Final version in *Journal of Functional and Logic Programming*,
Volume 2001, No. 2, EAPLS, 2001

An Evaluation Semantics for Narrowing-Based Functional Logic Languages

Michael Hanus* Salvador Lucas†

Abstract

We introduce a semantic characterization of narrowing, the computational engine of many functional logic languages. We use a functional domain for giving a denotation to the narrowing space associated to a given initial expression under an arbitrary narrowing strategy. Such a semantic description highlights (and favours) the operational notion of evaluation instead of the more usual model-theoretic notion of interpretation as the basis for the semantic description. The motivation is to obtain an abstract semantics which encodes information about the real operational framework used by a given (narrowing-based) functional logic language. Our aim is to provide a semantic foundation for the development of a general, suitable, and accurate framework for the analysis of functional logic programs.

Keywords: domain theory, functional logic languages, narrowing, program analysis, semantics.

1 Introduction

The ability of reasoning about program properties is essential in software design, implementations, and program manipulation. Program analysis is the task of producing (usually approximated) information about a program. The analysis of functional logic programs is one of the most challenging problems in declarative programming. Many works have already addressed the analysis of certain run-time properties of programs, e.g., mode inferencing [DW89, Han94b, HZ94, Lin88], demandedness patterns [MKMWH93, Zar97], equational unsatisfiability [AFM95, AFRV93, AFV96, BEØ93], detection of parallelism [HKL92, SR92] and a number of other properties which are also relevant for parallel execution

*Institut für Informatik, Christian-Albrechts-Universität Kiel, Olshausenstr. 40, D-24098 Kiel, Germany, mh@informatik.uni-kiel.de. Partially supported by the German Research Council (DFG) under grant Ha 2457/1-1.

†DSIC, UPV, Camino de Vera s/n, E-46071 Valencia, Spain, slucas@dsic.upv.es. Partially supported by EEC-HCM grant ERBCHRXCT940624, Spanish CICYT grant TIC 98-0445-C03-01, and Acción Integrada hispano–alemana HA1997-0073.

[KMH92]. Nevertheless, most of these approaches have been done in a rather ad hoc setting, gearing the analysis towards the application on hand. Up to now, there is no general approach for formulating and analyzing arbitrary properties of functional logic programs with respect to an arbitrary operational framework. Moreover, no attempt to formally connect (or use) properties from the pure logic or functional world in an integrated, functional logic setting has been made. In this paper we address these problems.

The key of our approach is domain theory [Sco82, Sco81, Sco70] since it provides a junction between semantics (spaces of points = denotations of *computational processes*) and logics (lattices of properties of processes) [Abr91, Rey75, Sco81, Vic89]. The computational process we are interested in is evaluation. In a programming language, the notion of *evaluation* emphasizes the idea that there exists a distinguished set of syntactic elements (the values) which have a predefined mathematical interpretation [Gun92, Pit97]. The other syntactic elements take meaning from the program definitions *and* the operational framework for the program's execution. In this way, the evaluation process (under a given operational framework), maps general input expressions (having an *a priori* unknown meaning) to values. This point of view favours the operational notion of evaluation instead of the more usual model-theoretic notion of interpretation as the basis for the semantic description.

Since functional logic languages with a complete operational semantics are based on narrowing, we center our attention on it. The idea of using narrowing as an *evaluation mechanism* for integrated languages comes from Reddy [Red85]: narrowing is the operational principle which computes the *non-ground value* (*ngv*) of an input expression. Given a domain D, a *ngv* is a *mapping* from valuations (on D) to values (in D). In moving valuations from being parameters of semantic functions (as usual in many approaches, e.g., [GHLR99, MR92]) to be components of a semantic domain, we understand narrowing as an *evaluation* mechanism which incorporates the instantiation of variables as a part of such evaluation mechanism. Since *ngv*'s are functional values, we use the domain-theoretic notion of *approximable mapping* [Sco82, Sco81] to give them a computable representation. We argue that this is a good starting point for expressing and managing *observable* properties of functional logic programs (along the lines of [Abr91, Smy83, Vic89]). Moreover, it reveals that, within an integrated framework, there exist semantic connections between purely functional and logic properties of programs. Termination and groundness are examples of such related properties. On the other hand, thanks to including operational information into the semantic description, we are able to derive interesting optimizations for program execution.

Section 2 gives some preliminary definitions. Section 3 introduces the main guidelines of our semantic approach with a simple application to the semantic description of rewriting computations and rewriting strategies. Section 4 discusses the description of narrowing as an evaluation mechanism and introduces approximable mappings. Section 5 formalizes the description of narrowing computations and narrowing strategies by using approximable mappings. Section 6 discusses how much operational information can be obtained back from our semantic descriptions of narrowing and rewriting. Section 7 discusses a semantic-based analysis framework for functional logic languages. Section 8 contains our conclusions.

2 Preliminaries

In this section, we give some preliminary definitions. For further details, we refer the reader to [DP90, GTWW77, Klo92, SLG94]. Given sets A, B, B^A (or $A \to B$) is the set of mappings from A to B and $\mathcal{P}(A)$ denotes the set of all subsets of A. A *preorder* on a set A is a reflexive and transitive relation on A. An *order* $\sqsubseteq$ on a set A is an anti-symmetric preorder on A. Given an ordered set $(A, \sqsubseteq)$, a chain is a (possibly infinite) sequence $a_1, \ldots, a_n, \ldots$ of elements $a_i \in A$, $i \geq 1$ such that, for all $i \geq 1$, $a_i \sqsubseteq a_{i+1}$. An element $\perp$ of an ordered set $(A, \sqsubseteq)$ is called a *least element* (or a *minimum*) if $\perp \sqsubseteq a$ for all $a \in A$. If such an element exists, then $(A, \sqsubseteq, \perp)$ is called a *pointed ordered set*. Given $S \subseteq A$, an element $a \in A$ is an *upper bound* of S if $x \sqsubseteq a$ for all $x \in S$. In this case we also say that S is a *consistent set*. An upper bound of S is a *least upper bound* (or *lub*, written $\bigsqcup S$) if, for all upper bounds b of S, we have $\bigsqcup S \sqsubseteq b$. A set $S \subseteq A$ is downward (upward) closed if whenever $a \in S$ and $b \sqsubseteq a$ ($a \sqsubseteq b$), we have that $b \in S$. If $S = \{x, y\}$, we write $x \sqcup y$ instead of $\bigsqcup S$. A non-empty set $S \subseteq A$ is *directed* if, for all $a, b \in S$, there is an upper bound $c \in S$ of $\{a, b\}$. An ideal is a downward closed, directed set and $Id(A)$ is the set of ideals of an ordered set A. For each $a \in A$, the set $a{\downarrow} = \{b \in A \mid b \sqsubseteq a\}$ is an ideal: the *principal ideal* generated by a. A pointed ordered set $(A, \sqsubseteq, \perp)$ is a *complete partial order* (*cpo*) if every directed set $S \subseteq A$ has a *lub* $\bigsqcup S \in A$. An element $a \in A$ of a *cpo* is called *compact* (or *finite*) if, whenever $S \subseteq A$ is a directed set and $a \sqsubseteq \bigsqcup S$, then there is $x \in S$ such that $a \sqsubseteq x$. The set of compact elements of a cpo A is denoted as $K(A)$. A cpo A is *algebraic* if for each $a \in A$, the set $\mathsf{approx}(a) = \{x \in K(A) \mid x \sqsubseteq a\}$ is directed and $a = \bigsqcup \mathsf{approx}(a)$. An algebraic cpo D is a *domain* if, whenever the set $\{x, y\} \subseteq K(D)$ is consistent, then $x \sqcup y$ exists in D.

Given sets A, B, C, D such that $B \subseteq C$, mappings $f : A \to B$ and $g : C \to D$ are composed as usual to yield a mapping $g \circ f : A \to D$. Given ordered sets $(A, \sqsubseteq_A)$, $(B, \sqsubseteq_B)$, a mapping $f : A \to B$ is monotone if $\forall a, b \in A$, $a \sqsubseteq_A b \Rightarrow f(a) \sqsubseteq_B f(b)$; $f : A \to A$ is idempotent if $\forall a \in A, f(f(a)) = f(a)$; it is decreasing if $\forall a \in A, f(a) \sqsubseteq_A a$. If $(A, \sqsubseteq_A)$, $(B, \sqsubseteq_B)$ are cpo's, we say that $f : A \to B$ is continuous if, for all directed set S, $f(\bigsqcup S) = \bigsqcup f(S)$; the set of continuous (strict) mappings from A to B is denoted by $[A \to B]$ (resp. $[A \to_\perp B]$).

By V we denote a countable set of *variables*; Σ denotes a *signature*, i.e., a set of *function symbols* $\{\mathtt{f}, \mathtt{g}, \ldots\}$, each with a fixed *arity* given by a function $ar : \Sigma \to \mathbb{N}$. We assume $\Sigma \cap V = \varnothing$. We denote by $\mathcal{T}(\Sigma, V)$ the set of (finite) terms built from symbols in the signature Σ and variables in V. A k-tuple $t_1, \ldots, t_k$ of terms is denoted as $\overline{t}$, where k will be clarified from the context.

Given a term t, $\mathcal{V}ar(t)$ is the set of variable symbols in t. Sometimes, we consider a fresh constant $\bot$ and $\Sigma_\bot = \Sigma \cup \{\bot\}$. Terms from $\mathcal{T}(\Sigma_\bot, V)$ are ordered by the usual *approximation ordering* which is the least ordering $\sqsubseteq$ satisfying $\bot \sqsubseteq t$ for all t and $f(\bar{t}) \sqsubseteq f(\bar{s})$ if $\bar{t} \sqsubseteq \bar{s}$, i.e., $t_i \sqsubseteq s_i$ for all $1 \leq i \leq ar(f)$.

Terms are viewed as labeled trees in the usual way. *Positions* $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. By Λ, we denote the empty chain. The set of positions of a term t is denoted by $\mathcal{P}os(t)$. A *linear term* is a term having no multiple occurrences of the same variable. The subterm of t at position p is denoted by $t|_p$. The set of positions of non-variable symbols in t is $\mathcal{P}os_\Sigma(t)$, and $\mathcal{P}os_V(t)$ is the set of variable positions. We denote by $t[s]_p$ the term t with the subterm at the position p replaced by s.

A *substitution* is a mapping $\sigma : V \to \mathcal{T}(\Sigma, V)$ which homomorphically extends to a mapping $\sigma : \mathcal{T}(\Sigma, V) \to \mathcal{T}(\Sigma, V)$. We denote by ε the "identity" substitution: $\varepsilon(x) = x$ for all $x \in V$. The set $\mathcal{D}om(\sigma) = \{x \in V \mid \sigma(x) \neq x\}$ is called the *domain* of σ and $\mathcal{R}ng(\sigma) = \cup_{x \in \mathcal{D}om(\sigma)} \mathcal{V}ar(\sigma(x))$ its *range*. $\sigma_{|U}$ denotes the *restriction* of a substitution σ to a subset of variables $U \subseteq V$. A substitution σ is idempotent if (and only if) $\mathcal{D}om(\sigma) \cap \mathcal{R}ng(\sigma) = \varnothing$. We write $\sigma \leq \sigma'$ if there is θ such that $\sigma' = \theta \circ \sigma$. A *unifier* of two terms t_1, t_2 is a substitution σ with $\sigma(t_1) = \sigma(t_2)$. A *most general unifier* (*mgu*) of t_1, t_2 is a unifier σ with $\sigma \leq \sigma'$ for all other unifiers σ' of t_1, t_2.

A *rewrite rule* (labeled α) is an ordered pair (l, r), written $\alpha : l \to r$ (or just $l \to r$), with $l, r \in \mathcal{T}(\Sigma, V)$, $l \notin V$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. l and r are called *left-hand side* (*lhs*) and *right-hand side* (*rhs*) of the rule, respectively. A *term rewriting system* (*TRS*) is a pair $\mathcal{R} = (\Sigma, R)$ where R is a set of rewrite rules. A TRS (Σ, R) is *left-linear*, if for all $l \to r \in R$, l is a linear term. Given $\mathcal{R} = (\Sigma, R)$, we consider Σ as the disjoint union $\Sigma = \mathcal{C} \uplus \mathcal{F}$ of symbols $c \in \mathcal{C}$, called *constructors* and symbols $f \in \mathcal{F}$, called *defined functions*, where $\mathcal{F} = \{f \mid f(\bar{l}) \to r \in R\}$ and $\mathcal{C} = \Sigma - \mathcal{F}$. A *constructor-based* TRS (*CB-TRS*) is a TRS with $l_1, \ldots, l_n \in \mathcal{T}(\mathcal{C}, V)$ for all rules $f(l_1, \ldots, l_n) \to r$.

For a given TRS $\mathcal{R} = (\Sigma, R)$, a term t *rewrites* to a term s (at position p), written $t \overset{[p,\alpha]}{\to}_{\mathcal{R}} s$ (or just $t \overset{p}{\to}_{\mathcal{R}} s$, $t \to_{\mathcal{R}} s$, or $t \to s$) if $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$, for some rule $\alpha : l \to r \in R$, position $p \in \mathcal{P}os(t)$ and substitution σ. A term t is in *normal form* if there is no term s with $t \to_{\mathcal{R}} s$. A TRS $\mathcal{R}$ (or the rewrite relation $\to_{\mathcal{R}}$) is called *confluent* if for all terms t, t_1, t_2 with $t \to^*_{\mathcal{R}} t_1$ and $t \to^*_{\mathcal{R}} t_2$, there exists a term t_3 with $t_1 \to^*_{\mathcal{R}} t_3$ and $t_2 \to^*_{\mathcal{R}} t_3$. A term t *narrows* to a term s, written $t \leadsto_{[p,\alpha,\theta]} s$ (or just $t \leadsto_\theta s$), if $p \in \mathcal{P}os_\Sigma(t)$ and there is a substitution[1] $\theta : \mathcal{V}ar(t) \to \mathcal{T}(\Sigma, V - \mathcal{V}ar(t))$ such that $\theta(t) \overset{[p,\alpha]}{\to}_{\mathcal{R}} s$. A narrowing derivation $t \leadsto^*_\sigma s$ is such that either $t = s$ and $\sigma = \varepsilon_{|\mathcal{V}ar(t)}$ or $t \leadsto_{\theta_0} t_1 \leadsto_{\theta_1} \cdots t_{n-1} \leadsto_{\theta_{n-1}} s$ and $\sigma = (\theta_{n-1} \circ \cdots \circ \theta_1 \circ \theta_0)_{|\mathcal{V}ar(t)}$ (i.e., we consider only the substitution of goal variables). As usual, we consider different new variables in each elementary narrowing step (i.e., the rules are always 'renamed apart').

[1]This substitution is usually a unifier for $t|_p$ and the left-hand side of the applied rule *restricted to the variables in t*. Note that we do not impose the use of *most general unifiers* for defining the narrowing steps.

3 The semantic approach

Following [Red85], a functional logic program is *functional in syntax and logic in semantics.* A (first-order) program $\mathcal{P} = (\mathcal{R}, t)$ consists of a TRS $\mathcal{R}$ (which establishes the distinction between constructor and defined symbols of the program), and an initial expression t to be *fully* evaluated. We make t explicit since the differences between the purely functional and functional logic styles arise in the different status of the variables occurring in the initial expression: in functional programming, those variables are not allowed (or they are considered as constants and cannot be instantiated). Functional logic languages deal with expressions having *logic* variables and narrowing provides for the necessary instantiations.

We take the following perspective: from the programmer's point of view, the *observed* semantics of the program actually depends on the current operational framework. In this setting, the notion of evaluation, rather than that of interpretation, becomes principal. Since only constructors are completely free from either rewriting or narrowing computations, we assume that only constructor symbols express completely defined information. Alternatively, one could say that only constructor symbols are *definitively observable* during a computation.

We characterize the information which is *currently* couched by a term s (which is supposed to be in an intermediate stage towards the full evaluation of the initial expression t) by means of a monotone, idempotent and decreasing mapping $(\!|\ |\!)$ from syntactic objects to values (remind that values are expected to be especial syntactic objects). We call $(\!|\ |\!)$ an *observation mapping.* Monotonicity of $(\!|\ |\!)$ ensures that refinements (w.r.t. $\sqsubseteq$) of the syntactic information correspond to refinements of the observed semantic information. Idempotency ensures that each observation is definitive. Decreasingness ensures that the semantic information is part of the syntactic information[2]. The adequacy of a given mapping $(\!|\ |\!)$ for observing computations performed by a given operational mechanism should be ensured by showing that $(\!|\ |\!)$ is a homomorphism between the relation among syntactic objects induced by the operational mechanism and the approximation ordering on values. This means that the operational mechanism refines the meaning of an expression as the computation continues.

3.1 Rewriting as an evaluation mechanism

The syntactic objects are terms $t \in \mathcal{T}(\Sigma_\perp, V)$ and the values are taken from $(\mathcal{T}^\infty(\mathcal{C}_\perp), \sqsubseteq, \perp)$, the domain of infinite, ground constructor (partial) terms. Formally, $(\mathcal{T}^\infty(\mathcal{C}_\perp), \sqsubseteq, \perp)$ is obtained from $\mathcal{T}(\mathcal{C}_\perp)$, which is not even a cpo, as (isomorphic to) its ideal completion $(Id(\mathcal{T}(\mathcal{C}_\perp)), \subseteq, \{\perp\})$ (see [DP90, SLG94]). In general, given a poset P, the mapping $[\,\cdot\,] : P \to Id(P)$ that associates the principal ideal $p\!\downarrow$ to each $p \in P$ is an embedding of P into the *cpo* $Id(P)$, i.e., for all $p, q \in P$, $p \sqsubseteq q$ if and only if $[p] \subseteq [q]$. Since $[\,\cdot\,]$ is injective, we can understand $Id(P)$ as a completion of P which actually 'includes' P. $(\mathcal{T}^\infty(\mathcal{C}_\perp, V), \sqsubseteq, \perp)$ is

[2] Strictness of $(\!|\ |\!)$ is a consequence of decreasingness.

the domain $(\mathcal{T}^\infty(\mathcal{C}_\perp \cup V), \sqsubseteq, \perp)$, where $ar(x) = 0$ for all $x \in V$.

For functional computations, we use $(\!|\ |\!)_F : \mathcal{T}(\Sigma_\perp, V) \to \mathcal{T}(\mathcal{C}_\perp, V)$ given by

$$\begin{array}{rclcrcll} (\!|x|\!)_F & = & x & & (\!|\perp|\!)_F & = & \perp & \\ (\!|c(\overline{t})|\!)_F & = & c((\!|\overline{t}|\!)_F) & \text{if } c \in \mathcal{C} & (\!|f(\overline{t})|\!)_F & = & \perp & \text{if } f \in \mathcal{F} \end{array}$$

Clearly, $(\!|\ |\!)_F$ is an observation mapping. The adequacy of this mapping for observing rewriting computations is stated in the following proposition which establishes that rewriting increases the current information of terms as given by $(\!|\ |\!)_F$.

Proposition 3.1 (Reduction increases information) *Let $\mathcal{R}$ be a TRS and $t, s \in \mathcal{T}(\Sigma_\perp, V)$. If $t \to^* s$, then $(\!|t|\!)_F \sqsubseteq (\!|s|\!)_F$.*

PROOF. By induction on the length n of the derivation $t \to^* s$. The case $n = 0$ is immediate. Otherwise, let $t \to t' \to^* s$. To prove that $t \to t'$ implies $(\!|t|\!)_F \sqsubseteq (\!|t'|\!)_F$, we proceed by induction on the length of the redex position $p \in \mathcal{P}os(t)$ of the first rewrite step. If $p = \Lambda$, then $t = \sigma(l) = f(\overline{t})$ for some rule $l \to r$ and defined symbol $f \in \mathcal{F}$ (because $l = f(\overline{l})$). Hence, $(\!|t|\!)_F = \perp \sqsubseteq (\!|t'|\!)_F$. If $p \neq \Lambda$, we have $p = i \cdot p'$. Then, $t = f(\overline{t})$, $t_i \to t'_i$, and $t_j = t'_j$ for all $1 \leq j \leq ar(f)$, $i \neq j$. If $f \in \mathcal{F}$, then $(\!|t|\!)_F = \perp \sqsubseteq (\!|t'|\!)_F$. If $f \in \mathcal{C}$, then $(\!|t|\!)_F = c((\!|\overline{t}|\!)_F)$ and, since $t \sqsubseteq t'$, $t' = c(\overline{t'})$. Therefore, by I.H., $(\!|t_i|\!)_F \sqsubseteq (\!|t'_i|\!)_F$ and $(\!|t_j|\!)_F = (\!|t'_j|\!)_F$ for all $1 \leq j \leq ar(f)$, $i \neq j$. Hence, by definition of $\sqsubseteq$, $(\!|t|\!)_F \sqsubseteq (\!|t'|\!)_F$. By (the first) I.H., $(\!|t'|\!)_F \sqsubseteq (\!|s|\!)_F$. Thus, the conclusion follows. $\square$

The function $Rew : \mathcal{T}(\Sigma_\perp, V) \to \mathcal{P}(\mathcal{T}(\mathcal{C}_\perp, V))$ provides a representation $Rew(t) = \{(\!|s|\!)_F \mid t \to^* s\}$ of the rewriting space of a given term t.

Proposition 3.2 *Let $\mathcal{R}$ be a confluent TRS. For all $t \in \mathcal{T}(\Sigma_\perp, V)$, $(Rew(t), \sqsubseteq)$ is a directed set.*

PROOF. Note that $Rew(t) \neq \varnothing$ because $(\!|t|\!)_F \in Rew(t)$. If $(\!|t'|\!)_F, (\!|t''|\!)_F \in Rew(t)$, then $t \to^* t'$ and $t \to^* t''$. By confluence, there exists a term s such that $t' \to^* s$ and $t'' \to^* s$. Hence, $t \to^* s$, and $(\!|s|\!)_F \in Rew(t)$. By Proposition 3.1, $(\!|t'|\!)_F \sqsubseteq (\!|s|\!)_F$ and $(\!|t''|\!)_F \sqsubseteq (\!|s|\!)_F$, i.e., $Rew(t)$ is directed. $\square$

The semantic function

$$CRew^\infty : \mathcal{T}(\Sigma_\perp, V) \to \mathcal{T}^\infty(\mathcal{C}_\perp, V)$$

gives the meaning of a term under evaluation by rewriting (for confluent TRSs):

$$CRew^\infty(t) = \bigsqcup Rew(t)$$

or even

$$CRew^\infty(t) = Rew(t){\downarrow}$$

in an equivalent expression which takes advantage of the correspondence between 'infinite terms' and ideals of finite terms (note that $Rew(t)\downarrow$ is an ideal). Thus, $CRew^{\infty}(t)$ is the most defined (possibly infinite) value which can be obtained (or approximated) by issuing rewritings from t. Note that we follow the convention of pursuing the *total evaluation* (infinite normalization) of the term and that $CRew^{\infty}$ is well defined for *confluent* TRS's; otherwise, we cannot ensure that $Rew(t)$ is a directed set and the *lub* may not exist. We also note that the use of infinite terms in the codomain of $CRew^{\infty}$ is necessary for dealing with non-terminating programs.

3.2 Rewriting strategies

For a *rewriting strategy* $\mathbb{F}$ (i.e., a mapping from terms to terms satisfying $\mathbb{F}(t) = t$ whenever t is a normal form and $t \to \mathbb{F}(t)$ otherwise [Klo92]), we define $Rew_{\mathbb{F}}(t) = \{(\!|\mathbb{F}^n(t)|\!)_F \mid n \geq 0\}$.

Proposition 3.3 *Let $\mathcal{R}$ be a TRS and $\mathbb{F}$ be a rewriting strategy for $\mathcal{R}$. For all $t \in \mathcal{T}(\Sigma_\perp, V)$, $Rew_{\mathbb{F}}(t)$ is a chain.*[3]

Proof. Immediate by Proposition 3.1. □

Thus, we define

$$CRew_{\mathbb{F}}^{\infty} : \mathcal{T}(\Sigma_\perp, V) \to \mathcal{T}^{\infty}(\mathcal{C}_\perp, V)$$

by

$$CRew_{\mathbb{F}}^{\infty}(t) = \bigsqcup Rew_{\mathbb{F}}(t)$$

Clearly, for all strategies $\mathbb{F}$, $CRew_{\mathbb{F}}^{\infty} \sqsubseteq CRew^{\infty}$ (i.e., $CRew_{\mathbb{F}}^{\infty}(t) \sqsubseteq CRew^{\infty}(t)$ $\forall t$). Thus, $CRew^{\infty}$ provides a semantic reference for rewriting strategies. Strategies that satisfy $CRew_{\mathbb{F}}^{\infty} = CRew^{\infty}$ can be thought of as correct strategies. They correspond to *infinitary normalizing* strategies—if we restrict our attention to computing (infinite) values rather than arbitrary (infinite) normal forms. It is possible to provide an effective notion of infinitary normalizing strategy by using Middeldorp's theory of root-needed computations [Mid97] and their decidable approximations [Luc98].

Remark 3.4 *We obtain a ground semantics for the defined symbols $f \in \mathcal{F}$ as follows: $f(\overline{\delta}) = CRew^{\infty}(f(\overline{\delta}))$ for all $\overline{\delta} \in \mathcal{T}(\mathcal{C}_\perp)^{ar(f)}$. Similarly, it is possible to describe a ground semantics under a given strategy $\mathbb{F}$ by using $CRew_{\mathbb{F}}^{\infty}$.*

4 Narrowing as an evaluation mechanism

Through its computed value $CRew^{\infty}(t)$, a *ground* term t denotes a value $[\![t]\!]_D$ in some domain D by just giving an interpretation for each constructor symbol c as a continuous function $c_D \in [D^{ar(c)} \to D]$: $[\![t]\!]_D = [\![CRew^{\infty}(t)]\!]_D$. However,

[3] Formally, $Rew_{\mathbb{F}}(t)$ is defined as a set but for the purpose of this proposition we identify it with a sequence.

our main interest are terms with variables. In this case, the most reasonable choice is to interpret a term as denoting a function. This definition is the natural one: a term with variables t denotes a continuous function $t_D \in [D^{\mathcal{V}ar(t)} \to D]$ which yields the value of t under each possible valuation $\phi \in D^{\mathcal{V}ar(t)}$ of its variables on a domain D. This is called a *non-ground value* (*ngv*) in [Red85] and a *derived operator* in [GTW78, GTWW77]. It is also essentially the same as in other algebraic approaches to semantics of TRS's and recursive program schemes such as [Bou85, Cou90, Gue81, Niv75].

Given domains D and E, the set $[D \to E]$ ($[D \to_\perp E]$) of (strict) continuous functions from D to E (pointwise) ordered by $f \sqsubseteq g$ iff $\forall x \in D,\ f(x) \sqsubseteq g(x)$, is a domain [Gun92, SLG94]. Given a set $W \subseteq V$ of variables, for proving that $[D^W \to D]$ is a domain whenever D is, we note that $W_\perp = W \uplus \{\perp\}$ supplied by the least ordering $\sqsubseteq$ such that $\perp \sqsubseteq x$ and $x \sqsubseteq x$ for all $x \in W$ is a domain (the flat domain of associated to the set W). The set D^W of arbitrary valuations from W to D is isomorphic to the domain $[W_\perp \to_\perp D]$ of continuous, strict mappings from $W_\perp$ to D. Thus, we can view D^W as this domain. In particular, if we take $\mathcal{T}^\infty(\mathcal{C}_\perp)$ as the domain D of values, then $\mathcal{T}^\infty(\mathcal{C}_\perp)^W$ is a domain whose least element is the mapping $\lambda x \in W.\perp$ (denoted $\perp^W_{Valuat}$). By abuse, we say that the *domain of a valuation* $\phi \in D^W$ is

$$\mathcal{D}om(\phi) = \{x \in W \mid \phi(x) \neq \perp\} \ .$$

Therefore, if D is a domain, $[D^V \to D]$ also is and, in particular, $[\mathcal{T}^\infty(\mathcal{C}_\perp)^V \to \mathcal{T}^\infty(\mathcal{C}_\perp)]$ is a domain. We write $\perp_{Valuat}$ instead of $\perp^V_{Valuat}$. Given a term t, $[\mathcal{T}^\infty(\mathcal{C}_\perp)^{\mathcal{V}ar(t)} \to \mathcal{T}^\infty(\mathcal{C}_\perp)]$ is also a domain whose least element is denoted $\perp^{\mathcal{V}ar(t)}_{Valuat}$.

4.1 Observation of narrowing computations

Our syntactic objects, now, are substitution/term pairs $\langle \sigma, t\rangle$: Given a term $t \in \mathcal{T}(\Sigma_\perp, V)$ a narrowing derivation

$$t = t_0 \leadsto_{\theta_0} t_1 \leadsto_{\theta_1} \cdots \leadsto_{\theta_{n-2}} t_{n-1} \leadsto_{\theta_{n-1}} t_n = s$$

is represented as

$$\langle \sigma_0, t_0\rangle \leadsto \langle \sigma_1, t_1\rangle \leadsto \cdots \leadsto \langle \sigma_{n-1}, t_{n-1}\rangle \leadsto \langle \sigma_n, t_n\rangle$$

where $\sigma_0 = \varepsilon_{|\mathcal{V}ar(t)}$ and $\sigma_{i+1} = (\theta_i \circ \sigma_i)_{|\mathcal{V}ar(t)}$ for $0 \leq i < n$. We also eventually write $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ instead, where $\sigma = \sigma_n$.

Note that, since we restrict our attention to instantiations of variables in $\mathcal{V}ar(t)$, we have that $\sigma_i : \mathcal{V}ar(t) \to \mathcal{T}(\Sigma, V)$ and $\mathcal{D}om(\sigma_i) \subseteq \mathcal{V}ar(t)$ for $i \geq 0$. Moreover, $\mathcal{D}om(\sigma_i) \subseteq \mathcal{D}om(\sigma_j)$ and $\sigma_i \leq \sigma_j$ whenever $i \leq j$.

Remark 4.1 *Since we use idempotent substitutions for performing the elementary narrowing steps, we have that* $\mathcal{D}om(\sigma_i) \cap \mathcal{R}ng(\sigma_i) = \varnothing$ *for* $i \geq 0$.

In order to *observe* the narrowing computations, we could naïvely extend $(\!|\ |\!)_F$ to deal with substitution/term pairs: $(\!|\langle\sigma,s\rangle|\!)_F = \langle(\!|\sigma|\!)_F,(\!|s|\!)_F\rangle$ where $(\!|\sigma|\!)_F$ is a substitution given by $(\!|\sigma|\!)_F(x) = (\!|\sigma(x)|\!)_F$ for all $x \in V$. Unfortunately, the semantic progress of a narrowing evaluation might not be captured by the computational ordering $\sqsubseteq$ (extended to pairs by $(\varsigma,\delta) \sqsubseteq (\varsigma',\delta')$ iff $\forall x \in V. \varsigma(x) \sqsubseteq \varsigma'(x)$ and $\delta \sqsubseteq \delta'$) and such an extension of $(\!|\ |\!)_F$.

Example 4.2 *Consider the TRS:*

$$\begin{array}{rcl@{\qquad}lcl}
\texttt{0+x} & \to & \texttt{x} & \texttt{0} \ \leq \ \texttt{x} & \to & \texttt{true}\\
\texttt{s(x)+y} & \to & \texttt{s(x+y)} & \texttt{s(x)} \ \leq \ \texttt{s(y)} & \to & \texttt{x} \ \leq \ \texttt{y}
\end{array}$$

and the narrowing step

$$\langle\varepsilon, \texttt{[x,x+y]}\rangle \leadsto \langle\{\texttt{x}\mapsto\texttt{0}\}, \texttt{[0,y]}\rangle$$

(where $\texttt{[}\cdot\texttt{,}\cdot\texttt{]}$ *denotes a 2-element list). We have*

$$(\!|\langle\varepsilon, \texttt{[x,x+y]}\rangle|\!)_F = \langle\varepsilon, \texttt{[x,}\bot\texttt{]}\rangle$$

and

$$(\!|\langle\{\texttt{x}\mapsto\texttt{0}\}, \texttt{[0,y]}\rangle|\!)_F = \langle\{\texttt{x}\mapsto\texttt{0}\}, \texttt{[0,y]}\rangle.$$

Therefore, we do not get the desired increasing computation, because $\varepsilon \not\sqsubseteq \{\texttt{x}\mapsto\texttt{0}\}$ *and* $\texttt{[x,}\bot\texttt{]} \not\sqsubseteq \texttt{[0,y]}$.

The problem is that narrowing introduces a new computational mechanism for increasing the information associated to a given term, i.e., instantiation of *logic* variables. Thus, we introduce the observation mapping $(\!|\ |\!)_{FL} : \mathcal{T}(\Sigma_\bot, V) \to \mathcal{T}(\mathcal{C}_\bot)$ which interprets uninstantiated variables as least defined elements:

$$\begin{array}{rcll@{\qquad}rcll}
(\!|x|\!)_{FL} & = & \bot & & (\!|\bot|\!)_{FL} & = & \bot & \\
(\!|c(\overline{t})|\!)_{FL} & = & c((\!|\overline{t}|\!)_{FL}) & \text{if } c \in \mathcal{C} & (\!|f(\overline{t})|\!)_{FL} & = & \bot & \text{if } f \in \mathcal{F}
\end{array}$$

Note that $(\!|t|\!)_{FL} = \bot_{\mathit{Valuat}}((\!|t|\!)_F)$ and $(\!|\sigma|\!)_{FL} = \bot_{\mathit{Valuat}} \circ (\!|\sigma|\!)_F$.

Example 4.3 *(Continuing Example 4.2) Now,*

$$\begin{array}{rcl}
(\!|\langle\varepsilon, \texttt{[x,x+y]}\rangle|\!)_{FL} & = & \langle\bot_{\mathit{Valuat}}, \texttt{[}\bot\texttt{,}\bot\texttt{]}\rangle\\
& \sqsubseteq & \langle\{\texttt{x}\mapsto\texttt{0}\}, \texttt{[0,}\bot\texttt{]}\rangle\\
& = & (\!|\langle\{\texttt{x}\mapsto\texttt{0}\}, \texttt{[0,y]}\rangle|\!)_{FL}
\end{array}$$

i.e., in this case, $(\!|\ |\!)_{FL}$ *satisfies the desired property.*

After introducing some results, we prove that narrowing computations are compatible with the new observation mapping.

Lemma 4.4 *Let* $t, s \in \mathcal{T}(\Sigma_\bot, V)$. *If* $(\!|t|\!)_F \sqsubseteq (\!|s|\!)_F$, *then* $(\!|t|\!)_{FL} \sqsubseteq (\!|s|\!)_{FL}$.

PROOF. Since $(\!|t|\!)_{FL} = \bot_{\mathit{Valuat}}((\!|t|\!)_F)$ and $(\!|s|\!)_{FL} = \bot_{\mathit{Valuat}}((\!|s|\!)_F)$, the conclusion follows by monotonicity of $\bot_{\mathit{Valuat}}$. $\square$

Lemma 4.5 *Let t be a finite term and σ be a substitution. Then $(\!|t|\!)_{FL} \sqsubseteq (\!|\sigma(t)|\!)_{FL}$.*

PROOF. By structural induction. If t is a variable, then $(\!|t|\!)_{FL} = \bot \sqsubseteq (\!|\sigma(t)|\!)_{FL}$. If t is a constant, $t = \sigma(t)$ and the conclusion follows. Let $t = f(\overline{t})$. If $f \in \mathcal{F}$, then $(\!|t|\!)_{FL} = \bot$ and the conclusion follows. If $f = c \in \mathcal{C}$, then $\sigma(c(\overline{t})) = c(\sigma(\overline{t}))$. By I.H., $(\!|t_i|\!)_{FL} \sqsubseteq (\!|\sigma(t_i)|\!)_{FL}$ for all i, $1 \leq i \leq ar(c)$. Therefore, by definition of $\sqsubseteq$, $(\!|c(\overline{t})|\!)_{FL} = c((\!|\overline{t}|\!)_{FL}) \sqsubseteq c((\!|\sigma(\overline{t})|\!)_{FL}) = (\!|c(\sigma(\overline{t}))|\!)_{FL} = (\!|\sigma(c(\overline{t}))|\!)_{FL}$. $\square$

Lemma 4.6 *Let σ, σ' be substitutions. If $\sigma \leq \sigma'$, then $(\!|\sigma|\!)_{FL} \sqsubseteq (\!|\sigma'|\!)_{FL}$.*

PROOF. If $\sigma \leq \sigma'$, there is θ such that $\sigma' = \theta \circ \sigma$. Thus, for all $x \in V$, $\sigma'(x) = \theta(\sigma(x))$. By Lemma 4.5, for all terms t, $(\!|\sigma(t)|\!)_{FL} \sqsubseteq (\!|\theta(\sigma(t))|\!)_{FL} = (\!|\sigma'(t)|\!)_{FL}$, i.e., $(\!|\sigma|\!)_{FL} \sqsubseteq (\!|\sigma'|\!)_{FL}$. $\square$

The following proposition establishes that narrowing increases the current information of substitution/term pairs as given by $(\!|\ |\!)_{FL}$.

Proposition 4.7 *Let $\mathcal{R}$ be a TRS. If $\langle \sigma, t\rangle \leadsto^* \langle \sigma', t'\rangle$, then $(\!|\langle \sigma, t\rangle|\!)_{FL} \sqsubseteq (\!|\langle \sigma', t'\rangle|\!)_{FL}$.*

PROOF. We proceed by induction on the length n of the narrowing derivation. If $n = 0$, it is immediate. If $n > 0$, let $\langle \sigma, t\rangle \leadsto \langle \theta \circ \sigma, s\rangle \leadsto^* \langle \sigma', t'\rangle$ where θ is the substitution used for issuing the narrowing step. By the induction hypothesis, $(\!|\langle \theta \circ \sigma, s\rangle|\!)_{FL} \sqsubseteq (\!|\langle \sigma', t'\rangle|\!)_{FL}$, i.e., $(\!|\theta \circ \sigma|\!)_{FL} \sqsubseteq (\!|\sigma'|\!)_{FL}$ and $(\!|s|\!)_{FL} \sqsubseteq (\!|t'|\!)_{FL}$. Since $\sigma \leq \theta \circ \sigma$, by Lemma 4.6, we have that $(\!|\sigma|\!)_{FL} \sqsubseteq (\!|\theta \circ \sigma|\!)_{FL}$; hence $(\!|\sigma|\!)_{FL} \sqsubseteq (\!|\sigma'|\!)_{FL}$. To prove that $(\!|t|\!)_{FL} \sqsubseteq (\!|s|\!)_{FL}$, we note that, by Lemma 4.5, $(\!|t|\!)_{FL} \sqsubseteq (\!|\theta(t)|\!)_{FL}$. Now, since $\theta(t) \to s$, by Proposition 3.1 and Lemma 4.4, $(\!|\theta(t)|\!)_{FL} \sqsubseteq (\!|s|\!)_{FL}$. $\square$

4.2 Approximable mappings

In the following, we are concerned with the representation of *functional* values. In this setting, we use the corresponding standard Scott's construction of *approximable mappings* [Sco81, SLG94].

A *precusl* is a structure $P = (P, \sqsubseteq, \sqcup, \bot)$ where $\sqsubseteq$ is a preorder, $\bot$ is a distinguished minimal element, and $\sqcup$ is a partial binary operation on P such that, for all $p, q \in P$, $p \sqcup q$ is defined if and only if $\{p, q\}$ is consistent in P and then $p \sqcup q$ is a (distinguished) *lub* of p and q [SLG94]. Approximable mappings allow us to represent arbitrary continuous mappings between domains on the representations of those domains (their compact elements) as relations between approximations of a given argument and approximations of its value at that argument [SLG94].

Definition 4.8 [SLG94] *Let $P = (P, \sqsubseteq, \sqcup, \bot), P' = (P', \sqsubseteq', \sqcup', \bot')$ be precusl's. A relation $f \subseteq P \times P'$ is an approximable mapping from P to P' if*

1. $\bot \; f \; \bot'$.

2. $p\ f\ p'$ *and* $p\ f\ q'$ *imply* $p\ f\ (p' \sqcup q')$.

3. $p\ f\ p'$, $p \sqsubseteq q$, *and* $q' \sqsubseteq' p'$ *imply* $q\ f\ q'$.

The ideal completion $(Id(P), \subseteq, \{\bot\})$ of a precusl P is a domain (see [SLG94]). If $P = (P, \sqsubseteq, \sqcup, \bot)$ is a *cusl*[4] (i.e., $\sqsubseteq$ is actually an ordering), then the mapping $[\,\cdot\,] : P \to Id(P)$ that associates the principal ideal $\{p\}\!\downarrow$ to each $p \in P$ is injective.

An approximable mapping f defines a continuous function $\overline{f} : Id(P) \to Id(P')$ given by [SLG94]

$$\begin{array}{rcl} \overline{f}(I) & = & \{p' \in P' \mid \exists p \in I.p\ f\ p'\} \\ & = & \bigcup_{p \in I}\{p' \in P' \mid p\ f\ p'\} \end{array}$$

Note that, for all $p \in I$, $\{p' \in P' \mid p\ f\ p'\}$ is an ideal (it is not empty because we always have $\bot\ f\ \bot'$, and thus $p\ f\ \bot'$ by following the third condition of Definition 4.8; it is directed due to the second condition of Definition 4.8; finally, it is downward closed because of, whenever we have $p\ f\ p'$ and $q' \sqsubseteq' p'$, we also have $p\ f\ q'$, third condition again).

Proposition 4.9 *Let* $P = (P, \sqsubseteq, \sqcup, \bot), P' = (P', \sqsubseteq', \sqcup', \bot')$ *be precusl's, and* $f, f' \subseteq P \times P'$ *be approximable mappings from* P *to* P'. *If* $f \subseteq f'$, *then* $\overline{f} \sqsubseteq \overline{f'}$.

PROOF. Immediate. $\square$

In the following, we are mainly involved with elements of $Id(P)$ which correspond to elements $p \in P$ via $[\,\cdot\,]$: in our context, P is either $\mathcal{T}(\mathcal{C}_\bot)$ or $\mathcal{T}(\mathcal{C}_\bot)^V$ and elements in P correspond to finite objects (finite values, or valuations mapping variables to finite values, respectively) of $Id(P)$. Thus, we can roughly consider elements of P as the finite or compact elements of $Id(P)$ (via $[\,\cdot\,]$).

Proposition 4.10 *Let* $P = (P, \sqsubseteq, \sqcup, \bot), P' = (P', \sqsubseteq', \sqcup', \bot')$ *be precusl's, and* $f \subseteq P \times P'$ *be an approximable mapping from* P *to* P'. *If* $p \in P$, *then* $\overline{f}([p]) = \{p' \in P' \mid p\ f\ p'\}$.

PROOF. We note that $\bigcup_{q \in [p]}\{p' \in P' \mid q\ f\ p'\} \subseteq \{p' \in P' \mid p\ f\ p'\}$: indeed, since for all $q \in [p]$, we have that $q \sqsubseteq p$, and by using Definition 4.8 (third point), whenever $q\ f\ p'$, we also have $p\ f\ p'$. Thus, since it is obvious that $\{p' \in P' \mid p\ f\ p'\} \subseteq \bigcup_{q \in [p]}\{p' \in P' \mid q\ f\ p'\}$, we write

$$\begin{array}{rcl} \overline{f}([p]) & = & \bigcup_{q \in [p]}\{p' \in P' \mid q\ f\ p'\} \\ & = & \{p' \in P' \mid p\ f\ p'\} \end{array}$$

$\square$

[4] *conditional upper semilattice with least element*, abbreviated *cusl* [SLG94].

Proposition 4.11 *Let $P = (P, \sqsubseteq, \sqcup, \bot), P' = (P', \sqsubseteq', \sqcup', \bot')$ be precusl's, $p \in P$ and $f \subseteq P \times P'$ be an approximable mapping from P to P'. If $[p'] = \overline{f}([p])$ for some $p' \in P'$, then for all $q \in [p]$, whenever $q\ f\ q'$ for some $q' \in P'$, we have that $q' \sqsubseteq' p'$.*

PROOF. Immediate. □

The following proposition establishes that, if $\overline{f}$ sets a connection between finite elements of domains $Id(P)$ and $Id(P')$, then f itself already connects those elements.

Proposition 4.12 *Let $P = (P, \sqsubseteq, \sqcup, \bot), P' = (P', \sqsubseteq', \sqcup', \bot')$ be precusl's, $p \in P$ and $f \subseteq P \times P'$ be an approximable mapping from P to P'. If $[p'] = \overline{f}([p])$ for some $p' \in P'$, then $p\ f\ p'$.*

PROOF. By definition of $\overline{f}$, $[p'] = \{q' \in P' \mid \exists q \in [p].q\ f\ q'\}$. In particular, since $p' \in [p']$, there must be $q \in [p]$ such that $q\ f\ p'$. Since $q \sqsubseteq p$, by Definition 4.8 (third condition), the conclusion follows. □

Proposition 4.13 *Let $P = (P, \sqsubseteq, \sqcup, \bot), P' = (P', \sqsubseteq', \sqcup', \bot')$ be precusl's and $\mathcal{I}$ be a set of indices. Let $f_\alpha \subseteq P \times P'$ be approximable mappings for all $\alpha \in \mathcal{I}$. If $f = \cup_{\alpha \in \mathcal{I}} f_\alpha$ is an approximable mapping, then $\overline{f} = \bigsqcup_{\alpha \in \mathcal{I}} \overline{f}_\alpha$.*

PROOF. For all $I \in Id(P)$, we have:

$$\begin{array}{rcl} (\bigsqcup_{\alpha \in \mathcal{I}} \overline{f}_\alpha)(I) & = & \bigcup_{\alpha \in \mathcal{I}} \overline{f}_\alpha(I) \\ & = & \bigcup_{\alpha \in \mathcal{I}} \{p' \in P' \mid \exists p \in I.\ p\ f_\alpha\ p'\} \\ & = & \{p' \in P' \mid \exists p \in P.\ p\ f\ p'\} \\ & = & \overline{f}(I) \end{array}$$

□

Proposition 4.14 *Let $P = (P, \sqsubseteq, \sqcup, \bot), P' = (P', \sqsubseteq', \sqcup', \bot')$ be precusl's and $\mathcal{I}$ be a set of indices. Let $f_\alpha \subseteq P \times P'$ be approximable mappings for all $\alpha \in \mathcal{I}$ such that $\{\overline{f}_\alpha \mid \alpha \in \mathcal{I}\}$ is bounded. Let $f = \bigsqcup_{\alpha \in \mathcal{I}} \overline{f}_\alpha$ and $p \in P$. If $[p'] = f([p])$ for some $p' \in P'$, then there exists $\alpha \in \mathcal{I}$ such that $[p'] = \overline{f}_\alpha([p])$.*

PROOF. Note that $f([p]) = (\bigsqcup_{\alpha \in \mathcal{I}} \overline{f}_\alpha)([p]) = \bigcup_{\alpha \in \mathcal{I}} \overline{f}_\alpha([p])$. Since $[p'] = \bigcup_{\alpha \in \mathcal{I}} \overline{f}_\alpha([p])$, it follows that $p' \in \bigcup_{\alpha \in \mathcal{I}} \{q' \in P' \mid \exists q \in [p].q\ f_\alpha\ q'\}$, i.e., there exists $\alpha \in \mathcal{I}$ such that $p' \in \{q' \in P' \mid \exists q \in [p].q\ f_\alpha\ q'\} = \overline{f}_\alpha([p])$. If $q' \sqsubseteq p'$, then, by Definition 4.8, we have that, being $q \in [p]$ such that $q\ f_\alpha\ p'$, we also have $q\ f_\alpha\ q'$. Thus, $q' \in \overline{f}_\alpha([p])$, i.e., $[p'] \subseteq \overline{f}_\alpha([p])$. On the other hand, if $[p'] \not\supseteq \overline{f}_\alpha([p])$, it is not possible that $[p'] = f([p])$. Hence, the conclusion follows. □

5 The narrowing space as an approximable mapping

Analogously to the construction $Rew(t)$, we can build a semantic description $Narr(t)$ of the narrowing evaluation of t. Nevertheless, since $Narr(t)$ is intended to be a representation of a *ngv*, i.e., a functional value, we are going to use the approximable mappings introduced in the previous section.

It is easy to see that $(\mathcal{T}(\mathcal{C}_\perp), \sqsubseteq, \sqcup, \perp)$, where $\sqsubseteq$ is the usual approximation ordering, is a *precusl* (in fact a *cusl*). Similarly, given a set $W \subseteq V$ of variables, $(\mathcal{T}(\mathcal{C}_\perp)^W, \sqsubseteq, \sqcup, \perp^W_{Valuat})$, where $\sqsubseteq$ is the pointwise extension of the ordering $\sqsubseteq$ on $\mathcal{T}(\mathcal{C}_\perp)$ to valuations $\phi \in \mathcal{T}(\mathcal{C}_\perp)^W$, is also a *cusl*. Given a term t, $NDeriv(t)$ is the set of narrowing derivations issued from t. We associate a relation $Narr^A(t) \subseteq \mathcal{T}(\mathcal{C}_\perp)^{\mathcal{V}ar(t)} \times \mathcal{T}(\mathcal{C}_\perp)$ to a given narrowing derivation $A \in NDeriv(t)$.

Definition 5.1 *Given a term $t \in \mathcal{T}(\Sigma_\perp, V)$ and a narrowing derivation*

$$A : \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle = \langle \sigma_0, t_0\rangle \rightsquigarrow \langle \sigma_1, t_1\rangle \rightsquigarrow \cdots \rightsquigarrow \langle \sigma_{n-1}, t_{n-1}\rangle \rightsquigarrow \langle \sigma_n, t_n\rangle$$

we define $Narr^A(t) = \cup_{0\leq i\leq n} Narr_i^A(t)$ where:

$$\begin{array}{rcl} Narr_i^A(t) & = & \{\langle \varsigma, \delta\rangle \in \mathcal{T}(\mathcal{C}_\perp)^{\mathcal{V}ar(t)} \times \mathcal{T}(\mathcal{C}_\perp) \mid \\ & & \exists \phi \in \mathcal{T}(\mathcal{C}_\perp)^V . (\!| \phi \circ \sigma_i |\!)_{FL} \sqsubseteq \varsigma \wedge \delta \sqsubseteq (\!| \phi(t_i) |\!)_{FL}\} \end{array}$$

where we assume that $\mathcal{D}om(\phi) \cap \mathcal{D}om(\sigma_i) = \varnothing$ for $0 \leq i \leq n$.

Notice that symbol $\sqsubseteq$ in Definition 5.1 is overloaded since it is used to compare (partial) values in $\mathcal{T}(\mathcal{C}_\perp)$ and valuations (in $\mathcal{T}(\mathcal{C}_\perp)^{\mathcal{V}ar(t)}$) of variables in $\mathcal{V}ar(t)$.

Remark 5.2 *Note that, under our assumptions for the narrowing of terms (see Remark 4.1) condition $\mathcal{D}om(\phi) \cap \mathcal{D}om(\sigma_i) = \varnothing$ of Definition 5.1 is a natural one and does not actually restrict anything: since σ_i is idempotent and $\sigma_i(t_i) = t_i$, variables in $\mathcal{D}om(\sigma_i)$ are not useful for either instantiating t_i or variables in $\mathcal{R}ng(\sigma_i)$.*

Example 5.3 *Consider the TRS $\mathcal{R}$ in Example 4.2 and term $t =$ `x+y`. For the narrowing derivations:*

$$A1: \quad \langle\{\mathtt{x}\mapsto\mathtt{x},\mathtt{y}\mapsto\mathtt{y}\},\mathtt{x+y}\rangle \quad \rightsquigarrow \quad \langle\{\mathtt{x}\mapsto\mathtt{0},\mathtt{y}\mapsto\mathtt{y'}\},\mathtt{y'}\rangle$$

and

$$\begin{array}{rrl} A2: \quad \langle\{\mathtt{x}\mapsto\mathtt{x},\mathtt{y}\mapsto\mathtt{y}\},\mathtt{x+y}\rangle & \rightsquigarrow & \langle\{\mathtt{x}\mapsto\mathtt{s(x')},\mathtt{y}\mapsto\mathtt{y'}\},\mathtt{s(x'+y')}\rangle \\ & \rightsquigarrow & \langle\{\mathtt{x}\mapsto\mathtt{s(0)},\mathtt{y}\mapsto\mathtt{y''}\},\mathtt{s(y'')}\rangle \end{array}$$

we show (part of) their semantic descriptions $Narr^{A1}$ and $Narr^{A2}$:

$$\begin{array}{rcl} Narr_0^{A1}(t) & = & \{\langle\varsigma,\perp\rangle \mid \varsigma \in \mathcal{T}^\infty(\mathcal{C}_\perp)^{\{\mathtt{x},\mathtt{y}\}}\} \\ Narr_1^{A1}(t) & = & \{\langle\{\mathtt{x}\mapsto\mathtt{0},\mathtt{y}\mapsto\perp\},\perp\rangle, \\ & & \langle\{\mathtt{x}\mapsto\mathtt{0},\mathtt{y}\mapsto\mathtt{0}\},\perp\rangle, \langle\{\mathtt{x}\mapsto\mathtt{0},\mathtt{y}\mapsto\mathtt{0}\},\mathtt{0}\rangle, \\ & & \langle\{\mathtt{x}\mapsto\mathtt{0},\mathtt{y}\mapsto\mathtt{s(}\perp\mathtt{)}\},\perp\rangle, \langle\{\mathtt{x}\mapsto\mathtt{0},\mathtt{y}\mapsto\mathtt{s(}\perp\mathtt{)}\},\mathtt{s(}\perp\mathtt{)}\rangle, \\ & & \ldots\} \end{array}$$

$$
\begin{array}{rcl}
Narr_0^{A2}(t) & = & \{\langle\varsigma,\bot\rangle \mid \varsigma\in\mathcal{T}^\infty(\mathcal{C}_\bot)^{\{\mathtt{x},\mathtt{y}\}}\} \\
Narr_1^{A2}(t) & = & \{\langle\{\mathtt{x}\mapsto\mathtt{s}(\bot),\mathtt{y}\mapsto\bot\},\bot\rangle,\langle\{\mathtt{x}\mapsto\mathtt{s}(\bot),\mathtt{y}\mapsto\bot\},\mathtt{s}(\bot)\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\bot\},\bot\rangle,\langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\bot\},\mathtt{s}(\bot)\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(\bot),\mathtt{y}\mapsto 0\},\bot\rangle,\langle\{\mathtt{x}\mapsto\mathtt{s}(\bot),\mathtt{y}\mapsto 0\},\mathtt{s}(\bot)\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto 0\},\bot\rangle,\langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto 0\},\mathtt{s}(\bot)\rangle, \\
& & \ldots\} \\
Narr_2^{A2}(t) & = & \{\langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\bot\},\bot\rangle,\langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\bot\},\mathtt{s}(\bot)\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto 0\},\bot\rangle,\langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto 0\},\mathtt{s}(\bot)\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto 0\},\mathtt{s}(0)\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\mathtt{s}(\bot)\},\bot\rangle,\langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\mathtt{s}(\bot)\},\mathtt{s}(\bot)\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\mathtt{s}(\bot)\},\mathtt{s}(\mathtt{s}(\bot))\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\mathtt{s}(0)\},\bot\rangle,\langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\mathtt{s}(0)\},\mathtt{s}(\bot)\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\mathtt{s}(0)\},\mathtt{s}(\mathtt{s}(\bot))\rangle, \\
& & \langle\{\mathtt{x}\mapsto\mathtt{s}(0),\mathtt{y}\mapsto\mathtt{s}(0)\},\mathtt{s}(\mathtt{s}(0))\rangle,\ldots\}
\end{array}
$$

We can prove that $Narr^A(t)$ is an approximable mapping for every narrowing derivation $A \in NDeriv(t)$. In order to achieve this, we need some lemmata.

Lemma 5.4 *Let $\phi \in \mathcal{T}(\mathcal{C}_\bot)^V$ and σ be a substitution. Then, $(\!|\phi\circ\sigma|\!)_{FL} = \phi\circ(\!|\sigma|\!)_F$.*

PROOF. $(\!|\phi\circ\sigma|\!)_{FL} = \bot_{Valuat}\circ(\!|\phi\circ\sigma|\!)_F = \bot_{Valuat}\circ(\!|\phi|\!)_F\circ(\!|\sigma|\!)_F = \bot_{Valuat}\circ\phi\circ(\!|\sigma|\!)_F = \phi\circ(\!|\sigma|\!)_F$. □

Lemma 5.5 *Let $t\in\mathcal{T}(\Sigma_\bot, V)$. If $\phi\in\mathcal{T}(\mathcal{C}_\bot)^V$, then $(\!|\phi(t)|\!)_{FL} = \phi((\!|t|\!)_F)$.*

PROOF. By using Lemma 5.4, we have $(\!|\phi(t)|\!)_{FL} = (\!|\phi|\!)_{FL}(t) = (\!|\phi\circ\varepsilon|\!)_{FL}(t) = \phi\circ(\!|\varepsilon|\!)_F(t) = \phi((\!|t|\!)_F)$. □

Proposition 5.6 *Let $\phi,\phi',\varsigma\in\mathcal{T}(\mathcal{C}_\bot)^V$ and σ be a substitution such that $\mathcal{D}om(\sigma)\cap\mathcal{D}om(\phi) = \varnothing$. If $\{\phi\circ\sigma,\phi'\circ\sigma\}$ is bounded by ς, then $\{\phi,\phi'\}$ is bounded and $(\phi\sqcup\phi')\circ\sigma\sqsubseteq\varsigma$.*

PROOF. First we prove that $\{\phi,\phi'\}$ is bounded, i.e., that for all $x\in V$, $\{\phi(x),\phi'(x)\}$ is bounded. For each $x\in V$, we consider two cases:

1. $x\notin\mathcal{D}om(\sigma)$, i.e., $\sigma(x) = x$: By hypothesis, we have $\phi(\sigma(x)) = \phi(x)\sqsubseteq\varsigma(x)$ and $\phi'(\sigma(x)) = \phi'(x)\sqsubseteq\varsigma(x)$.
2. $x\in\mathcal{D}om(\sigma)$: Then, by hypothesis, $x\notin\mathcal{D}om(\phi)$, i.e., $\phi(x) = \bot$. Thus, $\phi(x)\sqsubseteq\phi'(x)$.

Thus, since $\mathcal{T}(\mathcal{C}_\bot)^V$ is a cusl, $\phi\sqcup\phi'$ does exist. Now we have $(\phi\sqcup\phi')\circ\sigma = (\phi\circ\sigma)\sqcup(\phi'\circ\sigma)\sqsubseteq\varsigma$. □

Example 5.7 *Without imposing that $\mathcal{D}om(\sigma)\cap\mathcal{D}om(\phi) = \varnothing$, Proposition 5.6 could be false. For instance, let $\sigma(x) = a$, $\varsigma(x) = a$, $\phi(x) = a$, and $\phi'(x) = b$ for a given variable x and arbitrary constants a and b. Then, $\phi(\sigma(x)) = \phi'(\sigma(x)) = \varsigma(x) = a$, i.e., $\{\phi\circ\sigma,\phi'\circ\sigma\}$ is bounded by ς, but $\{\phi(x),\phi'(x)\} = \{a,b\}$ is not bounded.*

Lemma 5.8 *Let t be a finite term and σ, σ' be substitutions such that $\sigma \sqsubseteq \sigma'$. Then, $(\!|\sigma(t)|\!)_{FL} \sqsubseteq (\!|\sigma'(t)|\!)_{FL}$.*

Proof. Since $\sigma \sqsubseteq \sigma'$, $\sigma(t) \sqsubseteq \sigma'(t)$ for all terms t. The conclusion follows by monotonicity of $(\!|\ |\!)_{FL}$. □

Proposition 5.9 *Let $\mathcal{R}$ be a TRS, t be a term, and A be a narrowing derivation starting from t. Then, $Narr^A(t)$ is an approximable mapping.*

Proof. Let

$$A : \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle = \langle \sigma_0, t_0\rangle \rightsquigarrow \langle \sigma_1, t_1\rangle \rightsquigarrow \cdots \rightsquigarrow \langle \sigma_{n-1}, t_{n-1}\rangle \rightsquigarrow \langle \sigma_n, t_n\rangle$$

We abbreviate $Narr^A(t)$ by m. Then, we check the conditions of Definition 4.8. We silently use Lemma 5.4 to simplify the expressions.

1. Note that, for all derivations A starting from t,

   $$Narr_0^A(t) = \{\langle \varsigma, \delta\rangle \mid \exists \phi \in \mathcal{T}(\mathcal{C}_\perp)^V . (\!|\phi \circ \varepsilon_{|\mathcal{V}ar(t)}|\!)_{FL} \sqsubseteq \varsigma \wedge \delta \sqsubseteq (\!|\phi(t)|\!)_{FL}\} \subseteq m.$$

   We have that $(\!|\phi \circ \varepsilon_{|\mathcal{V}ar(t)}|\!)_{FL} = \phi \circ (\!|\varepsilon_{|\mathcal{V}ar(t)}|\!)_F = \phi \circ \varepsilon_{|\mathcal{V}ar(t)} = \phi_{|\mathcal{V}ar(t)}$. In particular, by choosing $\phi = \perp_{Valuat}$, $\varsigma = \perp_{Valuat}^{\mathcal{V}ar(t)}$ (note that $\mathcal{D}om(\perp_{Valuat}) = \varnothing$), and $\delta = \perp$, we obtain $\phi_{|\mathcal{V}ar(t)} = (\perp_{Valuat})_{|\mathcal{V}ar(t)} \sqsubseteq \perp_{Valuat}^{\mathcal{V}ar(t)}$, and $\delta = \perp \sqsubseteq (\!|\perp_{Valuat}(t)|\!)_{FL}$, i.e., $\perp_{Valuat}^{\mathcal{V}ar(t)}\ m\ \perp$.

2. Let $\varsigma\ m\ \delta$ and $\varsigma\ m\ \delta'$. By definition of m, there are $\phi_i, \phi_j \in \mathcal{T}(\mathcal{C}_\perp)^V$ such that $\phi_i \circ (\!|\sigma_i|\!)_F \sqsubseteq \varsigma$ and $\phi_j \circ (\!|\sigma_j|\!)_F \sqsubseteq \varsigma$ for some $0 \leq i < j \leq n$. Since $i < j$, there exists an idempotent substitution $\theta : \mathcal{V}ar(t_i) \to \mathcal{T}(\Sigma, V)$ such that $\sigma_j = \theta \circ \sigma_i$ (here the assumption about the usual variable renaming of rules of the TRS when applying narrowing steps is important, see [Pal90]). Therefore, we have that $\phi_i \circ (\!|\sigma_i|\!)_F \sqsubseteq \varsigma$ and $\phi_j \circ (\!|\sigma_j|\!)_F = \phi_j \circ (\!|\theta \circ \sigma_i|\!)_F \sqsubseteq \varsigma$. Let us show that $\phi_i \circ (\!|\sigma_j|\!)_F \sqsubseteq \varsigma$. Let $x \in \mathcal{V}ar(t)$ be such that $x \notin \mathcal{D}om((\!|\sigma_j|\!)_F)$. Since $\mathcal{D}om((\!|\sigma_i|\!)_F) \subseteq \mathcal{D}om((\!|\sigma_j|\!)_F)$, it follows that $x \notin \mathcal{D}om((\!|\sigma_i|\!)_F)$; thus, $\phi_i((\!|\sigma_j(x)|\!)_F) = \phi_i(x) = \phi_i((\!|\sigma_i(x)|\!)_F) \sqsubseteq \varsigma(x)$. If $x \in \mathcal{D}om((\!|\sigma_j|\!)_F)$, then, by using the fact that $\sigma_j = \theta \circ \sigma_i$, we distinguish two cases:

   (a) If $\mathcal{V}ar((\!|\sigma_i(x)|\!)_F) \cap \mathcal{D}om(\theta) = \varnothing$, then $(\!|\sigma_i(x)|\!)_F = (\!|\sigma_j(x)|\!)_F$; hence, $\phi_i((\!|\sigma_j(x)|\!)_F) \sqsubseteq \varsigma(x)$.

   (b) If $\mathcal{V}ar((\!|\sigma_i(x)|\!)_F) \cap \mathcal{D}om(\theta) \neq \varnothing$, then $(\!|\sigma_j(x)|\!)_F = \theta((\!|\sigma_i(x)|\!)_F)$ can contain variables which are already present in $(\!|\sigma_i(x)|\!)_F$ (i.e., variables y with $y \notin \mathcal{D}om(\theta)$) for which condition $\phi_i \circ (\!|\sigma_i|\!)_F \sqsubseteq \varsigma$ ensures the desired result. For the other variables, we can assume, w.l.o.g., that ϕ_i does not essentially modify anything (i.e., we can assume that $\mathcal{R}ng(\theta) \cap \mathcal{D}om(\phi_i) = \varnothing$). Hence, the condition $\perp_{Valuat} \circ (\!|\sigma_j|\!)_F \sqsubseteq \varsigma$ (an easy consequence of $\phi_j \circ (\!|\sigma_j|\!)_F \sqsubseteq \varsigma$) ensures the desired result.

Thus, since $\phi_i \circ (\!|\sigma_j|\!)_F \sqsubseteq \varsigma$ and $\phi_j \circ (\!|\sigma_j|\!)_F \sqsubseteq \varsigma$, by Proposition 5.6 (note that $\mathcal{D}om((\!|\sigma_j|\!)_F) \cap \mathcal{D}om(\phi_j) = \varnothing$), $\{\phi_i, \phi_j\}$ is bounded by $\phi = \phi_i \sqcup \phi_j$ and $\phi \circ (\!|\sigma_j|\!)_F \sqsubseteq \varsigma$. By definition of m, we also have $\delta \sqsubseteq (\!|\phi_i(t_i)|\!)_{FL}$ and $\delta' \sqsubseteq (\!|\phi_j(t_j)|\!)_{FL}$. By Proposition 4.7, $(\!|t_i|\!)_{FL} \sqsubseteq (\!|t_j|\!)_{FL}$. By using the fact that both $\phi_i \circ (\!|\sigma_i|\!)_F$ and $\phi_i \circ (\!|\sigma_j|\!)_F$ are bounded by ς, we conclude that $(\!|\phi_i(t_i)|\!)_{FL} \sqsubseteq (\!|\phi_i(t_j)|\!)_{FL}$. By Lemma 5.8, we obtain

$$\delta \sqsubseteq (\!|\phi_i(t_i)|\!)_{FL} \sqsubseteq (\!|\phi_i(t_j)|\!)_{FL} \sqsubseteq (\!|\phi(t_j)|\!)_{FL}$$

By Lemma 5.8 again,

$$\delta' \sqsubseteq (\!|\phi_j(t_j)|\!)_{FL} \sqsubseteq (\!|\phi(t_j)|\!)_{FL}$$

Thus, $\{\delta, \delta'\}$ is bounded, and $\delta \sqcup \delta' \sqsubseteq (\!|\phi(t_j)|\!)_{FL}$. Since $\phi \circ (\!|\sigma_j|\!)_F \sqsubseteq \varsigma$, by definition of m, we have $\varsigma\ m\ (\delta \sqcup \delta')$.

3. Let $\varsigma\ m\ \delta$, $\varsigma \sqsubseteq \varsigma'$, and $\delta' \sqsubseteq \delta$. Thus, there is $\phi \in \mathcal{T}(\mathcal{C}_\perp)^V$ and σ_i, $0 \le i \le n$ such that $\phi \circ (\!|\sigma_i|\!)_F \sqsubseteq \varsigma$ and $\delta \sqsubseteq (\!|\phi(t_i)|\!)_{FL}$. Since $\phi \circ (\!|\sigma_i|\!)_F \sqsubseteq \varsigma \sqsubseteq \varsigma'$ and $\delta' \sqsubseteq \delta \sqsubseteq (\!|\phi(t_i)|\!)_{FL}$, $\varsigma'\ m\ \delta'$ holds by definition of m.

□

Definition 5.10 *Given a term $t \in \mathcal{T}(\Sigma_\perp, V)$, we define the relation $Narr(t) \subseteq \mathcal{T}(\mathcal{C}_\perp)^{\mathcal{V}ar(t)} \times \mathcal{T}(\mathcal{C}_\perp)$ to be $Narr(t) = \bigcup_{A \in NDeriv(t)} Narr^A(t)$.*

Unfortunately, these semantic definitions are *not* consistent w.r.t. rewriting.

Example 5.11 *Consider the TRS:*

```
f(f(x)) → a
c       → b
```

and $A : \langle \varepsilon_{|\{\mathtt{x}\}}, t\rangle = \langle \{\mathtt{x} \mapsto \mathtt{x}\}, \mathtt{f(x)}\rangle \leadsto \langle \{\mathtt{x} \mapsto \mathtt{f(x')}\}, \mathtt{a}\rangle$. If $m = Narr^A(t)$, then $\{\mathtt{x} \mapsto \mathtt{a}\}\ m\ \mathtt{a}$ (we take $\phi = \perp_{Valuat}$, $\sigma = \{\mathtt{x} \mapsto \mathtt{f(x')}\}$ in Definition 5.1; hence, $(\!|\phi \circ \sigma|\!)_{FL} = \perp^{\mathcal{V}ar(t)}_{Valuat} \sqsubseteq \{\mathtt{x} \mapsto \mathtt{a}\} = \varsigma$). Thus, $\overline{Narr^A(t)}(\{\mathtt{x} \mapsto \mathtt{a}\}) = \mathtt{a}$. However, $\{\mathtt{x} \mapsto \mathtt{a}\}(t) = \mathtt{f(a)} \not\to^* \mathtt{a}$.

The problem here is that $(\!|\ |\!)_{FL}$ identifies (as $\perp$) parts of the bindings $\sigma(x)$ of a computed substitution σ which can be semantically refined by instantiation (of the variables in $\sigma(x)$) and other which cannot be further refined by instantiation (the operation-rooted subterms in $\sigma(x)$). If we deal with left-linear CB-TRS's and choose (idempotent) *mgu*'s as unifiers for the narrowing process, the substitutions which we deal with are *linear* constructor substitutions, i.e., for all narrowing derivations $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ and all $x \in \mathcal{V}ar(t)$, $\sigma(x)$ is a constructor term and $\{\!\{\sigma(x) \mid x \in \mathcal{D}om(\sigma)\}\!\}$ is a linear multiset of terms (i.e., no variable appears twice within them). Hence, the substitutions computed by narrowing have *no* partial information apart from the variable occurrences. In this case, $(\!|\sigma|\!)_F = \sigma$, $(\!|\sigma|\!)_{FL} = \perp_{Valuat} \circ (\!|\sigma|\!)_F = \perp_{Valuat} \circ \sigma$, and we have the following result.

Proposition 5.12 *Let σ be a linear constructor substitution and $\phi, \varsigma \in \mathcal{T}(\mathcal{C}_\perp)^V$ be such that $\mathcal{D}om(\varsigma) \cap \mathcal{R}ng(\sigma) = \varnothing$. If $\phi \circ \sigma \sqsubseteq \varsigma$, then there exists $\phi' \in \mathcal{T}(\mathcal{C}_\perp)^V$ such that $\phi \sqsubseteq \phi'$ and $\phi' \circ \sigma = \varsigma$.*

PROOF. Let $x \in V$. Since $\phi \circ \sigma \sqsubseteq \varsigma$, we have that $\phi(\sigma(x)) \sqsubseteq \varsigma(x)$. We consider two cases:

1. If $\sigma(x) \neq x$ (i.e., $x \in \mathcal{D}om(\sigma)$), then we distinguish, again, two cases:
   (a) If $\mathcal{V}ar(\sigma(x)) = \varnothing$, then, since σ is a constructor substitution, we have that $\sigma(x) \in \mathcal{T}(\mathcal{C})$. Hence, $\phi(\sigma(x)) = \sigma(x) \sqsubseteq \varsigma(x)$ which, in fact, means that $\sigma(x) = \varsigma(x)$. Thus, for all $\phi' \in \mathcal{T}(\mathcal{C}_\perp)^V$, $\phi'(\sigma(x)) = \sigma(x) = \varsigma(x)$.
   (b) If $\mathcal{V}ar(\sigma(x)) \neq \varnothing$, then there exists a context $C[\,]$ such that $\phi(\sigma(x)) = C[\phi(y_1), \ldots, \phi(y_n)] \sqsubseteq C[\delta_1, \ldots, \delta_n] = \varsigma(x)$ where $C[\ ]$ is just $\sigma(x)$ where variable occurrences have been replaced by $\Box$; this is correct because σ is a constructor substitution (and hence $\sigma(x)$ is a constructor term without $\perp$'s) and σ is linear (thus there are n different variables $y_1, \ldots, y_n$ in $\mathcal{V}ar(\sigma(x))$ whose bindings $\phi(y_1), \ldots, \phi(y_n)$ can be independently established). Thus, we let $\phi'(y_i) = \delta_i$ for $1 \leq i \leq n$; we can safely do this because of condition $\mathcal{D}om(\varsigma) \cap \mathcal{R}ng(\sigma) = \varnothing$ ensures that no further collisions would arise between these assignments for ϕ' and those given in case 2 below.

2. If $\sigma(x) = x$ (i.e., $x \notin \mathcal{D}om(\sigma)$), then we can just let $\phi'(x) = \varsigma(x)$.

$\Box$

Note that linearity of σ is necessary for ensuring this result.

Example 5.13 *Let $\sigma = \{\mathtt{u} \mapsto \mathtt{f(x,y)}, \mathtt{v} \mapsto \mathtt{f(x,z)}\}$, $\phi = \perp_{Valuat}$, and $\varsigma = \{\mathtt{u} \mapsto \mathtt{f(\perp,\perp)}, \mathtt{v} \mapsto \mathtt{f(c,\perp)}\}$. Clearly, $\phi \circ \sigma = \{\mathtt{u} \mapsto \mathtt{f(\perp,\perp)}, \mathtt{v} \mapsto \mathtt{f(\perp,\perp)}\} \sqsubseteq \varsigma$. However, there is no ϕ' such that $\phi' \circ \sigma = \varsigma$ because it would be necessary that, simultaneously, $\phi'(\mathtt{x}) = \perp$ and $\phi'(\mathtt{x}) = \mathtt{c}$.*

Moreover, the condition $\mathcal{D}om(\varsigma) \cap \mathcal{R}ng(\sigma) = \varnothing$ is also necessary for ensuring the result[5].

Example 5.14 *Let $\sigma = \{\mathtt{x} \mapsto \mathtt{s(y)}, \mathtt{y} \mapsto \mathtt{y}, \mathtt{z} \mapsto \mathtt{z}\}$, $\phi = \perp_{Valuat}$, and $\varsigma = \{\mathtt{x} \mapsto \mathtt{s(a)}, \mathtt{y} \mapsto \mathtt{b}, \mathtt{z} \mapsto \mathtt{b}\}$. Note that $\mathcal{D}om(\varsigma) \cap \mathcal{R}ng(\sigma) = \{\mathtt{y}\}$. Clearly, $\phi \circ \sigma = \{\mathtt{x} \mapsto \mathtt{s(\perp)}, \mathtt{y} \mapsto \perp, \mathtt{z} \mapsto \perp\} \sqsubseteq \varsigma$. However, there is no ϕ' such that $\phi' \circ \sigma = \varsigma$ because it would be necessary that, simultaneously, $\phi'(\mathtt{y}) = \mathtt{a}$ and $\phi'(\mathtt{y}) = \mathtt{b}$.*

Thus, we obtain a simpler, more readable expression for the approximable mapping which is associated to a given left-linear, CB-TRS by noting that

$$\begin{aligned} Narr_i^A(t) &= \{\langle \varsigma, \delta \rangle \mid \exists \phi \in \mathcal{T}(\mathcal{C}_\perp)^V . (\!|\phi \circ \sigma_i|\!)_{FL} \sqsubseteq \varsigma \wedge \delta \sqsubseteq (\!|\phi(t_i)|\!)_{FL}\} \\ &= \{\langle \varsigma, \delta \rangle \mid \exists \phi \in \mathcal{T}(\mathcal{C}_\perp)^V . \phi \circ \sigma_i = \varsigma \wedge \delta \sqsubseteq (\!|\phi(t_i)|\!)_{FL}\} \end{aligned}$$

[5]Example 5.14 was suggested by a referee.

This is easily proved correct when considering the special properties of (partial) computed substitutions σ_i within a narrowing derivation, specially the fact that the hypotheses for applying Proposition 5.12 are easily fulfilled.

The union of approximable mappings (considered as binary relations) need not to be an approximable mapping. Nevertheless, we have the following result.

Proposition 5.15 *Let $\mathcal{R}$ be a left-linear, confluent CB-TRS and t be a term. Then, $Narr(t)$ is an approximable mapping.*

PROOF. We abbreviate $Narr(t)$ by m. Then, we check the conditions of Definition 4.8. Again, we use Lemma 5.4 to simplify the expressions.

1. Since $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \rightsquigarrow^* \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle$, we have that $\perp_{Valuat} \circ \varepsilon_{|\mathcal{V}ar(t)} = \perp^{\mathcal{V}ar(t)}_{Valuat} \sqsubseteq \perp^{\mathcal{V}ar(t)}_{Valuat}$, and $\perp \sqsubseteq (\!|\perp^{\mathcal{V}ar(t)}_{Valuat}(t)|\!)_{FL}$, i.e., $\perp^{\mathcal{V}ar(t)}_{Valuat}\ m\ \perp$.

2. Let $\varsigma\ m\ \delta$ and $\varsigma\ m\ \delta'$. By definition of m, there are narrowing derivations $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \rightsquigarrow^* \langle \sigma_1, s_1\rangle$, $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \rightsquigarrow^* \langle \sigma_2, s_2\rangle$ and $\phi_1, \phi_2 \in \mathcal{T}(\mathcal{C}_\perp)^V$ such that $\phi_1 \circ (\!|\sigma_1|\!)_F \sqsubseteq \varsigma$ and $\phi_2 \circ (\!|\sigma_2|\!)_F \sqsubseteq \varsigma$. By Proposition 5.12, there exist $\theta_1, \theta_2 \in \mathcal{T}(\mathcal{C}_\perp)^V$ such that $\phi_1 \sqsubseteq \theta_1$, $\phi_2 \sqsubseteq \theta_2$, and $\varsigma = \theta_1 \circ (\!|\sigma_1|\!)_F$, $\varsigma = \theta_2 \circ (\!|\sigma_2|\!)_F$. We also have $\delta \sqsubseteq (\!|\phi_1(s_1)|\!)_{FL}$ and $\delta' \sqsubseteq (\!|\phi_2(s_2)|\!)_{FL}$.

   By Hullot's Theorem[6] [Hul80], $\sigma_1(t) \to^* s_1$ and $\sigma_2(t) \to^* s_2$. By stability, we have that $\theta_1(\sigma_1(t)) \to^* \theta_1(s_1)$ and $\theta_2(\sigma_2(t)) \to^* \theta_2(s_2)$. Since $\mathcal{R}$ is left-linear and constructor-based, σ_1 and σ_2 are constructor substitutions. Therefore, $(\!|\sigma_1|\!)_F = \sigma_1$, and $(\!|\sigma_2|\!)_F = \sigma_2$, and hence $\varsigma = \theta_1 \circ \sigma_1$ and $\varsigma = \theta_2 \circ \sigma_2$. Thus, $\varsigma(t) \to^* \theta_1(s_1)$ and $\varsigma(t) \to^* \theta_2(s_2)$. By confluence, there is a term s such that $\theta_1(s_1) \to^* s$ and $\theta_2(s_2) \to^* s$, hence $\varsigma(t) \to^* s$. By Proposition 3.1 and Lemma 4.4, $(\!|\theta_1(s_1)|\!)_{FL}, (\!|\theta_2(s_2)|\!)_{FL} \sqsubseteq (\!|s|\!)_{FL}$. By Hullot's Theorem[7] [Hul80], there is $\sigma \leq \varsigma$ such that $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \rightsquigarrow^* \langle \sigma, s'\rangle$ and $s' \leq s$, i.e., there exists a substitution ϕ such that $\varsigma = \phi \circ \sigma$ and $s = \phi(s')$. By hypothesis and by Lemma 5.8, $\delta \sqsubseteq (\!|\phi_1(s_1)|\!)_{FL} \sqsubseteq (\!|\theta_1(s_1)|\!)_{FL}$ and $\delta' \sqsubseteq (\!|\phi_2(s_2)|\!)_{FL} \sqsubseteq (\!|\theta_2(s_2)|\!)_{FL}$. Since $(\!|\theta_1(s_1)|\!)_{FL}, (\!|\theta_2(s_2)|\!)_{FL} \sqsubseteq (\!|s|\!)_{FL}$, it follows that $\{\delta, \delta'\}$ is bounded by $(\!|s|\!)_{FL}$, i.e., $\{\delta, \delta'\}$ is consistent. Since $\mathcal{T}(\mathcal{C}_\perp)$ is a *cusl*, $\delta \sqcup \delta'$ is the *lub* of δ and δ'. Hence, since $(\!|\phi \circ \sigma|\!)_{FL} = (\!|\varsigma|\!)_{FL} = \varsigma \sqsubseteq \varsigma$, and $\delta \sqcup \delta' \sqsubseteq (\!|s|\!)_{FL} = (\!|\phi(s')|\!)_{FL}$, by Definition 5.10, $\varsigma\ m\ (\delta \sqcup \delta')$.

3. We need to prove that, if $\varsigma\ m\ \delta$, $\varsigma \sqsubseteq \varsigma'$, and $\delta' \sqsubseteq \delta$, then then $\varsigma'\ m\ \delta'$. Since $\varsigma\ m\ \delta$, there is a narrowing derivation $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \rightsquigarrow^* \langle \sigma, s\rangle$ and a substitution $\phi \in \mathcal{T}(\mathcal{C}_\perp)^V$ such that $\phi \circ (\!|\sigma|\!)_F \sqsubseteq \varsigma$ and $\delta \sqsubseteq (\!|\phi(s)|\!)_{FL}$. If $\varsigma \sqsubseteq \varsigma'$, then $\phi \circ (\!|\sigma|\!)_F \sqsubseteq \varsigma'$. On the other hand, $\delta' \sqsubseteq \delta \sqsubseteq (\!|\phi(s)|\!)_{FL}$. Hence, by definition of m, $\varsigma'\ m\ \delta'$.

[6]In principle, Hullot's Theorem is valid for narrowing with *mgu*'s. However, Padawitz has shown [Pad88, Prop. 8.2.2] that this theorem also holds for narrowing with arbitrary unifiers.

[7]This application of Hullot's Theorem generally yields a narrowing derivation where only *mgu*'s are used; however, our result is valid for those specializations of narrowing that admit a similar completeness result (e.g., narrowing with arbitrary unifiers [Pad88] or needed narrowing [AEH00]).

□

We have the following compositionality result: the semantics of the whole narrowing process can be thought of as the *lub* of the semantics of each narrowing derivation.

Proposition 5.16 *Let $\mathcal{R}$ be a left-linear, confluent CB-TRS and t be a term. Then $\overline{Narr(t)} = \bigsqcup_{A \in NDeriv(t)} \overline{Narr^A(t)}$.*

Proof. Proposition 5.15, Proposition 5.9, and Proposition 4.13. □

Thus, we define the semantic function

$$CNarr^{\infty} : \mathcal{T}(\Sigma_{\perp}, V) \to [\mathcal{T}^{\infty}(\mathcal{C}_{\perp})^V \to \mathcal{T}^{\infty}(\mathcal{C}_{\perp})]$$

as follows:

$$CNarr^{\infty}(t) = \overline{Narr(t)}$$

i.e., $CNarr^{\infty}(t)$ is the continuous mapping associated to the approximable mapping $Narr(t)$ which represents the narrowing derivations starting from t. This semantics is consistent w.r.t. rewriting.

Theorem 5.17 *Let $\mathcal{R}$ be a left-linear, confluent CB-TRS. For all $t \in \mathcal{T}(\Sigma_{\perp}, V)$, $\varsigma \in \mathcal{T}(\mathcal{C}_{\perp})^V$, $CNarr^{\infty}(t)\ \varsigma = CRew^{\infty}(\varsigma(t))$.*

Proof. By using Proposition 4.10 (and according to Proposition 5.15), we can write:

$$\begin{array}{rcl} CNarr^{\infty}(t)\ [\varsigma] & = & \{\delta \mid \varsigma\ Narr(t)\ \delta\} \\ & = & \bigcup_{A \in NDeriv(t)} \{\delta \mid \varsigma\ Narr^A(t)\ \delta\} \end{array}$$

For each narrowing derivation

$$A : \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle = \langle \sigma_0, t_0\rangle \leadsto \langle \sigma_1, t_1\rangle \leadsto \cdots \leadsto \langle \sigma_{n-1}, t_{n-1}\rangle \leadsto \langle \sigma_n, t_n\rangle$$

such that $\varsigma = \phi \circ \sigma_i$ for some $1 \leq i \leq n$ and $\delta \sqsubseteq (\!|\phi(t_i)|\!)_{FL}$, by Hullot's Theorem, we have $\sigma_i(t) \to^* t_i$. By stability $\varsigma(t) \to^* \phi(t_i)$. Thus, since $\phi \in \mathcal{T}(\mathcal{C}_{\perp})^V$, we have that $(\!|\phi(t_i)|\!)_{FL} = (\!|\phi(t_i)|\!)_F \in Rew(\varsigma(t))$ and, in fact, $CNarr^{\infty}(t)\ [\varsigma] \subseteq Rew(\varsigma(t))\!\downarrow$. In order to prove that $Rew(\varsigma(t))\!\downarrow\ \subseteq CNarr^{\infty}(t)\ [\varsigma]$, let us consider $\delta \in Rew(\varsigma(t))\!\downarrow$. Then there exists $(\!|s|\!)_F \in Rew(\varsigma(t))$ such that $\delta \sqsubseteq (\!|s|\!)_F$. Hence, $\varsigma(t) \to^* s$ and there is a narrowing derivation $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s'\rangle$ with $\varsigma = \phi \circ \sigma$ for some $\phi \in \mathcal{T}(\mathcal{C}_{\perp})^V$ and $s = \phi(s')$. Therefore, since $\varsigma(t), s \in \mathcal{T}(\Sigma_{\perp})$, we have that $(\!|s|\!)_F = (\!|s|\!)_{FL} = (\!|\phi(s')|\!)_{FL} = \phi(s')$. Thus, $\varsigma\ Narr(t)\ \phi(s')$ and, since $Narr(t)$ is an approximable mapping and $\delta \sqsubseteq (\!|s|\!)_{FL} = \phi(s')$, we have $\varsigma\ Narr(t)\ \delta$, i.e., $Rew(\varsigma(t))\!\downarrow\ \subseteq CNarr^{\infty}(t)\ [\varsigma]$. □

5.1 Narrowing strategies

A narrowing strategy $\mathcal{N}$ is a restriction on the set of possible narrowing steps. Given a narrowing strategy $\mathcal{N}$ and a term t, we can consider the set $\mathit{NDeriv}_{\mathcal{N}}(t) \subseteq \mathit{NDeriv}(t)$ of derivations which start from t and conform to $\mathcal{N}$. By Proposition 5.9, each $A \in \mathit{NDeriv}_{\mathcal{N}}(t)$ defines an approximable mapping $Narr^A(t)$ which is obviously contained in $Narr(t)$. By Proposition 4.9 (when we consider left-linear, confluent CB-TRSs), $\overline{Narr^A(t)} \sqsubseteq \overline{Narr(t)} = CNarr^{\infty}(t)$. Therefore, $\{\overline{Narr^A(t)} \mid A \in \mathit{NDeriv}_{\mathcal{N}}(t)\}$ is bounded by $CNarr^{\infty}(t)$. Since $[\mathcal{T}^{\infty}(\mathcal{C}_{\perp})^V \to \mathcal{T}^{\infty}(\mathcal{C}_{\perp})]$ is a domain, it is consistently complete, i.e., the *lub* of every bounded subset actually exists (Theorem 3.1.10 in [SLG94]). Thus, for left-linear, confluent CB-TRSs, we fix

$$CNarr^{\infty}_{\mathcal{N}}(t) = \bigsqcup\{\overline{Narr^A(t)} \mid A \in \mathit{NDeriv}_{\mathcal{N}}(t)\}$$

to be the meaning of t when it is evaluated under the narrowing strategy $\mathcal{N}$. Clearly, for all narrowing strategies $\mathcal{N}$, $CNarr^{\infty}_{\mathcal{N}} \sqsubseteq CNarr^{\infty}$. Thus, $CNarr^{\infty}$ provides a semantic reference for narrowing strategies. Strategies that satisfy $CNarr^{\infty}_{\mathcal{N}} = CNarr^{\infty}$ can be thought of as correct strategies. Note that, being a continuous mapping, $CNarr^{\infty}_{\mathcal{N}}(t)$ also has an associated approximable mapping (see [SLG94]).

Remark 5.18 *Narrowing is able to yield* the graph *of a function f by computing $CNarr^{\infty}(f(\overline{x}))$, where $x_1, \ldots, x_{ar(f)}$ are different variables. This gives an interesting perspective of narrowing as an operational mechanism which computes denotations of* functions *as a whole, rather than only values of particular function calls. A similar observation can be made for narrowing strategies.*

In order to highlight similarities in the semantic description of narrowing and rewriting, let us compare the mathematical treatment of $\mathit{Rew}(t)$ and $Narr(t)$:

Rewriting	**Narrowing**
• $\mathit{Rew}(t)$ is a set of partial constructor terms $\delta \in \mathcal{T}(\mathcal{C}_{\perp}, V)$.	• $Narr(t)$ is a set of pairs $\langle\varsigma, \delta\rangle$, where ς is a valuation on $\mathcal{T}(\mathcal{C}_{\perp})$ and $\delta \in \mathcal{T}(\mathcal{C}_{\perp})$.
• $\mathit{Rew}(t)$ is a directed set.	• $Narr(t)$ is an approximable mapping.
• The limit $\mathit{CRew}^{\infty}(t)$ of $\mathit{Rew}(t)$ within the domain $\mathcal{T}^{\infty}(\mathcal{C}_{\perp}, V)$ is a (possibly infinite) *value*.	• The 'limit' of $Narr(t)$ is a continuous *mapping* $CNarr^{\infty}(t)$ from valuations to (infinite) constructor terms, i.e, a *non-ground value.*

6 Computational interpretation of the semantic descriptions

The aim of our semantic descriptions is to provide a clear computational interpretation of the semantic information. After the abstraction process that every semantic description involves (in our case, by using observation mappings), we ask ourselves: *what kind of operational information can be obtained from the semantic description?* This is essential for defining accurate analyses by using the semantic description. In this section we specially investigate the correspondence between the semantic description of the computational processes of rewriting and narrowing when they succeed in founding values.

Proposition 6.1 *Let $\mathcal{R}$ be a confluent TRS, $t \in \mathcal{T}(\Sigma_\perp, V)$, and $\delta \in \mathcal{T}(\mathcal{C}, V)$. Then, $\delta = CRew^\infty(t)$ if and only if $t \to^* \delta$.*

PROOF. If $t \to^* \delta$, then $(\!|\delta|\!)_F = \delta \in Rew(t)$. Since δ is maximal and, by Proposition 3.2, $Rew(t)$ is directed, it follows that $\delta = CRew^\infty(t)$. The opposite statement follows a similar reasoning. □

Proposition 6.2 *Let $\mathcal{R}$ be a TRS, $t \in \mathcal{T}(\Sigma_\perp, V)$, $\mathbb{F}$ be a rewriting strategy, and $\delta \in \mathcal{T}(\mathcal{C}, V)$. Then, $\delta = CRew^\infty_\mathbb{F}(t)$ if and only if $t \to^*_\mathbb{F} \delta$.*

PROOF. Similar to Proposition 6.1. □

Concerning narrowing computations, we have the following result.

Proposition 6.3 *Let $\mathcal{R}$ be a left-linear, confluent CB-TRS. Let t be a term, $\varsigma \in \mathcal{T}(\mathcal{C}_\perp)^{\mathcal{V}ar(t)}$, $m = CNarr^\infty(t)$, and $\delta = m(\varsigma)$.*

1. *For every narrowing derivation $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ such that $\phi \circ \sigma = \varsigma$ for some $\phi \in \mathcal{T}(\mathcal{C}_\perp)^V$, we have $(\!|\phi(s)|\!)_{FL} \sqsubseteq \delta$.*

2. *If $\delta \in \mathcal{T}(\mathcal{C}_\perp)$, there exists a narrowing derivation $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ and $\phi \in \mathcal{T}(\mathcal{C}_\perp)^V$ such that $\phi \circ \sigma = \varsigma$ and $\delta = (\!|\phi(s)|\!)_{FL}$.*

3. *If $\delta \in \mathcal{T}(\mathcal{C})$, then there exists a narrowing derivation $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ and $\phi \in \mathcal{T}(\mathcal{C}_\perp)^V$ such that $s \in \mathcal{T}(\mathcal{C}, V)$, $\phi \circ \sigma = \varsigma$, and $\delta = \phi(s)$.*

PROOF.

1. If $A : \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ is such that $\phi \circ \sigma = \varsigma$, by definition of $Narr^A(t)$ we have $\varsigma\ Narr^A(t)\ (\!|\phi(s)|\!)_{FL}$, i.e., $\varsigma\ Narr(t)\ (\!|\phi(s)|\!)_{FL}$. By Proposition 4.11, the conclusion follows.

2. By Proposition 4.12, we have that $\varsigma\ Narr(t)\ \delta$. Thus, by definition of $Narr(t)$, there is a narrowing derivation $A : \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ such that $\varsigma\ Narr^A(t)\ \delta$. Hence, there exists $\phi \in \mathcal{T}(\mathcal{C}_\perp)^V$ such that $\phi \circ \sigma = \varsigma$ and $\delta \sqsubseteq (\!|\phi(s)|\!)_{FL}$. Using (1), we conclude $\delta = (\!|\phi(s)|\!)_{FL}$.

3. By using (2), we conclude that there exists a narrowing derivation $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ and $\phi \in \mathcal{T}(\mathcal{C}_\perp)^V$ such that $\phi \circ \sigma = \varsigma$ and $\delta = (\!| \phi(s) |\!)_{FL}$. Assume that $s \notin \mathcal{T}(\mathcal{C}, V)$. Then, there exists a defined symbol $f \in \mathcal{F}$ in s. Then, $\perp$ occurs in $\delta = (\!| \phi(s) |\!)_{FL}$ thus contradicting the fact that $\delta \in \mathcal{T}(\mathcal{C})$.

□

Proposition 6.3(1) expresses that, given a (finite) valuation ς, we can use any narrowing derivation starting from a term t that computes a substitution more general than ς to *approximate* the value δ that, according to the semantic interpretation of t as a non-ground value, corresponds to ς. Proposition 6.3(2) and (3) say that every finite (partial) value which corresponds to a finite valuation can be exactly recovered by (observing) a narrowing derivation.

We are able to refine the computational information couched by the narrowing semantics by introducing a small modification on it.

Definition 6.4 *Given a term $t \in \mathcal{T}(\Sigma_\perp, V)$, and a narrowing derivation*

$$A : \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle = \langle \sigma_0, t_0\rangle \leadsto \langle \sigma_1, t_1\rangle \leadsto \cdots \leadsto \langle \sigma_{n-1}, t_{n-1}\rangle \leadsto \langle \sigma_n, t_n\rangle$$

we let $BNarr^A(t) = \cup_{0\le i\le n} BNarr_i^A(t)$ where:

$$BNarr_i^A(t) = \{\langle \varsigma, \delta\rangle \in \mathcal{T}(\mathcal{C}_\perp)^V \times \mathcal{T}(\mathcal{C}_\perp) \mid (\!| \sigma_i |\!)_{FL} \sqsubseteq \varsigma \wedge \delta \sqsubseteq (\!| t_i |\!)_{FL}\}$$

Proposition 6.5 *Let $\mathcal{R}$ be a TRS, t be a term and A be a narrowing derivation starting from t. Then $BNarr^A(t)$ is an approximable mapping.*

Proof. Let

$$A : \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle = \langle \sigma_0, t_0\rangle \leadsto \langle \sigma_1, t_1\rangle \leadsto \cdots \leadsto \langle \sigma_{n-1}, t_{n-1}\rangle \leadsto \langle \sigma_n, t_n\rangle$$

We abbreviate $BNarr^A(t)$ by m. Then, we check the conditions of Definition 4.8.

1. Note that, for all derivations A starting from t,

   $$BNarr_0^A = \{\langle \varsigma, \delta\rangle \mid (\!| \varepsilon_{|\mathcal{V}ar(t)} |\!)_{FL} \sqsubseteq \varsigma \wedge \delta \sqsubseteq (\!| t |\!)_{FL}\} \subseteq m.$$

   We have that $(\!| \varepsilon_{|\mathcal{V}ar(t)} |\!)_{FL} = \perp_{Valuat}^{\mathcal{V}ar(t)}$ and we obtain $\perp_{Valuat}^{\mathcal{V}ar(t)} \sqsubseteq \perp_{Valuat}^{\mathcal{V}ar(t)} = \varsigma$, and $\delta = \perp \sqsubseteq (\!| t |\!)_{FL}$, i.e., $\perp_{Valuat}^{\mathcal{V}ar(t)}\ m\ \perp$.

2. Let $\varsigma\ m\ \delta$ and $\varsigma\ m\ \delta'$. By definition of m, there are σ_i, σ_j, such that $(\!| \sigma_i |\!)_{FL} \sqsubseteq \varsigma$, $\delta \sqsubseteq (\!| t_i |\!)_{FL}$, $(\!| \sigma_j |\!)_{FL} \sqsubseteq \varsigma$, and $\delta' \sqsubseteq (\!| t_j |\!)_{FL}$ for some $0 \le i \le j \le n$. By Proposition 4.7 $(\!| t_i |\!)_{FL} \sqsubseteq (\!| t_j |\!)_{FL}$, i.e., $\{\delta, \delta'\}$ is bounded by $(\!| t_j |\!)_{FL}$. Thus, $\delta \sqcup \delta' \sqsubseteq (\!| t_j |\!)_{FL}$ and, by definition of m, $\varsigma\ m\ (\delta \sqcup \delta')$.

3. Let $\varsigma\ m\ \delta$, $\varsigma \sqsubseteq \varsigma'$, and $\delta' \sqsubseteq \delta$. Thus, there is σ_i, $0 \le i \le n$ such that $(\!| \sigma_i |\!)_{FL} \sqsubseteq \varsigma$ and $\delta \sqsubseteq (\!| t_i |\!)_{FL}$. Since $(\!| \sigma_i |\!)_{FL} \sqsubseteq \varsigma \sqsubseteq \varsigma'$ and $\delta' \sqsubseteq \delta \sqsubseteq (\!| t_i |\!)_{FL}$, by definition of m, we also have that $\varsigma'\ m\ \delta'$.

□

Since each $BNarr_i^A(t)$ is a special case of $Narr_i^A(t)$, in which only $\phi = \bot_{Valuat}$ is allowed, we have that $BNarr^A(t) \subseteq Narr^A(t)$. Therefore, by Propositions 5.9 and 6.5, and using Proposition 4.9, we have that, for all terms t, $\overline{BNarr^A(t)} \sqsubseteq \overline{Narr^A(t)}$. Whenever we consider left-linear, confluent CB-TRSs, Proposition 5.15 and Proposition 4.9 ensure that $\{\overline{BNarr^A(t)} \mid A \in \mathit{NDeriv}(t)\}$ is bounded by $CNarr^\infty(t)$. Thus, for left-linear, confluent CB-TRSs, we fix

$$BNarr^\infty(t) = \bigsqcup\{\overline{BNarr^A(t)} \mid A \in \mathit{NDeriv}(t)\}$$

as the basic description of narrowing computations. Clearly, $BNarr^\infty(t) \sqsubseteq CNarr^\infty(t)$.

Example 6.6 *Consider the TRS $\mathcal{R}$ in Example 4.2 and term $t = \mathtt{x+y}$. For the narrowing derivations:*

$$A1: \quad \langle\{\mathtt{x} \mapsto \mathtt{x}, \mathtt{y} \mapsto \mathtt{y}\}, \mathtt{x+y}\rangle \quad \rightsquigarrow \quad \langle\{\mathtt{x} \mapsto \mathtt{0}, \mathtt{y} \mapsto \mathtt{y'}\}, \mathtt{y'}\rangle$$

and

$$\begin{array}{rlcl} A2: & \langle\{\mathtt{x} \mapsto \mathtt{x}, \mathtt{y} \mapsto \mathtt{y}\}, \mathtt{x+y}\rangle & \rightsquigarrow & \langle\{\mathtt{x} \mapsto \mathtt{s(x')}, \mathtt{y} \mapsto \mathtt{y'}\}, \mathtt{s(x'+y')}\rangle \\ & & \rightsquigarrow & \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \mathtt{y''}\}, \mathtt{s(y'')}\rangle \end{array}$$

we show (part of) their semantic descriptions $BNarr^{A1}$ and $BNarr^{A2}$ (the reader can compare such semantic descriptions and those given by $Narr^{A1}$ and $Narr^{A2}$ in Example 5.3):

$$\begin{array}{rcl} BNarr_0^{A1}(t) & = & \{\langle\varsigma, \bot\rangle \mid \varsigma \in \mathcal{T}^\infty(\mathcal{C}_\bot)^{\{\mathtt{x},\mathtt{y}\}}\} \\ BNarr_1^{A1}(t) & = & \{\langle\{\mathtt{x} \mapsto \mathtt{0}, \mathtt{y} \mapsto \bot\}, \bot\rangle, \\ & & \langle\{\mathtt{x} \mapsto \mathtt{0}, \mathtt{y} \mapsto \mathtt{0}\}, \bot\rangle, \\ & & \langle\{\mathtt{x} \mapsto \mathtt{0}, \mathtt{y} \mapsto \mathtt{s(}\bot\mathtt{)}\}, \bot\rangle, \\ & & \ldots\} \end{array}$$

$$\begin{array}{rcl} BNarr_0^{A2}(t) & = & \{\langle\varsigma, \bot\rangle \mid \varsigma \in \mathcal{T}^\infty(\mathcal{C}_\bot)^{\{\mathtt{x},\mathtt{y}\}}\} \\ BNarr_1^{A2}(t) & = & \{\langle\{\mathtt{x} \mapsto \mathtt{s(}\bot\mathtt{)}, \mathtt{y} \mapsto \bot\}, \bot\rangle, \langle\{\mathtt{x} \mapsto \mathtt{s(}\bot\mathtt{)}, \mathtt{y} \mapsto \bot\}, \mathtt{s(}\bot\mathtt{)}\rangle, \\ & & \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \bot\}, \bot\rangle, \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \bot\}, \mathtt{s(}\bot\mathtt{)}\rangle, \\ & & \langle\{\mathtt{x} \mapsto \mathtt{s(}\bot\mathtt{)}, \mathtt{y} \mapsto \mathtt{0}\}, \bot\rangle, \langle\{\mathtt{x} \mapsto \mathtt{s(}\bot\mathtt{)}, \mathtt{y} \mapsto \mathtt{0}\}, \mathtt{s(}\bot\mathtt{)}\rangle, \\ & & \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \mathtt{0}\}, \bot\rangle, \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \mathtt{0}\}, \mathtt{s(}\bot\mathtt{)}\rangle, \\ & & \ldots\} \\ BNarr_2^{A2}(t) & = & \{\langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \bot\}, \bot\rangle, \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \bot\}, \mathtt{s(}\bot\mathtt{)}\rangle, \\ & & \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \mathtt{0}\}, \bot\rangle, \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \mathtt{0}\}, \mathtt{s(}\bot\mathtt{)}\rangle, \\ & & \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \mathtt{s(}\bot\mathtt{)}\}, \bot\rangle, \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \mathtt{s(}\bot\mathtt{)}\}, \mathtt{s(}\bot\mathtt{)}\rangle, \\ & & \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \mathtt{s(0)}\}, \bot\rangle, \langle\{\mathtt{x} \mapsto \mathtt{s(0)}, \mathtt{y} \mapsto \mathtt{s(0)}\}, \mathtt{s(}\bot\mathtt{)}\rangle, \\ & & \ldots\} \end{array}$$

The basic description $BNarr^\infty(t)$ is closer to the computational mechanism of narrowing. The following propositions formalize this claim.

Proposition 6.7 *Let $\mathcal{R}$ be a left-linear, confluent CB-TRS, t be a term, $\varsigma \in \mathcal{T}(\mathcal{C}_\perp)^V$, $m = BNarr^\infty(t)$, and $\delta = m(\varsigma)$.*

1. *For every narrowing derivation $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ such that $(\!|\sigma|\!)_{FL} \sqsubseteq \varsigma$, it is $(\!|s|\!)_{FL} \sqsubseteq \delta$.*
2. *If $\delta \in \mathcal{T}(\mathcal{C}_\perp)$, there exists a narrowing derivation $\langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ such that $\phi \circ \sigma = \varsigma$ and $\delta = (\!|s|\!)_{FL}$ for some $\phi \in \mathcal{T}(\mathcal{C}_\perp)^V$.*

PROOF.

1. If $A : \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ is such that $(\!|\sigma|\!)_{FL} \sqsubseteq \varsigma$, then, by definition of $BNarr^A(t)$, we have that $\varsigma\ BNarr^A(t)\ (\!|s|\!)_{FL}$. Therefore, $(\!|s|\!)_{FL} \sqsubseteq \overline{BNarr^A(t)}\ \varsigma \sqsubseteq m(\varsigma) = \delta$.
2. By Proposition 4.14, there is a narrowing derivation $A : \langle \varepsilon_{|\mathcal{V}ar(t)}, t\rangle \leadsto^* \langle \sigma, s\rangle$ such that $\delta = \overline{BNarr^A(t)}\ \varsigma$. By Proposition 4.12, $\varsigma\ BNarr^A(t)\ \delta$. Since $(\!|\sigma|\!)_{FL} \sqsubseteq \varsigma$, by using (1), we conclude $(\!|s|\!)_{FL} \sqsubseteq \delta$. By definition of $BNarr^A(t)$, $\delta \sqsubseteq (\!|s|\!)_{FL}$ and the conclusion follows.

□

Proposition 6.8 *Let $\mathcal{R}$ be a left-linear, confluent CB-TRS, t be a term, and $m = BNarr^\infty(t)$. If $\langle \varepsilon, t\rangle \leadsto^* \langle \sigma, \delta\rangle$ and $\delta \in \mathcal{T}(\mathcal{C})$, then $m((\!|\sigma|\!)_{FL}) = \delta$.*

PROOF. Let $\delta' = m((\!|\sigma|\!)_{FL})$. By Proposition 6.7(1), $(\!|\delta|\!)_{FL} \sqsubseteq \delta'$. Since $\delta \in \mathcal{T}(\mathcal{C})$, $(\!|\delta|\!)_{FL} = \delta$; moreover, since δ is maximal, $\delta \not\sqsubset \delta'$. Hence, $\delta = \delta' = m((\!|\sigma|\!)_{FL})$. □

The basic description $BNarr^\infty(t)$ is closer to narrowing as an operational mechanism. However, $CNarr^\infty(t)$ actually provides a more complete semantic description as stressed by the following example.

Example 6.9 *Consider the TRS $\mathcal{R}$ in Example 4.2 and term $t =$ `x+y`. According to Example 6.6, we have that*

$$BNarr^\infty(\texttt{x+y})\ \{\texttt{x} \mapsto \texttt{0}, \texttt{y} \mapsto \texttt{0}\} = \perp$$

However, according to Example 5.3, we have that

$$CNarr^\infty(\texttt{x+y})\ \{\texttt{x} \mapsto \texttt{0}, \texttt{y} \mapsto \texttt{0}\} = \texttt{0}$$

Moreover, Example 6.9 shows that, different to $CNarr^\infty$ (see Theorem 5.17), $BNarr^\infty$ is *not* complete w.r.t. the rewriting semantics.

7 Towards a semantics-based analysis framework

In the previous sections we have developed a semantic characterization of the evaluation of expressions under narrowing or arbitrary narrowing strategies as the computation of *functional* values. To demonstrate the usefulness of this semantics for the analysis of functional logic programs, we provide in this section an algebraic perspective of the analysis of functional logic programs where the functional construction is also essential. We also sketch a possible application: the combined analysis of termination and groundness properties of functional logic programs.

Domain theory provides a framework for formulating properties of programs and discussing about them [Abr91, Sco81]: A property π of a program $\mathcal{P}$ whose denotation $[\![\mathcal{P}]\!]$ is taken from a domain D (i.e., $[\![\mathcal{P}]\!] \in D$) can be identified with a predicate $\pi : D \to \mathbf{2}$, where $\mathbf{2}$ is the two point domain $\mathbf{2} = \{\bot, \top\}$ ordered by $\bot \sqsubseteq \top$ (where $\bot$ can be thought of as false and $\top$ as true). A program $\mathcal{P}$ satisfies π if $\pi([\![\mathcal{P}]\!]) = \top$ (alternatively, if $[\![\mathcal{P}]\!] \in \pi^{-1}(\top)$). As usual in domain theory, we require continuity of π for achieving computability (or *observability*, see [Smy83, Vic89]). The set $[D \to \mathbf{2}]$ of observable properties is (isomorphic to) the family of open sets of the *Scott's topology* associated to D [Abr91]. A *topology* is a pair (X, τ) where X is a set and $\tau \subseteq \mathcal{P}(X)$ is a family of subsets of X (called the *open* sets) such that [SLG94]: $X, \varnothing \in \tau$; if $U, V \in \tau$, then $U \cap V \in \tau$; and if $U_i \in \tau$ for $i \in I$, then $\bigcup_{i \in I} U_i \in \tau$. The Scott's topology associated to a domain D is given by the set of upward closed subsets $U \subseteq D$ such that, whenever $A \subseteq D$ is directed and $\bigsqcup A \in U$, then $\exists x \in A.x \in U$ [SLG94].

Note that, when considering the Scott's topology (D, τ_D) of a domain D, the open set D denotes a trivial property which every program satisfies; $\varnothing$, the least element of lattice τ_D, denotes the 'impossible' property, which no program satisfies.

7.1 Analysis of functional logic programs

A program analysis consists in the definition of a continuous function $\alpha : D \to A$ between topologic spaces (D, τ_D) and (A, τ_A) which expresses concrete and abstract properties, respectively. By the topological definition of continuity, each open set $V \in \tau_A$ maps to an open set $U \in \tau_D$ via α^{-1}, i.e., $\alpha^{-1} : \tau_A \to \tau_D$ is a mapping from abstract properties (open sets of τ_A) to concrete properties (open sets of τ_D). It is easy to see that $(D, \{\alpha^{-1}(V) \mid V \in \tau_A\})$ is a subtopology of D (i.e., $\{\alpha^{-1}(V) \mid V \in \tau_A\} \subseteq \tau_D$). Therefore, each analysis distinguishes a subset of properties of D which is itself a topology. Note that τ_A plays the role of an *abstract domain* in the usual, lattice-based, abstract interpretation approaches. For instance, the Scott's topology of $\mathbf{2}$ is given by $\tau_{\mathbf{2}} = \{\varnothing, \{\top\}, \mathbf{2}\}$. Such a topology permits to express only one non-trivial property, namely, the one which corresponds to the open set $\{\top\}$.

In functional logic languages, the semantic domain under observation is $[D^V \to D]$ where $D = \mathcal{T}^\infty(\mathcal{C}_\bot)$. Observable properties of functional logic programs are open sets of its Scott's topology. Approximations to such properties can be obtained by abstracting $[D^V \to D]$ into a suitable abstract domain (see below).

Every continuous function $f : D \to E$ maps observable properties of the codomain E into observable properties of D (by $f^{-1} : \tau_E \to \tau_D$). In particular, elements of $[D^V \to D]$, i.e., denotations of functional logic programs, map properties of D (we call them 'functional' properties) into properties of D^V ('logic' properties). This provides an additional, interesting analytic perspective: By rephrasing Dybjer [Dyb91], we can computationally interpret this correspondence as establishing the extent that a 'logic property' (concerning valuations) needs to be ensured to guarantee a property of its functional part (computed value). There is a simple way to obtain an abstraction of the logic part D^V of $[D^V \to D]$ from an abstraction of its functional part D.

Definition 7.1 *Let D, V, A be sets. Let $\alpha_F : D \to A$ be a mapping. Then, $\alpha_L : D^V \to A^V$ given by $\alpha_L(\phi) = \alpha_F \circ \phi$, for all $\phi \in D^V$, is called the logic abstraction induced by α_F.*

If $\alpha_F : D \to A$ is strict (surjective, continuous), then α_L is strict (surjective, continuous). Whenever α_F is a *continuous* mapping from a domain D to $\mathbf{2}$, α_F expresses, in fact, a single observable property $\alpha^{-1}(\{\top\})$ of D. We can thought of α_F as a *functional* property. Thus, Definition 7.1 associates an abstraction α_L of D^V to a given property identified by α_F. Thus, each functional property induces a related *set* of logic properties which is a *subtopology* of τ_{D^V}. In Section 7.3 we show that groundness (a logic property), is induced by the functional property of termination.

7.2 Approximation of functions

Abstractions $\alpha_D : D \to A$ and $\alpha_E : E \to B$ (A and B being algebraic lattices), induce *safety* and *liveness* abstractions $\alpha^S_{D\to E}, \alpha^L_{D\to E} : (D \to E) \to (A \to B)$, of continuous mappings by [Abr90]

$$\alpha^S_{D\to E}(f)(d) = \sqcup\{(\alpha_E \circ f)(d') \mid \alpha_D(d') \sqsubseteq d\}, \quad \text{and}$$
$$\alpha^L_{D\to E}(f)(d) = \sqcap\{(\alpha_E \circ f)(d') \mid \alpha_D(d') \sqsupseteq d\}$$

where the following correctness result holds:

Theorem 7.2 (The semi-homomorphism property [Abr90]**)** *Let $f : D \to E$, $f^S = \alpha^S_{D\to E}(f)$, and $f^L = \alpha^L_{D\to E}(f)$. Then, $f^L \circ \alpha_D \sqsubseteq \alpha_E \circ f \sqsubseteq f^S \circ \alpha_D$.*

Consider an abstraction $\alpha_E : E \to \mathbf{2}$ which can be thought of as a *property* of elements of the codomain E of $f : D \to E$. For analytic purposes, the correctness condition $f^S \circ \alpha_D \sqsupseteq \alpha_E \circ f$ ensures that, for all $x \in D$, whenever the abstract computation $f^S(\alpha_D(x))$ yields $\perp$, the concrete computation $f(x)$ does *not* satisfy the property α_E, i.e., $\alpha_E(f(x)) = \perp$. On the other hand, the correctness condition $f^L \circ \alpha_D \sqsubseteq \alpha_E \circ f$ ensures that, whenever $f^L(\alpha_D(x))$ yields $\top$, the concrete computation $f(x)$ actually satisfies α_E, i.e., $\alpha_E(f(x)) = \top$. We use this computational interpretation later.

7.3 Termination analysis and groundness analysis

The functional structure of the semantic domain of *ngv*'s reveals connections between apparently disconnected analyses. Consider $h_t : \mathcal{T}^\infty(\mathcal{C}_\perp) \to \mathbf{2}$ defined by

$$h_t(\delta) = \begin{cases} \top & \text{if } \delta \in \mathcal{T}(\mathcal{C}) \\ \perp & \text{otherwise} \end{cases}$$

and let $h_g : \mathcal{T}^\infty(\mathcal{C}_\perp)^V \to \mathbf{2}^V$ be the logic abstraction induced by h_t. Note that both h_t and h_g are strict and continuous. Abstractions h_t and h_g express the observable properties of (successful) termination and groundness, respectively: Recall that the only nontrivial open set of the the Scott's topology of $\mathbf{2}$ is $\{\top\}$. By continuity of h_t, $h_t^{-1}(\{\top\})$ is the (open) set of finite, totally defined values which actually corresponds to terminating successful evaluations.

Remark 7.3 *h_t and Mycroft's abstraction:*

$$halt(d) = \begin{cases} \top & \text{if } d \neq \perp \\ \perp & \text{if } d = \perp \end{cases}$$

for termination analysis [Myc80] are similar. However, halt expresses termination only if $\mathcal{C}$ contains only constant symbols. It is easy to see that, in this case, $h_t = halt$.

On the other hand, each open set of $\mathbf{2}^V$ is (isomorphic to) an upward closed collection of sets of variables ordered by inclusion. In this case, $h_g^{-1}(U)$ for a given open set U is a set of substitutions whose bindings for variables belonging to $X \in U$ are ground. This formally relates groundness and termination: groundness is the 'logic' property which corresponds to the 'functional' property of termination. In fact, $\mathbf{2}^V$ is a well-known abstract domain for *groundness* analysis in logic programming [JS87].

If $\mathcal{C}$ has constructors with positive arity, then $h_t^{-1}(\{\top\})$ is the set of constructor-rooted values (they correspond to terms having a constructor-rooted head-normal form). In this case, $h_g^{-1}(U)$ for a given open set U is a set of substitutions whose bindings for variables belonging to $X \in U$ has been instantiated with some constructor-rooted term.

7.4 Using semantic information for improving the evaluation

Groundness information can be used to improve the narrowing evaluation of a term $t = C[t_1, \ldots, t_n]$: if we know that every successful evaluation of t_i grounds the variables of t_j, for some $1 \leq i, j \leq n$, $i \neq j$, then it is sensible to evaluate t by first narrowing t_i (up to a value) and next evaluating t'_j (i.e., t_j after instantiating its variables using the bindings created by the evaluation of t_i) by *rewriting* because, after evaluating t_i, we know that t'_j is ground and we do not need to provide code for unification, instantiation of other variables, etc.

Example 7.4 *Consider the following TRS:*

```
        0+x → x              if(true ,x,y) → x
     s(x)+y → s(x+y)         if(false,x,y) → y
even(0)     → true           even(s(s(x))) → even(x)
even(s(0)) → false
```

For an initial (conditional) expression "`if even(x) then x+x else s(x+x)`*" (we use the more familiar notation* `if then else` *for* `if` *expressions), it is clear that* `x` *becomes ground after every successful narrowing evaluation of the condition* `even(x)`*. Thus, we can evaluate* `x+x` *by rewriting instead of narrowing.*

Additionally, we need to ensure that the evaluation of t_i is safe under the context C (i.e., that failing evaluations of t_i do not prevent the evaluation of t). Eventually, we should also ensure that the complete evaluation of t'_j is safe. Strictness information can be helpful here: if the (normalizing) narrowing strategy is not able to obtain any value, this means that the whole expression does not have a value. However, we should only use non-contextual strictness analyses (like Mycroft's [Myc80] is). In this way, we ensure that the strict character of an argument is not altered after a possible instantiation of its surrounding context.

In order to ensure that *every* successful narrowing derivation grounds a given variable $x \in \mathcal{V}ar(t)$, we use the safety abstraction $m^S \in \mathbf{2}^V \to \mathbf{2}$ of $m = BNarr^\infty(t)$ (based on h_t and h_g).

Example 7.5 *(Continuing Example 7.4) For* $t = \mathtt{even(x)}$*, we have:*

$$
\begin{array}{lll}
BNarr^\infty(t) & = \{ & \{\mathtt{x} \mapsto \bot\} \mapsto \bot, \qquad\qquad \{\mathtt{x} \mapsto \mathtt{0}\} \mapsto \mathtt{true}, \\
 & & \{\mathtt{x} \mapsto \mathtt{s(\bot)}\} \mapsto \bot, \qquad\quad \{\mathtt{x} \mapsto \mathtt{s(0)}\} \mapsto \mathtt{false}, \\
 & & \{\mathtt{x} \mapsto \mathtt{s(s(\bot))}\} \mapsto \bot, \quad \{\mathtt{x} \mapsto \mathtt{s(s(0))}\} \mapsto \mathtt{true}, \\
 & & \ldots\}
\end{array}
$$

In general, if we can prove that, for all abstract substitutions $\phi^\# \in \mathbf{2}^V$ with $\phi^\#(x) = \bot$, it is $m^S(\phi^\#) = \bot$, then we can ensure that x is grounded in every successful derivation from t. To see this point, consider a successful derivation $\langle \varepsilon, t\rangle \leadsto^* \langle \sigma, \delta\rangle$ such that $\delta \in \mathcal{T}(\mathcal{C})$ and $\sigma(x) \notin \mathcal{T}(\mathcal{C})$, i.e., x is not grounded. By Proposition 6.8, $m((\!|\sigma|\!)_{FL}) = \delta$. By definition of m^S, $m^S(h_g((\!|\sigma|\!)_{FL})) = \top$. Since $(\!|\sigma(x)|\!)_{FL} \notin \mathcal{T}(\mathcal{C})$, we have $h_g((\!|\sigma|\!)_{FL})(x) = h_t((\!|\sigma(x)|\!)_{FL}) = \bot$, thus contradicting (a particularization of) our initial assumption, $m^S(h_g((\!|\sigma|\!)_{FL})) = \bot$.

Example 7.6 *(Continuing Example 7.5) For* $t = \mathtt{even(x)}$*, we have* $m^S = \{\{x \mapsto \bot\} \mapsto \bot, \{x \mapsto \top\} \mapsto \top\}$*. Thus,* `x` *is grounded in every successful derivation of* `even(x)`*.*

The previous considerations show that the semantic dependency expressed by the *ngv*'s has the corresponding translation for the analysis questions. However, the detailed development of such a program analysis framework is outside the scope of this paper and a topic for future work.

8 Related work and concluding remarks

The idea of giving denotational descriptions of different operational frameworks is not new. For instance, [Bak76] assigns different *fixpoint* semantics for a program under either call-by-name or call-by-value strategies. This shows that, in some sense, the semantic descriptions also (silently) assume some underlying operational approach (usually, call-by-name like).

In [Red85], the notion of *ngv* as the semantic object that a narrowing computation should compute was already introduced. It was also noted that narrowing only computes a *representation* of the object, not the object itself. However, it was not clearly explained how this connection can be done.

In [MR92], domains are used to give semantics to the functional logic language BABEL. However, the style of the presentation is model theoretic: all symbols take meaning from a given interpretation and the connection between the declarative and operational semantics (lazy narrowing) are given by means of the usual completeness/correctness results. The semantic domain is different from ours because of valuations are just a parameter of the semantic functions rather than as a component of the domain. Thus, the *Herbrand domain* $\mathcal{T}^\infty(\mathcal{C}_\perp)$ is the semantic domain in [MR92]. A similar remark can be made for [JPP91].

The semantic approach in [GHLR99] is much more general than [MR92] (covering non-deterministic computations), but the style of the presentation is model theoretic, too. The basic semantic domain is also different from ours: no functional domain for denotations is used and, in fact, bounded completeness, which is essential in our setting to deal with the functional construction and with narrowing strategies, is not required in [GHLR99].

In [Zar97], a denotational description of a particular narrowing strategy (the needed narrowing strategy [AEH00]) is given. The semantics is nicely applied to demandedness analysis but nothing has been said about how to use such a semantic description for more general analysis problems. This question is important since the notion of demandedness pattern is essential for the definition of the semantics itself.

We have presented a domain-theoretic approach for describing the semantics of integrated functional logic languages based on narrowing. Our semantics is parameterized by the narrowing strategy which is used by the language. The semantics is not 'model-theoretic' in the sense that we let within the operational mechanism (the narrowing strategy) to establish the 'real' meaning of the functions defined by the program rules. In this way, we are able to include more operational information into the semantic description. As far as we know, previous works have not explicitly considered arbitrary strategies for parameterizing the semantics of either functional or functional logic languages, that is, the operational-oriented denotational description formalized in this work is novel in the literature of the area.

Another interesting point of our work is its applicability to the analysis of functional logic programs. Since we use a functional domain (the domain of *non-ground-values*), we are able to associate a denotation to a term with variables.

Thus, narrowing is reformulated as an evaluation mechanism which computes the denotation of the input expression. This was already suggested by Reddy [Red85] but it is only formally established in this paper by using approximable mappings. Thanks to this perspective, we can use the standard frameworks for program analysis based on the denotational description of programs. In other words, the approximation of the domain of non-ground values provides the basis for the analysis of functional logic programs. Our description also reveals unexplored connections between purely functional and logic properties. These connections suggest that, within the functional logic setting, we have ascertained a kind of 'duality' between purely functional and purely logic properties. As far as we know, this had not been established before.

Future work includes a more detailed study about how to use this semantics to develop practical methods for the analysis of functional logic programs. For instance, we can use an abstract narrowing calculus (see, for example, [AFRV93, AFM95, Vid96]) to directly build (correct) abstract versions of the semantic functions via abstract approximable mappings. We can also adapt the Dybjer's calculus of inverse images [Dyb91] for relating functional and logic properties. Another interesting task is to extend this semantics to more general classes of programs and computation models for declarative languages [Han97].

We have presented an algebraic framework to express analysis of functional logic programs. Our intention is to use the existing (abstract interpretation based) analyses for pure functional and logic programming in our integrated framework. The explicit semantic connections between the basic paradigms allow us to combine those analyses by using the existing tools to combine abstract domains [GR95]. Particularly interesting, as a subject of future work, is the possibility of giving a logic interpretation to these domain combinations [GS97, GS98].

References

[Abr90] S. Abramsky. Abstract Interpretation, Logical Relations, and Kan Extensions. *Journal of Logic and Computation* 1(1):5-40, 1990.

[Abr91] S. Abramsky. Domain Theory in Logic Form. *Annals of Pure and Applied Logic* 51:1-77, 1991.

[AEH00] S. Antoy, R. Echahed and M. Hanus. A needed narrowing strategy. *Journal of the ACM*, 47(4):776-822, 2000.

[AFM95] M. Alpuente, M. Falaschi, and F. Manzo. Analyses of Unsatisfiability for Equational Logic Programming. *Journal of Logic Programming*, 22(3):221-252, 1995.

[AFRV93] M. Alpuente, M. Falaschi, M.J. Ramis, and G. Vidal. Optimization of Equational Logic Programs Using Abstract Narrowing. In M. Bruynooghe and J. Penjam, editors, *Proc. 5th International Symposium on Programming Language Implementation and Logic Programming, PLILP'93*, LNCS 714:391-409, Springer-Verlag, Berlin, 1993.

[AFV96] M. Alpuente, M. Falaschi, and G. Vidal. A Compositional Semantic Basis for the Analysis of Equational Horn Programs. *Theoretical Computer Science*, 165(1):97-131, 1996.

[Bak76] J.W. de Bakker. Least Fixed Points Revisited. *Theoretical Computer Science*, 2:155-181, 1976.

[BEØ93] D. Bert, R. Echahed, and B.M. Østvold. Abstract Rewriting. In P. Cousot, M. Falaschi, G. File, and A. Rauzy,, editors, *Proc. of 3rd International Workshop on Static Analysis, WSA'93*, LNCS 724:178-192, Springer-Verlag, Berlin, 1993.

[Bou85] G. Boudol. Computational semantics of term rewriting systems. In M. Nivat and J. Reynolds, editors, *Algebraic Methods in Semantics*, pages 169-236, Cambridge University Press, Cambridge, 1985.

[Cou90] B. Courcelle. Recursive Applicative Program Schemes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 459-492, Elsevier, 1990.

[DP90] B.A. Davey and H.A. Priestley. Introduction to Lattices and Order. Cambridge University Press, 1990.

[DW89] S. Debray and D. S. Warren. Functional Computations in Logic Programs. *ACM Transactions on Programming Languages and Systems* 11(3):451–481 1989.

[Dyb91] P. Dybjer. Inverse Image Analysis Generalises Strictness Analysis. *Information and Computation* 90:194-216, 1991.

[GHLR99] J.C. González-Moreno, M.T. Hortalá-González, F.J. López-Fraguas, and M. Rodríguez-Artalejo. An approach to declarative programming based on a rewriting logic. *Journal of Logic Programming* 40(1):47-87, 1999.

[GR95] R. Giacobazzi and F. Ranzato. Functional Dependencies and Moore-Set Completions of Abstract Interpretations and Semantics. In J.W. Lloyd, editor, *Proc. of the 1995 International Symposium on Logic Programming, ILPS'95*, pages 321-335, The MIT Press, Cambridge, MA, 1995.

[GS97] R. Giacobazzi and F. Scozzari. Intuitionistic Implication in Abstract interpretation. In H. Glaser and H. Kuchen, editors, *Proc. of the 9th International Symposium on Programming Languages, Implementations, Logics, and Programs, PLILP'97*, LNCS 1292:175-189

[GS98] R. Giacobazzi and F. Scozzari. A logical model for relational abstract domains *ACM Transactions on Programming Languages and Systems* 20(5), 1998.

[GTW78] J.A. Goguen, J.W. Thatcher, and E.G. Wagner. An Initial Algebra Approach to the Specification, Correctness, and Implementation of Abstract Data Types. In R.T. Yeh, editor, *Current Trends in Programming Methodology, vol. IV: data structuring*, pages 80-149, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[GTWW77] J.A. Goguen, J.W. Thatcher, E.G. Wagner, and J.B. Wright. Initial Algebra Semantics and Continuous Algebras. *Journal of the ACM* 24(1):68-95, 1977.

[Gue81] I. Guessarian. Algebraic Semantics. LNCS 99, Springer-Verlag, Berlin, 1981.

[Gun92] C.A. Gunter. Semantics of Programming Languages. The MIT Press, Cambridge, MA, 1992.

[Han94] M. Hanus. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming* 19&20:583-628, 1994.

[Han94b] M. Hanus. Towards the Global Optimization of Functional Logic Programs. In *Proc. 5th International Conference on Compiler Construction.* LNCS 786:68-82, Springer Verlag, Berlin, 1994.

[Han97] M. Hanus. A Unified Computation Model for Functional and Logic Programming. In *Conference Record of the 24th Symposium on Principles of Programming Languages POPL'97*, pages 80-93, ACM Press, 1997.

[HKL92] G. Hogen, A. Kindler and R. Loogen. Automatic Parallelization of Lazy Functional Programs. In B. Krieg-Brückner, editor, *Proc. of 4th European Symposium on Programming, ESOP'92*, LNCS 582:254-268, Springer-Verlag, Berlin, 1992.

[Hul80] J.-M. Hullot. Canonical forms and unification. In *Proc. 5th Conference on Automated Deduction*, pages 318–334. Springer LNCS 87, 1980.

[HZ94] M. Hanus and F. Zartmann. Mode Analysis of Functional Logic Programs. In *Proc. SAS'94.* LNCS 864:26-42, Springer Verlag, Berlin, 1994.

[JPP91] R. Jagadeesan, K. Pingali, P. Panangaden. A Fully Abstract Semantics for a First-Order Functional Language with Logic Variables. *ACM Transactions on Programming Languages and Systems* 13(4):577-625, 1991.

[JS87] N.D. Jones and H. Søndergaard. A semantics-based framework for the abstract interpretation of PROLOG. In S. Abramsky and C. Hankin, editors, *Abstract Interpretation of Declarative Languages*, pp. 123–142. Ellis Horwood, 1987.

[Klo92] J.W. Klop. Term Rewriting Systems. In S. Abramsky, D.M. Gabbay and T.S.E. Maibaum. *Handbook of Logic in Computer Science*, volume 3, pages 1-116. Oxford University Press, 1992.

[KMH92] H. Kuchen, J.J. Moreno-Navarro, and M. Hermenegildo. Independent AND-Parallel Implementation of Narrowing. In M. Bruynooghe and M. Wirsing, editors, *Proc. of 4th International Symposium on Programming Language Implementation and Logic Programming, PLILP'92*, LNCS 631:24-38, Springer-Verlag, Berlin, 1992.

[Lin88] G. Lindstrom. Static Analysis of Functional Programs with Logical Variables. In P. Deransart, B. Lorho, and J. Maluszynski, editors, *Proc. of International Workshop on Programming Languages Implementation and Logic Programming, PLILP'88*, LNCS 348:1-19, Springer-Verlag, Berlin, 1989.

[Luc98] S. Lucas. Root-neededness and approximations of neededness. *Information Processing Letters*, 67(5):245-254, 1998.

[Mid97] A. Middeldorp. Call by Need Computations to Root-Stable Form. In *Conference Record of the 24th ACM Symposium on Principles of Programming Languages*, pages 94-105, 1997.

[MKMWH93] J.J. Moreno-Navarro, H. Kuchen, J. Mariño, S. Winkler and W. Hans. Efficient Lazy Narrowing using Demandedness Analysis. In *Proc. PLILP'93*. LNCS 714, Springer-Verlag, Berlin, 1993, pages 167–183.

[MR92] J.J. Moreno-Navarro and M. Rodríguez-Artalejo. Logic programming with functions and predicates: the language BABEL. *Journal of Logic Programming*, 12:191-223, 1992.

[Myc80] A. Mycroft. The theory and practice of transforming call-by-need into call-by-value. In *Proc. International Symposium on Programming*, pages 269-281, Springer LNCS 83, 1975.

[Niv75] M. Nivat. On the Interpretation of Recursive Program Schemes. *Symposia Mathematica*, vol. 15, pages 255-281, Academic Press, 1975.

[Pad88] P. Padawitz. *Computing in Horn Clause Theories*, volume 16 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1988.

[Pal90] C. Palamidessi. Algebraic Properties of Idempotent Substitutions. In M.S. Patterson, editor, *Proc. of the 17th International Colloquium on Automata, Languages, and Programming, ICALP'90*, LNCS 443:386-399, Springer-Verlag, Berlin, 1990.

[Pit97] A. Pitts. Operationally-Based Theories of Program Equivalence. In A. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*, pages 241-298, Cambridge University Press, 1997.

[Red85] U.S. Reddy. Narrowing as the Operational Semantics of Functional Languages In *Proc. of IEEE International Symposium on Logic Programming*, pages 138-151, 1985.

[Rey75] J.C. Reynolds. On the interpretation of Scott's domains. *Symposia Mathematica*, vol. 15, pages 123-135, Academic Press, 1975.

[Sco70] D. Scott. Outline of a Mathematical Theory of Computation. Technical Monograph PRG-2, November 1970, Oxford University Computing Laboratory, 1970.

[Sco81] D. Scott. Lectures on a mathematical theory of computation. Monograph PRG-19. Computing Laboratory, Oxford University, 1981.

[Sco82] D. Scott. Domains for Denotational Semantics. In M. Nielsen and E.M. Schmidt, editors, *Proc. of 9th International Colloquium on Automata, Languages and Programming, ICALP'82*, LNCS 140:577-613, Springer-Verlag, Berlin, 1982.

[SLG94] V. Stoltenberg-Hansen, I. Lindström, and E.R. Griffor. Mathematical Theory of Domains. Cambridge University Press, 1994.

[Smy83] M.B. Smyth. Powerdomains and predicate transformers: a topological view. In J. Díaz, editor, *Proc. of 10th International Colloquium on Automata, Languages and Programming, ICALP'83*, LNCS 154:662-675, Springer-Verlag, Berlin, 1983.

[SR92] F. Sáenz and J. J. Ruz. Parallelism Identification in BABEL Functional Logic Programs. In *Proc. of Italian Conference on Logic Programming GULP'92*, pages 409–423, CittàStudi, 1992.

[Vic89] S. Vickers. Topology via Logic. Cambridge University Press, 1989.

[Vid96] G. Vidal. Semantics-Based Analysis and Transformation of Functional Logic Programs. Ph.D. Thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Sep 1996 (In spanish).

[Zar97] F. Zartmann. Denotational Abstract Interpretation of Functional Logic Programs. In P. Van Hentenryck, editor, *Proc. of the 4th International Static Analysis Symposium, SAS'97*, LNCS 1302:141-159, Springer-Verlag, Berlin, 1997.