv \{f(se_1, \ldots, se_n)\} \twoheadrightarrow st$ by hypothesis. But then by the proof of the cases when $se \equiv \{X\}$ we get some $\theta \in [\![\sigma]\!]$ such that $Y\theta \equiv \{Y\}\theta \twoheadrightarrow st$. But then $st \in [\![Y\theta]\!] \subseteq [\![se\theta]\!]$ by Lemma 11, and so $se\theta \twoheadrightarrow st$.

b) $se \cap \mathcal{V} = \emptyset$: Then $se\sigma \equiv \{f(se_1, \ldots, se_n)\}$ implies $se \not\equiv \emptyset$. Therefore $\exists ese \in se = (se \setminus \mathcal{V})$ such that $ese\sigma \equiv \{f(se_1, \ldots, se_n)\}$, which implies $ese \equiv f(se'_1, \ldots, se'_n)$ such that $\forall i, se'_i\sigma \equiv se_i \twoheadrightarrow \tilde{p_i}\mu$, to which we can apply the IH or the proof for the cases when $se \equiv \{X\}$ to get some $\theta_i \in [\![\sigma]\!]$ such that $se'_i\theta_i \twoheadrightarrow \tilde{p_i}\mu$. But then we can take $\theta = \bigcup\{\theta_1, \ldots, \theta_n\} \in [\![\sigma]\!]$ by Lemma 14, to get $\forall i, \theta_i \trianglelefteq \theta$ by Lemma 14, hence $\forall i, se'_i\theta \twoheadrightarrow \tilde{p_i}\mu$ by Lemma 2 and we can build the following proof:

$$\frac{se'_1\theta \twoheadrightarrow \tilde{p_1}\mu \quad \ldots \quad se'_n\theta \twoheadrightarrow \tilde{p_n}\mu \qquad \overset{hypothesis}{\tilde{r}\mu \twoheadrightarrow st}}{ese\theta \equiv \{f(se'_1\theta, \ldots, se'_n\theta)\} \twoheadrightarrow st}\ \text{ROR}$$

But then $st \in [\![ese\theta]\!] \subseteq [\![se\theta]\!]$ by Lemma 11, and so $se\theta \twoheadrightarrow st$.

Lemma 15. *Under any program, $\forall e, e' \in Exp$ if $e \to e'$ then $[\![\tilde{e'}]\!] \subseteq [\![\tilde{e}]\!]$.*

Proof. Assume $e \to e'$. If the step was performed at the root then we have $e \equiv f(p_1, \ldots, p_n)\sigma \to r\sigma \equiv e'$ for some rule $(f(p_1, \ldots, p_n) \to r) \in \mathcal{P}$. Assume $\tilde{e'} \equiv \widetilde{r\sigma} \twoheadrightarrow st$ for some $st \in SCTerm$, then we have $\tilde{r}\tilde{\sigma} \equiv \widetilde{r\sigma} \twoheadrightarrow st$ by Lemma 9, but then $\exists \theta \in [\![\tilde{\sigma}]\!]$ such that $\tilde{r}\theta \twoheadrightarrow st$, by Lemma 1. Besides for each p_i we have $p_i \in CTerm$ but then is easy to prove that $\tilde{p_i} \in SCTerm$ and so $\tilde{p_i}\theta \in SCTerm$, because $\theta \in [\![\tilde{\sigma}]\!]$ implies $\theta \in SCSubst$. But then by Lemma 6 we have $\tilde{p_i}\theta \twoheadrightarrow \tilde{p_i}\theta$, and then we can build

$$\frac{\tilde{p_1}\theta \twoheadrightarrow \tilde{p_1}\theta \quad \ldots \quad \tilde{p_n}\theta \twoheadrightarrow \tilde{p_n}\theta \quad \tilde{r}\theta \twoheadrightarrow st}{\widetilde{f(p_1, \ldots, p_n)}\theta \equiv \{f(\tilde{p_1}\theta, \ldots, \tilde{p_n}\theta)\} \twoheadrightarrow st}\ \text{ROR}$$

But $\theta \in [\![\tilde{\sigma}]\!]$ implies $\theta \trianglelefteq \tilde{\sigma}$ by Lemma 12, and so $\widetilde{f(p_1, \ldots, p_n)}\theta \twoheadrightarrow st$ implies $\widetilde{f(p_1, \ldots, p_n)}\tilde{\sigma} \twoheadrightarrow st$ by Lemma 2. But $\widetilde{f(p_1, \ldots, p_n)}\tilde{\sigma} \equiv \widetilde{f(p_1, \ldots, p_n)}\sigma$ by Lemma 9, therefore $\tilde{e} \equiv \widetilde{f(p_1, \ldots, p_n)}\sigma \equiv \widetilde{f(p_1, \ldots, p_n)}\tilde{\sigma} \twoheadrightarrow st$, and so we have proved that $[\![\tilde{e'}]\!] \subseteq [\![\tilde{e}]\!]$.

Otherwise if the step was not performed at the root we have $e \equiv \mathcal{C}[f(\overline{p})\sigma] \to \mathcal{C}[r\sigma] \equiv e'$, where $f(\overline{p})\sigma \to r\sigma$ has been performed at the root and so by the proof of the other case $[\![r\sigma]\!] \subseteq [\![f(\overline{p})\sigma]\!]$. Assume $\tilde{e'} \equiv \widetilde{\mathcal{C}[r\sigma]} \twoheadrightarrow st$ for some $st \in SCTerm$, then we can chain:

$$\begin{array}{ll}
[\![\tilde{e'}]\!] = [\![\widetilde{\mathcal{C}[r\sigma]}]\!] & \\
= [\![\tilde{\mathcal{C}}[\widetilde{r\sigma}]]\!] & \text{by Lemma 9} \\
= \bigcup_{st \in [\![\widetilde{r\sigma}]\!]} [\![\tilde{\mathcal{C}}[st]]\!] & \text{by Theorem 1} \\
\subseteq \bigcup_{st \in [\![\widetilde{f(\overline{p})\sigma}]\!]} [\![\tilde{\mathcal{C}}[st]]\!] & \text{as } [\![r\sigma]\!] \subseteq [\![f(\overline{p})\sigma]\!] \\
= [\![\widetilde{\mathcal{C}[f(\overline{p})\sigma]}]\!] & \text{by Theorem 1} \\
= [\![\widetilde{\mathcal{C}[f(\overline{p})\sigma]}]\!] = [\![\tilde{e}]\!] & \text{by Lemma 9}
\end{array}$$

Proof (For Proposition 4). A simple induction on the length of the derivation $e \to^* e'$. The base case is trivial and the inductive step is straightforward too, using Lemma 15 for the first step, applying the IH to the others and using the transitivity of set inclusions to chain those results.

Lemma 16. *Under any program and $\forall st, st' \in SCTerm, est, est' \in ESCTerm, e \in Exp$:*

- *If $st \sqsubseteq st'$ and $st' \lessdot e$ then $st \lessdot e$.*
- *If $est \sqsubseteq est'$ and $est' \lessdot e$ then $est \lessdot e$.*

Proof. We proceed by induction on the structure of st' and est'. The base cases are the following:

- $st' \equiv \emptyset$: Then $st \sqsubseteq st'$ implies $st \equiv \emptyset$, but then $st \equiv \emptyset \lessdot e$.
- $est' \equiv X \in \mathcal{V}$: Then $est \sqsubseteq est'$ implies $est \equiv X \equiv est'$, but then $est \equiv est' \lessdot e$ by hypothesis.
- $est' \equiv c \in CS^0$: Then $est \sqsubseteq est'$ implies $est \equiv c \equiv est'$, but then $est \equiv est' \lessdot e$

Concerning the inductive step:

- $st' \not\equiv \emptyset$: Then $st \sqsubseteq st'$ implies that $\forall est \in st, \exists est' \in st'$ such that $est \sqsubseteq est'$. But as $st' \lessdot e$, then for that est' we have $est' \lessdot e$ too. So we can apply the IH with $est \sqsubseteq est'$ and $est' \lessdot e$ to get that $est \lessdot e$, hence $\forall est \in st, est \lessdot e$, that is, $st \lessdot e$.
- $est' \equiv c(st'_1, \ldots, st'_n)$: Then $est \sqsubseteq est'$ implies $est \equiv c(st_1, \ldots, st_n)$ such that $\forall i, st_i \sqsubseteq st'_i$, and $est' \lessdot e$ implies $e \rightarrow^* c(e_1, \ldots, e_n)$ such that $\forall i, st'_i \lessdot e_i$. But then we can apply the IH to each $st_i \sqsubseteq st'_i$ and $st'_i \lessdot e_i$ to get $st_i \lessdot e_i$, hence $est \lessdot e$.

Lemma 17. *Under any program and $\forall \theta \in SCSubst, \sigma \in Subst$ if $dom(\theta) = dom(\sigma)$ and $\forall X \in dom(\theta), \theta(X) \lessdot \sigma(X)$, then $\theta \lessdot \sigma$.*

Proof. Given $X \in \mathcal{V}$, if $X \in dom(\theta)$ then $\theta(X) \lessdot \sigma(X)$ by hypothesis. Otherwise $X \notin dom(\theta) = dom(\sigma)$, therefore $\theta(X) \equiv X \lessdot X \equiv \sigma(X)$, as $X \rightarrow^0 X$.

Lemma 18. *Under any program and $\forall e \in Exp, p \in CTerm$ linear and $\theta \in SCSubst$ such that $dom(\theta) \subseteq var(p)$ we have that $\tilde{p}\theta \lessdot e$ implies that $\exists \sigma \in Subst$ such that $dom(\sigma) = dom(\theta)$, $\theta \lessdot \sigma$ and $e \rightarrow^* p\sigma$.*

Proof. We proceed by induction on the structure of p, the base cases are the following:

- $p \equiv X \in \mathcal{V}$: If $X \notin dom(\theta) \subseteq var(p) = \{X\}$ then $\theta = \epsilon$, hence $\tilde{p}\theta \lessdot e$ is equivalent to $\{X\} \lessdot e$, which implies $e \rightarrow^* X$. But then we can take $\sigma = \epsilon$ for which $dom(\sigma) = \emptyset = dom(\theta)$, $\theta = \epsilon \lessdot \epsilon = \sigma$ (because $\epsilon \lessdot \epsilon$ by Lemma 17) and $e \rightarrow^* X \equiv X\epsilon \equiv p\sigma$.
  On the other hand if $X \in dom(\theta) \subseteq var(p) = \{X\}$ then $dom(\theta) = \{X\}$ and $\tilde{p}\theta \lessdot e$ is equivalent to $\theta(X) \lessdot e$. But then we can take $\sigma = [X/e]$ for which $dom(\theta) = \{X\} = dom(\sigma)$ and $\theta(X) \lessdot e \equiv \sigma(X)$, therefore $\theta \lessdot \sigma$ by Lemma 17. Besides $e \rightarrow^0 e \equiv X[X/e] \equiv p\sigma$, so we are done.
- $p \equiv c \in CS^0$: Then as $dom(\theta) \subseteq var(p) = \emptyset$ we have $\theta = \epsilon$, hence $\tilde{p}\theta \lessdot e$ is equivalent to $\{c\} \lessdot e$, which implies $e \rightarrow^* c$. But then we can take $\sigma = \epsilon$ for which $dom(\sigma) = \emptyset = dom(\theta)$, $\theta = \epsilon \lessdot \epsilon = \sigma$ (because $\epsilon \lessdot \epsilon$ by Lemma 17) and $e \rightarrow^* c \equiv c\epsilon \equiv p\sigma$.

Concerning the inductive step, this happens when $p \equiv c(p_1, \ldots, p_n)$. Then as p is linear and $dom(\theta) \subseteq var(p)$, if $\forall i \in \{1, \ldots, n\}$ we define $\theta_i = \theta|_{var(p_i)}$, then $\theta = \theta_1 \uplus \ldots \uplus \theta_n$. Besides the hypothesis $\tilde{p}\theta \lessdot e$ is equivalent to $\{c(\tilde{p_1}, \ldots, \tilde{p_n})\}\theta \equiv \{c(\tilde{p_1}\theta, \ldots, \tilde{p_n}\theta)\} \equiv \{c(\tilde{p_1}\theta_1, \ldots, \tilde{p_n}\theta_n)\} \lessdot e$ (as $var(e_i) = var(\tilde{e_i})$, by Prop. 8), hence $e \rightarrow^* c(e_1, \ldots, e_n)$ such that $\forall i, \tilde{p_i}\theta_i \lessdot e_i$. But then by IH $\forall i, \exists \sigma_i \in Subst$ such that $dom(\sigma_i) = dom(\theta_i)$, $\theta_i \lessdot \sigma_i$ and $e_i \rightarrow^* p_i\sigma_i$. But $\forall i, dom(\sigma_i) = dom(\theta_i) \subseteq var(p_i)$ hence as p is linear then $\sigma = \sigma_1 \uplus \ldots \uplus \sigma_n$ is correctly defined and $\theta \lessdot \sigma$ holds, because $dom(\sigma) = \bigcup_i dom(\sigma_i) = \bigcup_i dom(\theta_i) = dom(\theta)$ and given some $X \in dom(\theta)$ then $X \in dom(\theta_i)$ for some θ_i and then $\theta(X) \equiv \theta_i(X) \lessdot \sigma_i(X) \equiv \sigma(X)$, therefore we can apply Lemma 17. Finally we can chain $e \rightarrow^* c(e_1, \ldots, e_n) \rightarrow^* c(p_1\sigma_1, \ldots, p_n\sigma_n) \equiv c(p_1, \ldots, p_n)\sigma \equiv p\sigma$, so we are done.

Lemma 19. *Under any program and $\forall e, e' \in Exp, st \in SCTerm$ if $st \lessdot e'$ and $e \rightarrow^* e'$ then $st \lessdot e$.*

Proof. It is enough to check that $\forall est \in st, est \lessdot e$, and that happens because:

- If $est \equiv X \in \mathcal{V}$ then $est \in st$ and $st \lessdot e'$ implies $X \equiv est \lessdot e'$, and so $e' \rightarrow^* X$. But then $e \rightarrow^* e' \rightarrow^* X$, hence $X \equiv est \lessdot e$ too.
- If $est \equiv c(st_1, \ldots, st_n)$ then $est \in st$ and $st \lessdot e'$ implies $c(st_1, \ldots, st_n) \equiv est \lessdot e'$, and so $e' \rightarrow^* c(e_1, \ldots, e_n)$ such that $\forall i, st_i \lessdot e_i$. But then $e \rightarrow^* e' \rightarrow^* c(e_1, \ldots, e_n)$, hence $c(st_1, \ldots, st_n) \equiv est \lessdot e$ too.

Proof (For Lemma 2). We proceed by a case distinction over e.
If $e \equiv X \in \mathcal{V}$ then $\tilde{e} \equiv \{X\} \in SCTerm$, but as $\theta \in SCSubst$ then $\tilde{e}\theta \equiv \theta(X) \in SCTerm$, and so $\theta(X) \equiv \tilde{e}\theta \twoheadrightarrow st$ implies $st \sqsubseteq \theta(X)$, by Lemma 8. Besides $\theta \lessdot \sigma$ implies $\theta(X) \lessdot \sigma(X)$ by definition, therefore we can combine it with $st \sqsubseteq \theta(X)$ to get $st \lessdot \sigma(X) \equiv e\sigma$ by Lemma 16.
Otherwise we will prove the case when e is not restricted to be a variable by induction on the structure of $\tilde{e}\theta \twoheadrightarrow st$. The base cases are the following:

E Then $st \equiv \emptyset$, therefore for any $\sigma \in Subst$ we have $st \equiv \emptyset \lessdot e\sigma$.
RR Then $\tilde{e}\theta \equiv \{X\}$, but that implies $e \in \mathcal{V}$, so we can proceed like in the previous case, at the beginning of the proof.

DC If $e \in \mathcal{V}$ then we proceed like in the previous case. Otherwise $e \equiv c \in CS^0$ and $st \equiv \{c\}$. But then for any $\sigma \in Subst$ $e\sigma \equiv c \to^0 c$, therefore $st \equiv \{c\} \lessdot e\sigma$.

Regarding the inductive steps:

DC If $e \in \mathcal{V}$ then we proceed like in the previous case. Otherwise $e \equiv c(e_1, \ldots, e_n)$ and the proof has the following case:

$$\frac{\tilde{e_1}\theta \twoheadrightarrow st_1 \quad \ldots \quad \tilde{e_n}\theta \twoheadrightarrow st_n}{\tilde{e}\theta \equiv \{c(\tilde{e_1}\theta, \ldots, \tilde{e_n}\theta)\} \twoheadrightarrow \{c(st_1, \ldots, st_n)\} \equiv st} \text{ DC}$$

Now given some $\sigma \in Subst$ such that $\theta \lessdot \sigma$ we can apply the IH to each $\tilde{e_i}\theta \twoheadrightarrow st_i$ to get that $st_i \lessdot e_i\sigma$. But then $e\sigma \equiv c(e_1\sigma, \ldots, e_n\sigma) \to^0 c(e_1\sigma, \ldots, e_n\sigma)$, therefore $st \equiv \{c(st_1, \ldots, st_n)\} \lessdot e\sigma$.

More Then the proof has the following shape:

$$\frac{\tilde{e}\theta \twoheadrightarrow st_1 \quad \ldots \quad \tilde{e}\theta \twoheadrightarrow st_n}{\tilde{e}\theta \twoheadrightarrow st_1 \cup \ldots \cup st_n \equiv st} \text{ More}$$

Now given some $\sigma \in Subst$ such that $\theta \lessdot \sigma$ we can apply the IH to each $\tilde{e}\theta \twoheadrightarrow st_i$ to get $st_i \lessdot e\sigma$, which implies that $\forall i, \forall est \in st_i, est \lessdot e\sigma$, therefore $st \equiv st_1 \cup \ldots \cup st_n \lessdot e\sigma$.

Less If $e \in \mathcal{V}$ then we proceed like in the previous case. Otherwise $e \equiv h(e_1, \ldots, e_n)$, hence $\tilde{e}\theta \equiv \{h(\tilde{e_1}\theta, \ldots, \tilde{e_n}\theta)\}$ and therefore Less cannot have been applied.

ROR If $e \in \mathcal{V}$ then we proceed like in the previous case. Otherwise $e \equiv f(e_1, \ldots, e_n)$ and the proof has the following shape:

$$\frac{\tilde{e_1}\theta \twoheadrightarrow \tilde{p_1}\mu \quad \ldots \quad \tilde{e_n}\theta \twoheadrightarrow \tilde{p_n}\mu \quad \tilde{r}\mu \twoheadrightarrow st}{\tilde{e}\theta \equiv \{f(\tilde{e_1}\theta, \ldots, \tilde{e_n}\theta)\} \twoheadrightarrow st} \text{ ROR}$$

for some $(f(p_1, \ldots, p_n) \to r) \in \mathcal{P}$. But then $var(r) \subseteq var(f(p_1, \ldots, p_n))$ and so we may assume $dom(\mu) \subseteq var(f(p_1, \ldots, p_n))$ without loss of generality. Besides, as $f(p_1, \ldots, p_n)$ is linear, if for each $i \in \{1, \ldots, n\}$ we define $\mu_i = \mu|_{var(\tilde{p_i})} = \mu_{var(p_i)}$ (as $var(\tilde{p_i}) = var(p_i)$ by Prop. 8), then $\mu = \mu_1 \uplus \ldots \uplus \mu_n$ and $\forall i, \tilde{p_i}\mu \equiv \tilde{p_i}\mu_i$.
Now given some $\sigma \in Subst$ such that $\theta \lessdot \sigma$ we can apply the IH to each $\tilde{e_i}\theta \twoheadrightarrow \tilde{p_i}\mu$ to get $\forall i, \tilde{p_i}\mu_i \equiv \tilde{p_i}\mu \lessdot e_i\sigma$. But then by Lemma 18 we have that $\forall i, \exists \sigma_i' \in Subst$ such that $dom(\sigma_i') = dom(\mu_i)$, $\mu_i \lessdot \sigma_i'$ and $e_i\sigma \to^* p_i\sigma_i'$. As $\forall i, dom(\sigma_i') = dom(\mu_i) \subseteq var(p_i)$, this combined with the linearity of $f(p_1, \ldots, p_n)$ implies that $\sigma' = \sigma_1' \uplus \ldots \uplus \sigma_n'$ is correctly defined. Furthermore, $\mu \lessdot \sigma'$ because $dom(\sigma') = \bigcup_i dom(\sigma_i') = \bigcup_i dom(\mu_i) = dom(\mu)$ and given some $X \in dom(\mu)$ then $X \in dom(\mu_i)$ for some i and then $\mu(X) \equiv \mu_i(X) \lessdot \sigma_i'(X) \equiv \sigma'(X)$, as $\mu_i \lessdot \sigma_i'$, therefore we can apply Lemma 17.
Finally, we can use σ' to apply the IH to $\tilde{r}\mu \twoheadrightarrow st$, getting that $st \lessdot r\sigma'$. Besides $e\sigma \equiv f(e_1\sigma, \ldots, e_n\sigma) \to^* f(p_1\sigma_1', \ldots, p_n\sigma_n') \equiv f(p_1, \ldots, p_n)\sigma' \to r\sigma'$, so we can combine $st \lessdot r\sigma'$ with $e\sigma \to^* r\sigma'$ using Lemma 19 to get that $st \lessdot e\sigma$.

Proof (For Lemma 3 (Domination)). Assume $\tilde{e} \twoheadrightarrow st$, then as $\epsilon \lessdot \epsilon$ we can apply Lemma 2 using $\theta = \epsilon = \sigma$ to get $st \lessdot e\sigma \equiv e$. To prove the converse implication we will prove that $\forall e \in Exp, st \in SCTerm, est \in ESCTerm$ $st \lessdot e$, $st \lessdot e$ implies $\tilde{e} \twoheadrightarrow st$ and $est \lessdot e$ implies $\tilde{e} \twoheadrightarrow \{est\}$, by induction on the proof for $st \lessdot e$ and $est \lessdot e$. The base cases are the following:

- $st \equiv \emptyset$: Then $\tilde{e} \twoheadrightarrow \emptyset \equiv st$ by E.
- $est \equiv X$: Then $X \lessdot e$ implies $e \to^* X$, which implies $[\![\{X\}]\!] \subseteq [\![\tilde{e}]\!]$, by Prop. 4. But $\{X\} \in [\![\{X\}]\!] \subseteq [\![\tilde{e}]\!]$, that is, $\tilde{e} \twoheadrightarrow \{X\} \equiv \{est\}$.
- $est \equiv c$: Then $c \lessdot e$ implies $e \to^* c$, which implies $[\![\{c\}]\!] \subseteq [\![\tilde{e}]\!]$, by Prop. 4. But $\{c\} \in [\![\{c\}]\!] \subseteq [\![\tilde{e}]\!]$, that is, $\tilde{e} \twoheadrightarrow \{c\} \equiv \{est\}$.

Concerning the inductive steps:

- $st \not\equiv \emptyset$: Then $\sharp st > 0$ and we have the following possibilities:
  i) $\sharp st = 1$: Then $st \equiv \{est\} \lessdot e$ implies $est \lessdot e$ by definition, but then $\tilde{e} \twoheadrightarrow \{est\} \equiv st$ by IH.
  ii) $\sharp st > 1$: Then $st \equiv \{est_1, \ldots, est_n\}$ and $st \lessdot e$ implies $\forall st_i \in st, est_i \lessdot e$ by definition. But then we can apply the IH to each $est_i \lessdot e$ to get $\tilde{e} \twoheadrightarrow \{est_i\}$ by IH, and we can build the following proof:

$$\frac{\tilde{e} \twoheadrightarrow \{est_1\} \quad \ldots \quad \tilde{e} \twoheadrightarrow \{est_n\}}{\tilde{e} \twoheadrightarrow \{est_1\} \cup \ldots \cup \{est_n\} \equiv st} \text{ More}$$

– $est \equiv c(st_1, \ldots, st_n)$: Then $c(st_1, \ldots, st_n) \lessdot e$ implies $e \rightarrow^* c(e_1, \ldots, e_n)$ such that $\forall i \in \{1, \ldots, n\}, st_i \lessdot e_i$. Then, by Prop. 4, $e \rightarrow^* c(e_1, \ldots, e_n)$ implies $[\![\widetilde{c(e_1, \ldots, e_n)}]\!] \subseteq [\![\tilde{e}]\!]$. Besides we can apply the IH to each $st_i \lessdot e_i$ to get $\tilde{e_i} \twoheadrightarrow st_i$, and build the following proof:

$$\frac{\tilde{e_1} \twoheadrightarrow st_1 \quad \ldots \quad \tilde{e_n} \twoheadrightarrow st_n}{\widetilde{c(e_1, \ldots, e_n)} \equiv \{c(\tilde{e_1}, \ldots, \tilde{e_n})\} \twoheadrightarrow \{c(st_1, \ldots, st_n)\} \equiv \{est\}} \text{ DC}$$

But then $\{est\} \in [\![\widetilde{c(e_1, \ldots, e_n)}]\!] \subseteq [\![\tilde{e}]\!]$ implies $\tilde{e} \twoheadrightarrow \{est\}$.

Proof (For Lemma 4). To prove this lemma we prove a slight generalization of it:

$\forall st \in SCTerm, est \in ESCTerm, e \in Exp$ if $st \lessdot e$ ($est \lessdot e$ resp.) then $\forall t \in flat(st)$ ($\forall t \in flat(est)$ resp.) we have $e \rightarrow^* e'$ for some $e' \in Exp$ such that $t \sqsubseteq |e'|$.

We proceed by induction on the structure of $SExp$ and $ESExp$. Concerning the base cases:

– $st \equiv \emptyset$: Then $flat(st) = \{\bot\}$ and $e \rightarrow^0 e$, with $\bot \sqsubseteq |e|$, so we are done.
– $est \equiv X$: Then $flat(est) = \{X\}$ and $X \equiv est \lessdot e$ implies $e \rightarrow^* X$, with $X \sqsubseteq X \equiv |X|$, so we are done.
– $est \equiv c$: Then $flat(est) = \{c\}$ and $c \equiv est \lessdot e$ implies $e \rightarrow^* c$, with $c \sqsubseteq c \equiv |c|$, so we are done.

Concerning the inductive steps:

– $st \not\equiv \emptyset$: Then given $t \in flat(st) = \bigcup_{est \in st} flat(est)$ then $\exists est_i \in st$ such that $t \in flat(est_i)$. Besides $st \lessdot e$ implies $est_i \lessdot e$ by definition, and so we can apply the HI over $est_i \lessdot e$ and $t \in flat(est_i)$ to get that $e \rightarrow^* e'$ such that $t \sqsubseteq |e'|$.
– $est \equiv c(st_1, \ldots, st_n)$: Then given $t \in flat(est)$ it must be $t \equiv c(t_1, \ldots, t_n)$ such that $\forall i \in \{1, \ldots, n\}, t_i \in flat(st_i)$. Besides $est \lessdot e$ implies $e \rightarrow^* c(e_1, \ldots, e_n)$ such that $\forall i \in \{1, \ldots, n\}, st_i \lessdot e_i$, hence we can apply the IH to each $st_i \lessdot e_i$ and $t_i \in flat(st_i)$ to get $e_i \rightarrow^* e_i'$ such that $t_i \sqsubseteq |e_i'|$. But then $e \rightarrow^* c(e_1, \ldots, e_n) \rightarrow^* c(e_1', \ldots, e_n')$ with $t \equiv c(t_1, \ldots, t_n) \sqsubseteq c(|e_1'|, \ldots, |e_n'|) \equiv |c(e_1', \ldots, e_n')|$.

A.4 For Section 4

Proof (Prop. 5). The $\Rightarrow$-implication is trivial. For the $\Leftarrow$-implication, assume $[\![e]\!]_{S'} = [\![e']\!]_{S'}$ and let $st \equiv \{est_1, \ldots, est_n\} \in [\![e]\!]_S$. By definition of $[\![e]\!]_{S'}$, each $est_i \in [\![e]\!]_{S'}$, and therefore each $est_i \in [\![e']\!]_{S'}$, which means that there exist $st_i \in [\![e']\!]_S$ such that $est_i \in st_i$. But now, since $\{est_i\} \subseteq st_i$, we know from polarity (Prop. 1 that $\{est_i\} \in [\![e']\!]_S$, for each i, and polarity implies also $st \equiv \{est_1, \ldots, est_n\} \in [\![e']\!]_S$. We have then proved that $[\![e]\!]_S \subseteq [\![e']\!]_S$. The other inclusion holds similarly, arriving to the desired $[\![e]\!]_S = [\![e']\!]_S$.

Proof (Prop. 6).

(a) Full abstraction of $[\![\_]\!]$ wrt $\mathcal{O}_t$ and wrt $\mathcal{O}_{t_\bot}$ means, respectively:

$$[\![e]\!] = [\![e']\!] \Leftrightarrow \forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e'])$$

and

$$[\![e]\!] = [\![e']\!] \Leftrightarrow \forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_{t_\bot}^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}_{t_\bot}^{\mathcal{P}'}(\mathcal{C}[e'])$$

where $\mathcal{P}'$ range over safe extensions, and $\mathcal{C}$ over $\mathcal{P}'$-contexts. Therefore, we must simply prove:

$$\forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e']) \Leftrightarrow \forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_{t_\bot}^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}_{t_\bot}^{\mathcal{P}'}(\mathcal{C}[e'])$$

The $\Leftarrow$-implication is obvious. For the reverse implication $\Rightarrow$, assume

$$\forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e'])$$

We will prove $\forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_{t_\bot}^{\mathcal{P}'}(\mathcal{C}[e]) \subseteq \mathcal{O}_{t_\bot}^{\mathcal{P}'}(\mathcal{C}[e'])$ (the other inclusion holds similarly). Assume $t \in \mathcal{O}_{t_\bot}^{\mathcal{P}'}(\mathcal{C}[e])$, which means $\mathcal{C}[e] \rightarrow^* e'$ for some e' with $t \sqsubseteq |e'|$. Let t' the result of substituting fresh variables for each occurrence of $\bot$ in $|e'|$, and t'' the result of substituting a new constant *bot* for each occurrence of $\bot$ in $|e'|$. We introduce a new function symbol f_t defined by the rule $f_t(t') \rightarrow t''$. We have $f_t(\mathcal{C}[e]) \rightarrow^* t''$, which means $t'' \in \mathcal{O}_t^{\mathcal{P}''}(f_t(\mathcal{C}[e]))$, where $\mathcal{P}''$ is the safe extension of $\mathcal{P}'$ made with *bot*, f_t. By hypothesis, $t'' \in \mathcal{O}_t^{\mathcal{P}''}(f_t(\mathcal{C}[e']))$, which means $f_t(\mathcal{C}[e']) \rightarrow^* e''$ for some e'' with $t'' \sqsubseteq |e''|$. As t'' is total by construction, it must be $t'' = |e''| = e''$, and therefore $f_t(\mathcal{C}[e']) \rightarrow^* t''$. Using the rewrite rule of f_t, it is not difficult to see that there must exist e''' with $\mathcal{C}[e'] \rightarrow^* e'''$ and $|e'| \sqsubseteq |e'''|$. But then $t \sqsubseteq |e'''|$, and therefore $t \in \mathcal{O}_{t_\bot}^{\mathcal{P}'}(\mathcal{C}[e'])$, as desired.

(b) The proof of equivalence of full abstraction wrt $\mathcal{O}_t$ and $\mathcal{O}_{t_\perp}$ proceeds as in *(a)*. It remains to prove that $[\![\_]\!]$ is fully abstract wrt $\mathcal{O}_t \Leftrightarrow$ is fully abstract wrt $\mathcal{O}_{true}$, for which we must see that

$$\forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e']) \Leftrightarrow \forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_{true}^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}_{true}^{\mathcal{P}'}(\mathcal{C}[e'])$$

where in this case $\mathcal{C}[e], \mathcal{C}[e']$ are restricted to be ground. The $\Rightarrow$-implication is obvious. For the reverse implication $\Leftarrow$, assume $\forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_{true}^{\mathcal{P}'}(\mathcal{C}[e]) = \mathcal{O}_{true}^{\mathcal{P}'}(\mathcal{C}[e'])$. We will prove $\forall \mathcal{P}', \mathcal{C}.\ \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e]) \subseteq \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e'])$ (the other inclusion holds similarly). Assume $t \in \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e])$, which means $\mathcal{C}[e] \rightarrow^* t$. Notice that, since $\mathcal{C}[e]$ is ground, t must also be ground. We introduce a new function symbol f_t defined by the rule $f_t(t) \rightarrow true$. We have $f_t(\mathcal{C}[e]) \rightarrow^* true$, which means $true \in \mathcal{O}_{true}^{\mathcal{P}''}(f_t(\mathcal{C}[e]))$, where $\mathcal{P}''$ is the safe extension of $\mathcal{P}'$ made with f_t. By hypothesis, $true \in \mathcal{O}_{true}^{\mathcal{P}''}(f_t(\mathcal{C}[e']))$, which means $f_t(\mathcal{C}[e']) \rightarrow^* true$. But, looking at the rewrite rule of f_t, this can only happen if $\mathcal{C}[e'] \rightarrow^* t$, that is, $t \in \mathcal{O}_t^{\mathcal{P}'}(\mathcal{C}[e']$. Notice how the groundness hypothesis is used: if t is not ground, the condition $f_t(\mathcal{C}[e']) \rightarrow^* true$ does not imply $\mathcal{C}[e'] \rightarrow^* t$, but only $\mathcal{C}[e'] \rightarrow^* t'$ for some instance t' of t.

Proof (Lemma 5). We assume a given $\mathcal{P}$. and mentions to P, P' are omitted in the notations below, since they can be deduced by the context. We reason by induction on the structure of st, noting first that, by definition of $[\![\_]\!]_S$ and Lemma 3, $st \in [\![e]\!]_S \Leftrightarrow st \in [\![\tilde{e}]\!] \Leftrightarrow e \succcurlyeq st$. Thus, the lemma can be reformulated as: $e \succcurlyeq st \Leftrightarrow f_{st}(e) \rightarrow^*_{\mathcal{P}'} \widehat{st}$

- $st \equiv \emptyset$: this case is obvious since $\emptyset \in [\![e]\!]_S, \forall e$ and $f_\emptyset(e) \rightarrow \langle\ \rangle_0 \equiv \widehat{\emptyset}, \forall e$.
- $st \equiv \{X\}$: then $e \succcurlyeq \{X\} \Leftrightarrow e \rightarrow^* X \Leftrightarrow f_{\{X\}}(e) \rightarrow^* X \equiv \widehat{\{X\}}$.
- $st \equiv \{c(st_1, \ldots, st_n)\}$: then $e \succcurlyeq \{c(st_1, \ldots, st_n)\} \Leftrightarrow \exists e_1, \ldots e_n . e \rightarrow^* c(e_1, \ldots, e_n)$ with $e_i \succcurlyeq st_i$ for each i. By IH, $e_i \succcurlyeq st_i \Leftrightarrow f_{st_i}(e_i) \rightarrow^* \widehat{st_i}$. On the other hand, looking at the program rule for $f_{\{c(st_1,\ldots,st_n)\}}$, we have also that $f_{\{c(st_1,\ldots,st_n)\}}(e) \rightarrow^* \widehat{c(st_1, \ldots, st_n)} (\equiv c(\widehat{st_1}, \ldots, \widehat{st_n})) \Leftrightarrow \exists e_1, \ldots e_n . e \rightarrow^* c(e_1, \ldots, e_n)$ with $f_{st_i}(e_i) \rightarrow^* \widehat{st_i}$.
- $st \equiv \{est_1, \ldots, st_n)\}$: then $e \succcurlyeq \{est_1, \ldots, st_n)\} \Leftrightarrow e \succcurlyeq \{est_i\} \forall i \Leftrightarrow_{IH} f_{\{est_i\}}(e) \rightarrow^* \widehat{\{est_i\}} \forall i$ (†). We must see that this condition is equivalent to $f_{\{est_1,\ldots,st_n)\}}(e) \rightarrow^* \{\{\widehat{est_1, \ldots, st_n)}\}\} \equiv \langle \widehat{\{est_1\}}, \ldots, \widehat{\{est_n\}} \rangle_n$ (‡). That † $\Rightarrow$ ‡ is clear, just starting by the step $f_{\{est_1,\ldots,st_n)\}}(e) \rightarrow \langle f_{\{est_1\}}(e), \ldots, f_{\{est_n\}}(e) \rangle_n$ and using † in each component of the tuple. For ‡ $\Rightarrow$ †, notice that any rewrite sequence $f_{\{est_1,\ldots,st_n)\}}(e) \rightarrow^* \{\{\widehat{est_1, \ldots, est_n)}\}\} \equiv \langle \widehat{\{est_1\}}, \ldots, \widehat{\{est_n\}} \rangle_n$ must start with the form

$$f_{\{est_1,\ldots,st_n)\}}(e) \rightarrow^* f_{\{est_1,\ldots,st_n)\}}(e') \rightarrow \langle f_{\{est_1\}}(e'), \ldots, f_{\{est_n\}}(e') \rangle_n \rightarrow^* \langle \widehat{\{est_1\}}, \ldots, \widehat{\{est_n\}} \rangle_n$$

where $e \rightarrow *e'$ and $f_{\{est_i\}}(e') \rightarrow^* \widehat{\{est_i\}}$ for each i. But this leads to $f_{\{est_i\}}(e) \rightarrow^* \widehat{\{est_i\}}$ for each i, which is precisely (†).