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 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 runtime properties of programs, e.g., mode inference [DW89, Han94b, HZ94, Lin88], demandedness patterns [MKMWH93, Zar97], equational unsatisﬁability [AFM95, AFRV93, AFV96, BE93], 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, m@informatik.uni-kiel.de. Partially supported by the German Research Council (DFG) under grant Ha 2457/1-1.
†DISIC, UPV, Camino de Vera s/n, E-46071 Valencia, Spain, s.lucas@disic.upv.es. Partially supported by EEC-HCM grant ERBCHRXT9490024, Spanish CICYT grant TIC 98-0445-C03-01, and Acción Integrada hispano-alemana HA 997-0073.
[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 (n-gv) of an input expression. Given a domain $D$, a n-gv 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 n-gv'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 semantics-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, Klop92, SL94]. 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 $\bot$ of an ordered set $(A, \sqsubseteq)$ is called a least element (or a minimum) if $\bot \sqsubseteq a$ for all $a \in A$. If such an element exists, then $(A, \sqsubseteq, \bot)$ 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 $\bigsqcup S \subseteq b$. A set $S \subseteq A$ is downward (upward) closed if whenever $a \in S$ and $b \sqsubseteq a$ (a $\sqsupseteq b$), we have that $b \in S$. If $S = \{x, y\}$, we write $x \cup y$ 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 $\text{Id}(A)$ 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 $(A, \sqsubseteq, \bot)$ 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 $\text{approx}(a) = \{x \in K(A) \mid x \sqsubseteq a\}$ is directed and $a = \bigsqcup \text{approx}(a)$. An algebraic cpo $D$ is a domain if, whenever the set $\{x, y\} \subseteq K(D)$ is consistent, then $x \cup y$ 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 $\forall a \in A, f(a) \sqsupseteq_A a$. 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. $[A \to B]_\bot$).

By $V$ we denote a countable set of variables; $\Sigma$ denotes a signature, i.e., a set of function symbols $\{f, g, \ldots\}$, each with a fixed arity given by a function $\text{ar} : \Sigma \to \mathbb{N}$. We assume $\Sigma \cap V = \emptyset$. We denote by $\mathcal{T}(\Sigma, V)$ the set of (finite) terms built from symbols in the signature $\Sigma$ and variables in $V$. A $k$-tuple $t_1, \ldots, t_k$ of terms is denoted as $\langle t \rangle$, where $k$ will be clarified from the context.
Given a term \( t \), \( \text{Var}(t) \) is the set of variable symbols in \( t \). Sometimes, we consider a fresh constant \( \perp \) and \( \Sigma_\perp = \Sigma \cup \{ \perp \} \). Terms from \( \mathcal{T}(\Sigma, V) \) are ordered by the usual approximation ordering which is the least ordering \( \sqsubseteq \) satisfying \( \perp \sqsubseteq t \) for all \( t \) and \( f(\overline{t}) \sqsubseteq f(\overline{t}) \) if \( \overline{t} \sqsubseteq \overline{t} \), i.e., \( t_i \sqsubseteq s_i \) for all \( 1 \leq i \leq \text{ar}(f) \).

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 \( \Lambda \), we denote the empty chain. The set of positions of a term \( t \) is denoted by \( \text{Pos}(t) \). 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 \( \text{Pos}_\Sigma(t) \), and \( \text{Pos}_V(t) \) is the set of variable positions.

We denote by \( [t[x]_p \ term \ t \ with \ the \ subterm \ at \ the \ position \ replaced \ by \ s] \). 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 \( \varepsilon \) the “identity” substitution: \( \varepsilon(x) = x \) for all \( x \in V \). The set \( \text{Dom}(\sigma) = \{ x \in V \mid \sigma(x) \neq x \} \) is called the domain of \( \sigma \) and \( \text{Rng}(\sigma) = \cup_{x \in \text{Dom}(\sigma)} \text{Var}(\sigma(x)) \) its range. \( \sigma|_U \) denotes the restriction of a substitution \( \sigma \) to a subset of variables \( U \subseteq V \). A substitution \( \sigma \) is idempotent if (and only if) \( \text{Dom}(\sigma) \cap \text{Rng}(\sigma) = \emptyset \). We write \( \sigma \leq \sigma' \) if there is \( \theta \) such that \( \sigma' = \theta \circ \sigma \). A unifier of two terms \( t_1, t_2 \) is a substitution \( \sigma \) with \( \sigma(t_1) = \sigma(t_2) \). A most general unifier (mgu) of \( t_1, t_2 \) is a unifier \( \sigma \) with \( \sigma \leq \sigma' \) for all other unifiers \( \sigma' \) of \( t_1, t_2 \).

A rewrite rule (labeled \( \alpha \)) is an ordered pair \( (l, r) \), written \( \alpha : l \overset{r}{\rightarrow} r \) (or \( l \rightarrow_r r \)), with \( l, r \in \mathcal{T}(\Sigma, V) \), \( l \not\in V \) and \( \text{Var}(r) \subseteq \text{Var}(l) \). \( l \) 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 \( (\Sigma, R) \) is left-linear if for all \( l \rightarrow_r r \), \( l \) is a linear term. Given \( \mathcal{R} = (\Sigma, R) \), we consider \( \Sigma \) as the disjoint union \( \Sigma = \Sigma \cap F \cup \Sigma \cap C \), called constructors and symbols \( F \in \mathcal{F} \), called defined functions, where \( \mathcal{F} = \{ f \mid f(\overline{t}) \rightarrow_r r \} \) and \( \mathcal{C} = \Sigma - \mathcal{F} \). A constructor-based TRS (CB-TRS) is a TRS with \( t_1, \ldots, t_n \in \mathcal{T}(\Sigma, V) \) for all rules \( f(t_1, \ldots, t_n) \rightarrow_r r \).

For a given TRS \( \mathcal{R} = (\Sigma, R) \), a term \( t \) reduces to a term \( s \) (at position \( p \)), written \( t \overset{[p]=\alpha}{\rightarrow}_R s \) (or just \( t \overset{\alpha}{\rightarrow}_R s \)), if \( t \overset{[p]=\alpha}{\rightarrow}_R t_p \) and \( s = t[\sigma(r)]_p \), for some rule \( \alpha : l \overset{r}{\rightarrow} R \), position \( p \in \text{Pos}(t) \) and substitution \( \sigma \). A term \( t \) is in normal form if there is no term \( s \) with \( t \rightarrow_r s \). A TRS \( \mathcal{R} \) (or the rewrite relation \( \rightarrow_R \)) is called confluent if for all terms \( t, t_1, t_2 \) with \( t \overset{\alpha}{\rightarrow} R t_1 \) and \( t \overset{\beta}{\rightarrow} R t_2 \), there exists a term \( t_3 \) with \( t_1 \overset{\gamma}{\rightarrow} R t_3 \) and \( t_2 \overset{\delta}{\rightarrow} R t_3 \). A term \( t \) narrows to a term \( s \), written \( t \overset{[p,a,b]}{\rightarrow}_R s \) (or just \( t \overset{\alpha}{\rightarrow}_R s \)), if \( p \in \text{Pos}(t) \) and there is a substitution\(^1 \) \( \theta : \text{Var}(t) \to \mathcal{T}(\Sigma, V - \text{Var}(t)) \) such that \( \theta(t)[p,a,b] \overset{[p,a,b]}{\rightarrow}_R s \). A narrowing derivation \( t \overset{[p,a,b]}{\rightarrow}_R s \) is such that either \( t = s \) and \( \sigma = \varepsilon|_{\text{Var}(t)} \) or \( t \overset{[p,a,b]}{\rightarrow}_R t_1 \overset{[p,a,b]}{\rightarrow}_R \cdots \overset{[p,a,b]}{\rightarrow}_R t_{n+1} \overset{[p,a,b]}{\rightarrow}_R s \) and \( \sigma = (\theta_{n+1} \circ \cdots \circ \theta_1 \circ \theta_0)|_{\text{Var}(t)} \) (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 appart’).

---

\(^1\) 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 $[\cdot]$ from syntactic objects to values (remind that values are expected to be special syntactic objects). We call $[\cdot]$ an *observation mapping*. Monotonicity of $[\cdot]$ ensures that refinements (w.r.t. $\subseteq$) 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 $[\cdot]$ for observing computations performed by a given operational mechanism should be ensured by showing that $[\cdot]$ 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 $t \in \mathcal{T}(\Sigma, V)$ and the values are taken from $(\mathcal{T}^\infty(C_\bot), \subseteq, \bot)$, the domain of infinite, ground constructor (partial) terms. Formally, $(\mathcal{T}^\infty(C_\bot), \subseteq, \bot)$ is obtained from $\mathcal{T}(C_\bot)$, which is not even a cpo, as (isomorphic to) its ideal completion $(\text{Id}(\mathcal{T}(C_\bot)), \subseteq, \{\bot\})$ (see [DP90, SLG94]). In general, given a poset $P$, the mapping $[\cdot] : P \to \text{Id}(P)$ that associates the principal ideal $p_i$ to each $p \in P$ is an embedding of $P$ into the cpo $\text{Id}(P)$, i.e., for all $p, q \in P$, $p \subseteq q$ if and only if $[p] \subseteq [q]$. Since $[\cdot]$ is injective, we can understand $\text{Id}(P)$ as a completion of $P$ which actually ‘includes’ $P$. $(\mathcal{T}^\infty(C_\bot, V), \subseteq, \bot)$ is

\(^2\) Strictness of $[\cdot]$ is a consequence of decreasingness.
the domain \( \mathcal{T}^\infty(\mathcal{C}_\bot \cup V), \sqsubseteq, \bot, \) where \( ar(x) = 0 \) for all \( x \in V \).

For functional computations, we use \( \| \|_F : \mathcal{T}(\Sigma_\bot, V) \rightarrow \mathcal{T}(\mathcal{C}_\bot, V) \) given by

\[
\begin{align*}
\| x \|_F &= x \\
\| \zeta(\overline{\tau}) \|_F &= c(\| \overline{\tau} \|_F) \quad \text{if } c \in \mathcal{C} \\
\| \bot \|_F &= \bot \\
\| f(\overline{x}) \|_F &= \bot \quad \text{if } f \in \mathcal{F}
\end{align*}
\]

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 \( t, s \in \mathcal{T}(\Sigma_\bot, V) \). If \( t \rightarrow^* s \), then \( \| t \|_F \sqsubseteq \| s \|_F \).

**Proof.** By induction on the length \( n \) of the derivation \( t \rightarrow^* s \). The case \( n = 0 \) is immediate. Otherwise, let \( t \rightarrow t' \rightarrow^* s \). To prove that \( t \rightarrow t' \) implies \( \| t \|_F \sqsubseteq \| t' \|_F \), we proceed by induction on the length of the redex position \( p \in \mathcal{Pos}(t) \) of the first rewrite step. If \( p = \lambda \), then \( t = \sigma(t) = f(\overline{\tau}) \) for some rule \( \lambda \rightarrow \tau \) and defined symbol \( f \in \mathcal{F} \) (because \( \lambda = f(\overline{\tau}) \)). Hence, \( \| t \|_F = \bot \sqsubseteq \| t' \|_F \).

If \( p \neq \lambda \), we have \( p = i \cdot p' \). Then, \( t = f(\overline{\tau}) \), \( t_i \rightarrow t'_i \), and \( t_j = t'_j \) for all \( 1 \leq j \leq ar(f), i \neq j \). If \( f \in \mathcal{F} \), then \( \| t \|_F = \bot \sqsubseteq \| t' \|_F \).

If \( f \in \mathcal{C} \), then \( \| t \|_F = c(\| \overline{\tau} \|_F) \) and, since \( t \sqsubseteq t' \), \( t' = c(\overline{\tau'}) \). Therefore, by I.H., \( \| t_i \|_F \sqsubseteq \| t'_i \|_F \) and \( \| t_j \|_F \sqsubseteq \| t'_j \|_F \) 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 \( \text{Rew} : \mathcal{T}(\Sigma_\bot, V) \rightarrow \mathcal{P}(\mathcal{T}(\mathcal{C}_\bot, V)) \) provides a representation \( \text{Rew}(t) = \{ \| t_i \|_F \mid t \rightarrow^* t_i \} \) of the rewriting space of a given term \( t \).

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

**Proof.** Note that \( \text{Rew}(t) \neq \emptyset \) because \( \| t \|_F \in \text{Rew}(t) \). If \( \| t' \|_F, \| t'' \|_F \in \text{Rew}(t) \), then \( t \rightarrow^* t' \) and \( t \rightarrow^* t'' \). By confluence, there exists a term \( s \) such that \( t' \rightarrow^* s \) and \( t'' \rightarrow^* s \). Hence, \( t \rightarrow^* s \), and \( \| s \|_F \in \text{Rew}(t) \). By Proposition 3.1, \( \| t' \|_F \sqsubseteq \| t \|_F \) and \( \| t'' \|_F \sqsubseteq \| t \|_F \), i.e., \( \text{Rew}(t) \) is directed. \( \square \)

The semantic function

\[
\text{CRew}^\infty : \mathcal{T}(\Sigma_\bot, V) \rightarrow \mathcal{T}^\infty(\mathcal{C}_\bot, V)
\]

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

\[
\text{CRew}^\infty(t) = \bigsqcup_{t \in \text{Rew}(t)} \text{Rew}(t)
\]

or even

\[
\text{CRew}^\infty(t) = \text{Rew}(t)\downarrow
\]

6
in an equivalent expression which takes advantage of the correspondence between 'infinite terms' and ideals of finite terms (note that $\text{Rew}(t) \downarrow$ is an ideal). Thus, $C\text{Rew}^\infty(t)$ 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 $C\text{Rew}^\infty$ is well defined for confluent TRSs; otherwise, we cannot ensure that $\text{Rew}(t)$ is a directed set and the lub may not exist. We also note that the use of infinite terms in the codomain of $C\text{Rew}^\infty$ is necessary for dealing with non-terminating programs.

3.2 Rewriting strategies

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

**Proposition 3.3** Let $R$ be a TRS and $F$ be a rewriting strategy for $R$. For all $t \in T(\Sigma, V)$, $\text{Rew}_F(t)$ is a chain.\(^3\)

**Proof.** Immediate by Proposition 3.1. \[\square\]

Thus, we define

$$C\text{Rew}^\infty_F : T(\Sigma, V) \to T^\infty(\Sigma, V)$$

by

$$C\text{Rew}^\infty_F(t) = \bigsqcup\text{Rew}_F(t)$$

Clearly, for all strategies $F$, $C\text{Rew}^\infty_F \subseteq C\text{Rew}^\infty$ (i.e., $C\text{Rew}^\infty_F(t) \subseteq C\text{Rew}^\infty(t)$ $\forall t$). Thus, $C\text{Rew}^\infty$ provides a semantic reference for rewriting strategies. Strategies that satisfy $C\text{Rew}^\infty_F = C\text{Rew}^\infty$ 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 $f \in F$ as follows: $f(\bar{\delta}) = C\text{Rew}^\infty_F(f(\bar{\delta}))$ for all $\bar{\delta} \in T(\Sigma, F)$. Similarly, it is possible to describe a ground semantics under a given strategy $F$ by using $C\text{Rew}^\infty_F$.

4 Narrowing as an evaluation mechanism

Through its computed value $C\text{Rew}^\infty(t)$, 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 $c_D \in [D^{ar(c)} \to D]$: $[t]_D = [C\text{Rew}^\infty(t)]_D$. However, \(^3\)Formally, $\text{Rew}_F(t)$ 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 \( t_D \in [D^n \rightarrow D] \) which yields the value of \( t \) under each possible valuation \( d \in D \) of its variables on a domain \( D \). This is called a non-ground value (ngv) in [Red85] and a derived operator in [GTW78, GTW77]. It is also essentially the same as in other algebraic approaches to semantics of TRS’s and recursive program schemes such as [Bou85, Con90, Gue81, Niv75].

Given domains \( D \) and \( E \), the set \([D \rightarrow E] = \{ f : D \rightarrow E \} \) of (strict) continuous functions from \( D \) to \( E \) (pointwise) ordered by \( f \subseteq g \) if \( \forall x \in D, f(x) \subseteq g(x) \), is a domain [Gun92, SLG94]. Given a set \( W \subseteq V \) of variables, for proving that \([D^W \rightarrow D] \) is a domain whenever \( D \) is, we note that \( W_\perp = W \cup \perp \) supplied by the least ordering \( \sqsubseteq \) such that \( x \sqsubseteq x \) 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 \([W_\perp \rightarrow D] \) of continuous, strict mappings from \( W_\perp \) to \( D \). Thus, we can view \( D^W \) as this domain. In particular, if we take \( T^\infty(C_\perp) \) as the domain \( D \) of values, then \( T^\infty(C_\perp)^W \) is a domain whose least element is the mapping \( \lambda x \in W_\perp (\text{denoted } \perp^W_{\text{val}}) \). By abuse, we say that the domain of a valuation \( \phi \in D^W \) is

\[
\text{Dom}(\phi) = \{ x \in W \mid \phi(x) \neq \perp \}.
\]

Therefore, if \( D \) is a domain, \([D^V \rightarrow D] \) also is and, in particular, \([T^\infty(C_\perp)^V \rightarrow T^\infty(C_\perp)] \) is a domain. We write \( \perp_{\text{val}} \) instead of \( \perp^W_{\text{val}} \). Given a term \( t \), \([T^\infty(C_\perp)^{\text{Var}(t)} \rightarrow T^\infty(C_\perp)] \) is also a domain whose least element is denoted \( \perp_{\text{Var}(t)} \).

### 4.1 Observation of narrowing computations

Our syntax objects, now, are substitution/term pairs \( \langle \sigma, t \rangle \): Given a term \( t \in T(\Sigma, V) \) a narrowing derivation

\[
t = t_0 \sim_{\theta_0} t_1 \sim_{\theta_1} \cdots \sim_{\theta_{n-2}} t_{n-1} \sim_{\theta_{n-1}} t_n \sim_{\theta_n} s
\]

is represented as

\[
\langle \sigma_0, t_0 \rangle \sim \langle \sigma_1, t_1 \rangle \sim \cdots \sim \langle \sigma_{n-2}, t_{n-1} \rangle \sim \langle \sigma_{n-1}, t_n \rangle \sim \langle \sigma_n, t_n \rangle
\]

where \( \sigma_0 = \xi_{\text{Var}(t)} \) and \( \sigma_{i+1} = (\theta_i \circ \sigma_i)_{\text{Var}(t)} \) for \( 0 \leq i < n \). We also eventually write \( \langle \xi_{\text{Var}(t)}, t \rangle \sim^* \langle \sigma, s \rangle \) instead, where \( \sigma = \sigma_n \).

Note that, since we restrict our attention to instantiations of variables in \( \text{Var}(t) \), we have that \( \sigma_i : \text{Var}(t) \rightarrow T(\Sigma, V) \) and \( \text{Dom}(\sigma_i) \subseteq \text{Var}(t) \) for \( i \geq 0 \). Moreover, \( \text{Dom}(\sigma_i) \subseteq \text{Dom}(\sigma_j) \) 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 \( \text{Dom}(\sigma_i) \cap \text{Rng}(\sigma_i) = \emptyset \) for \( i \geq 0 \).
In order to observe the narrowing computations, we could naïvely extend \( \models F \) to deal with substitution/term pairs: \( \models [\xi, s] F = \langle [\xi] F, [s] F \rangle \) where \( [\xi] F \) is a substitution given by \( [\xi] F(x) = [\xi(x)] F \) 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 \( (\xi, \delta) \sqsubseteq (\xi', \delta') \) if \( \forall x \in V, \xi(x) \sqsubseteq \xi'(x) \) and \( \delta \sqsubseteq \delta' \)) and such an extension of \( \models F \).

**Example 4.2** Consider the TRS:

\[
0 + x \to x \\
s(x) + y \to s(x + y) \\
s(x) \leq s(y) \to x \leq y
\]

and the narrowing step

\[
\langle \varepsilon, [x, x + y], \rangle \leadsto \langle [x \to 0], [0, y] \rangle
\]

(\( \langle \cdot, \cdot \rangle \) denotes a 2-element list). We have

\[
\theta \models [\varepsilon, [x, x + y]] F = \langle \varepsilon, [x, \bot] \rangle
\]

and

\[
\theta \models [x \to 0], [0, y] F = \langle [x \to 0], [0, y] \rangle.
\]

Therefore, we do not get the desired increasing computation, because \( \varepsilon \not\sqsubseteq \{x \to 0\} \) and \( [x, \bot] \not\sqsubseteq [0, y] \).

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 \( \models_{FL} : \mathcal{T}(\Sigma_{\perp}, V) \to \mathcal{T}(\mathcal{C}_{\perp}) \) which interprets uninstantiated variables as least defined elements:

\[
\models_{FL} x = \bot \\
\models_{FL} \xi F = \xi \models \models_{FL} \bot = \bot
\]

\[
\models_{FL} c(\xi F) = \xi \models \models_{FL} \bot = \bot
\]

if \( c \in \mathcal{C} \)

\[
\models_{FL} \bot = \bot_{Valbot}(\bot F) \quad \text{and} \quad \models_{FL} \xi F = \bot_{Valbot} \circ (\xi F).
\]

**Example 4.3** (Continuing Example 4.2) Now,

\[
\models_{FL} [\varepsilon, [x, x + y]] F = \langle \bot_{Valbot}, [\bot, \bot] \rangle \\
\quad \subseteq \langle [x \to 0], [0, \bot] \rangle \\
\quad = \models_{FL} [x \to 0], [0, y] F
\]

i.e., in this case, \( \models_{FL} \) satisfies the desired property.

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

**Lemma 4.4** Let \( t, s \in \mathcal{T}(\Sigma_{\perp}, V) \). If \( \models_{F} s \subseteq \models_{F} t \), then \( \models_{FL} s \subseteq \models_{FL} t \).

**Proof.** Since \( \models_{FL} t = \bot_{Valbot}(\models_{FL} t) \) and \( \models_{FL} s = \bot_{Valbot}(\models_{FL} s) \), the conclusion follows by monotonicity of \( \bot_{Valbot} \). \( \square \)
Lemma 4.5 Let $t$ be a finite term and $\sigma$ be a substitution. Then $\llbracket t \rrbracket_{FL} \subseteq \llbracket \sigma(t) \rrbracket_{FL}$.

**Proof.** By structural induction. If $t$ is a variable, then $\llbracket t \rrbracket_{FL} = \bot \subseteq \llbracket \sigma(t) \rrbracket_{FL}$. If $t$ is a constant, $t = \sigma(t)$ and the conclusion follows. Let $t = f(t_1, \ldots, t_n)$. If $f \in \mathcal{F}$, then $\llbracket t \rrbracket_{FL} = \bot$ and the conclusion follows. If $f = c \in \mathcal{C}$, then $\llbracket c \rrbracket_{FL} = \llbracket c(c) \rrbracket_{FL}$. By I.H., $\llbracket t_i \rrbracket_{FL} \subseteq \llbracket \sigma(t_i) \rrbracket_{FL}$ for all $i, 1 \leq i \leq \alpha(c)$. Therefore, by definition of $\subseteq$, $\llbracket c(c) \rrbracket_{FL} = c(\llbracket c \rrbracket_{FL}) \subseteq c(\llbracket \sigma(c) \rrbracket_{FL}) = \llbracket \sigma(c) \rrbracket_{FL} = \llbracket \sigma(c) \rrbracket_{FL}$.

Lemma 4.6 Let $\sigma, \sigma'$ be substitutions. If $\sigma \leq \sigma'$, then $\llbracket \sigma \rrbracket_{FL} \subseteq \llbracket \sigma' \rrbracket_{FL}$.

**Proof.** If $\sigma \leq \sigma'$, there is $\theta$ such that $\sigma' = \theta \circ \sigma$. Thus, for all $x \in V$, $\sigma'(x) = \theta(\sigma(x))$. By Lemma 4.5, for all terms $t$, $\llbracket \sigma(t) \rrbracket_{FL} \subseteq \llbracket \theta(\sigma(t)) \rrbracket_{FL}$, i.e., $\llbracket \sigma \rrbracket_{FL} \subseteq \llbracket \sigma' \rrbracket_{FL}$.

The following proposition establishes that narrowing increases the current information of substitution/term pairs as given by $\llbracket \cdot \rrbracket_{FL}$.

**Proposition 4.7** Let $\mathcal{R}$ be a TRS. If $\langle \sigma, t \rangle \rightsquigarrow^* \langle \sigma', t' \rangle$, then $\llbracket \langle \sigma, t \rangle \rrbracket_{FL} \subseteq \llbracket \langle \sigma', t' \rangle \rrbracket_{FL}$.

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

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 prestruct 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 prestructs. A relation $f \subseteq P \times P'$ is an approximable mapping from $P$ to $P'$ if

1. $\bot \in f \bot'$. 

10
2. \( p \uparrow p' \) and \( p \uparrow q' \) imply \( p \uparrow (p' \sqcup q') \).

3. \( p \uparrow p', p \subseteq q, \) and \( q' \subseteq p' \) imply \( q \uparrow q' \).

The ideal completion \( (I(P), \subseteq, \{ \perp \}) \) of a precusl \( P \) is a domain (see [SLG94]). If \( P = (P, \subseteq, \sqcup, \perp) \) is a \text{cusi} \(^4\) (i.e., \( \subseteq \) is actually an ordering), then the mapping \([ \cdot ] : P \rightarrow I(P)\) that associates the principal ideal \( \{ p \downarrow \} \) to each \( p \in P \) is injective.

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

\[
\overline{f}(I) = \{ p' \in P' \mid \exists p \in I \quad p \uparrow p' \} = \bigcup_{p \in I} \{ p' \in P' \mid p \uparrow p' \}
\]

Note that, for all \( p \in I \), \( \{ p' \in P' \mid p \uparrow p' \} \) is an ideal (it is not empty because we always have \( \perp \uparrow \perp' \), and thus \( p \uparrow \perp' \) 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 \( p \uparrow p' \) and \( q' \subseteq p' \), we also have \( p \uparrow q' \), third condition again).

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

**Proof.** Immediate. \( \square \)

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

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

**Proof.** We note that \( \bigcup_{q \in \{ p \}} \{ p' \in P' \mid q \uparrow p' \} \subseteq \{ p' \in P' \mid p \uparrow p' \} \); indeed, since for all \( q \in \{ p \} \), we have that \( q \subseteq p \), and by using Definition 4.8 (third point), whenever \( q \uparrow p' \), we also have \( p \uparrow p' \). Thus, since it is obvious that \( \{ p' \in P' \mid p \uparrow p' \} \subseteq \bigcup_{q \in \{ p \}} \{ p' \in P' \mid q \uparrow p' \} \), we write

\[
\overline{f}(\{ p \}) = \bigcup_{q \in \{ p \}} \{ p' \in P' \mid q \uparrow p' \} = \{ p' \in P' \mid p \uparrow p' \} \]

\( \square \)

\(^4\)Conditional upper semilattice with least element, abbreviated \text{cusi} [SLG94].
Proposition 4.11 Let $P = (P, \subseteq, \sqcup, 
abla)$, $P' = (P', \subseteq, \sqcup', \nabla')$ be precusl's, $p \in P$ and $f \subseteq P \times P'$ be an approximable mapping from $P$ to $P'$. If $[p'] = \mathcal{T}(p)$ for some $p' \in P'$, then for all $q \in [p]$, whenever $q \neq q'$ for some $q' \in P'$, we have that $q' \nsubseteq p'$.

**Proof.** Immediate. □

The following proposition establishes that, if $\mathcal{T}$ sets a connection between finite elements of domains $\text{Id}(P)$ and $\text{Id}(P')$, then $f$ itself already connects these elements.

**Proposition 4.12** Let $P = (P, \subseteq, \sqcup, 
abla)$, $P' = (P', \subseteq, \sqcup', \nabla')$ be precusl's, $p \in P$ and $f \subseteq P \times P'$ be an approximable mapping from $P$ to $P'$. If $[p'] = \mathcal{T}(p)$ for some $p' \in P'$, then $p f p'$.

**Proof.** By definition of $\mathcal{T}$, $[p'] = \{q' \in P' \mid \exists q \in [p] \text{ s.t. } q f q'\}$. In particular, since $p' \in [p']$, there must be $q \in [p]$ such that $q f p'$. Since $q \subseteq p$, by Definition 4.8 (third condition), the conclusion follows. □

**Proposition 4.13** Let $P = (P, \subseteq, \sqcup, 
abla)$, $P' = (P', \subseteq, \sqcup', \nabla')$ be precusl's and $I$ be a set of indices. Let $f_a \subseteq P \times P'$ be approximable mappings for all $a \in I$. If $f = \bigcup_{a \in I} f_a$ is an approximable mapping, then $\mathcal{T} = \bigcup_{a \in I} \mathcal{T}_a$.

**Proof.** For all $I \in \text{Id}(P)$, we have:
\[
(\bigcup_{a \in I} \mathcal{T}_a)(I) = \bigcup_{a \in I} \mathcal{T}_a(I) = \bigcup_{a \in I} \{p' \in P' \mid \exists p \in I, p f_a p'\} = \{p' \in P' \mid \exists p \in P, p f p'\} = \mathcal{T}(I).
\]

□

**Proposition 4.14** Let $P = (P, \subseteq, \sqcup, 
abla)$, $P' = (P', \subseteq, \sqcup', \nabla')$ be precusl's and $I$ be a set of indices. Let $f_a \subseteq P \times P'$ be approximable mappings for all $a \in I$ such that $\{\mathcal{T}_a \mid a \in I\}$ is bounded. Let $f = \bigcup_{a \in I} f_a$ and $p \in P$. If $[p'] = f([p])$ for some $p' \in P'$, then there exists $a \in I$ such that $[p'] = \mathcal{T}_a([p])$.

**Proof.** Note that $f([p]) = \bigcup_{a \in I} \mathcal{T}_a([p]) = \bigcup_{a \in I} \mathcal{T}_a([p])$. Since $[p'] = \bigcup_{a \in I} \mathcal{T}_a([p])$, it follows that $p' \in \bigcup_{a \in I} \{q' \in P' \mid \exists q \in [p] q f_a q'\}$, i.e., there exists $a \in I$ such that $p' \in \{q' \in P' \mid \exists q \in [p] q f_a q'\} = \mathcal{T}_a([p])$. If $q' \subseteq p'$, then, by Definition 4.8, we have that, being $q \in [p]$ such that $q f_a q'$, we also have $q f_a q'$. Thus, $q' \in \mathcal{T}_a([p])$, i.e., $[p'] \subseteq \mathcal{T}_a([p])$. On the other hand, if $[p'] \nsubseteq \mathcal{T}_a([p])$, it is not possible that $[p'] = f([p])$. Hence, the conclusion follows. □
5 The narrowing space as an approximable mapping

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

It is easy to see that \((T(C_\perp), \sqsubseteq, \sqcup, \perp)\), where \( \sqsubseteq \) is the usual approximation ordering, is a precall in fact a call. Similarly, given a set \( W \subseteq V \) of variables, \((T(C_\perp)^W, \sqsubseteq, \sqcup, \perp^W_{\text{Value}})\), where \( \sqsubseteq \) is the pointwise extension of the ordering \( \sqsubseteq \) on \( T(C_\perp) \) to valuations \( \phi \in T(C_\perp)^W \), is also a call. Given a term \( t \), \textit{NDeriv}(t) is the set of narrowing derivations issued from \( t \). We associate a relation \( \text{Narr}^A(t) \subseteq T(C_\perp)^{Var(t)} \times T(C_\perp)^{Var(t)} \) to a given narrowing derivation \( A \in \text{NDeriv}(t) \).

**Definition 5.1** Given a term \( t \in T(C_\perp, V) \) and a narrowing derivation

\[
A : (\sigma_0, t_0) \sim (\sigma_1, t_1) \sim \cdots \sim (\sigma_n, t_n)
\]

we define \( \text{Narr}^A(t) = \sqcup_{0 \leq i \leq n} \text{Narr}^A(t) \) where:

\[
\text{Narr}^A_i(t) = \left\{ (\sigma, \delta) \in T(C_\perp)^{Var(t)} \times T(C_\perp) \mid \exists \phi \in T(C_\perp)^V (\{ \phi \circ \sigma_i \}_{FL} \sqsubseteq \delta \cup \{ \phi(t_i) \}_{FL}) \right\}
\]

where we assume that \( \text{Dom}(\phi) \cap \text{Dom}(\sigma_i) = \emptyset \) 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 \( T(C_\perp) \) and valuations (in \( T(C_\perp)^{Var(t)} \)) of variables in \( Var(t) \).

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

**Example 5.3** Consider the TRS \( R \) in Example 4.2 and term \( t = x + y \). For the narrowing derivations:

\[
A_1 : \{ x \mapsto x, y \mapsto y \}; x + y \quad \sim \quad \{ x \mapsto 0, y \mapsto y \}; y
\]

and

\[
A_2 : \{ x \mapsto x, y \mapsto y \}; x + y \quad \sim \quad \{ x \mapsto s(x'), y \mapsto y' \}; s(x' + y')
\]

\[
\sim \quad \{ x \mapsto s(0), y \mapsto y'' \}; s(y'')
\]

we show (part of) their semantic descriptions \( \text{Narr}^{A_1} \) and \( \text{Narr}^{A_2} \):

\[
\text{Narr}^{A_1}(t) = \{ (\sigma, \perp) \mid \sigma \in T^{\infty}(C_\perp)^{\langle x, y \rangle} \}
\]

\[
\text{Narr}^{A_2}(t) = \{ (\langle x \mapsto 0, y \mapsto \perp \rangle, \perp), (\langle x \mapsto 0, y \mapsto 0 \rangle, 0), (\langle x \mapsto 0, y \mapsto s(\perp) \rangle, \perp), (\langle x \mapsto 0, y \mapsto s(\perp) \rangle, s(\perp)), \ldots \}
\]
Let \( \phi \in \mathcal{T}(\Sigma) \) and \( \sigma \) be a substitution. Then, \( (\phi \circ \sigma)|_{FL} = \phi \circ (|\sigma|$.  

**Proof.** \( (\phi \circ \sigma)|_{FL} = \bot \circ Val \circ (\phi \circ \sigma)|{F} = \bot \circ Val \circ (\phi \circ | \sigma | \circ F = \bot \circ Val \circ (\phi \circ | \sigma |$. 

**Lemma 5.5** Let \( t \in \mathcal{T}(\Sigma) \) and \( \phi \in \mathcal{T}(\Sigma). \) If \( \phi(t)|_{FL} = \phi(t) \), then \( |\phi(t)|_{FL} = |\phi(t)\). 

**Proof.** By using Lemma 5.4, we have \( (\phi(t)|_{FL} = (\phi \circ | \sigma | \circ F(t) = \phi \circ | \sigma |$. 

**Proposition 5.6** Let \( \phi, \phi', \zeta \in \mathcal{T}(\Sigma) \) and \( \sigma \) be a substitution such that \( \text{Dom}(\sigma) \cap \text{Dom}(\phi) = \emptyset. \) If \( \{\phi \circ \sigma, \phi' \circ \sigma\} \) is bounded by \( \zeta \), then \( \{\phi, \phi'\} \) is bounded and \( \phi \cup \phi' \circ \sigma \subseteq \zeta \). 

**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. \( x \notin \text{Dom}(\sigma) \), i.e. \( \sigma(x) = \bot \). By hypothesis, we have \( \phi(\sigma(x)) = \phi(x) \subseteq \zeta(x) \) and \( \phi'(\sigma(x)) = \phi'(x) \subseteq \zeta(x) \).

2. \( x \in \text{Dom}(\sigma) \). Then, by hypothesis, \( x \notin \text{Dom}(\phi) \), i.e. \( \phi(x) = \bot \). Thus, \( \phi(x) \subseteq \phi'(x) \).

Thus, since \( \mathcal{T}(\Sigma) \) is a csl, \( \phi \cup \phi' \) always exists. Now we have \( (\phi \cup \phi') \circ \sigma = (\phi \circ \sigma) \cup (\phi' \circ \sigma) \subseteq \zeta \).

**Example 5.7** Without imposing that \( \text{Dom}(\sigma) \cap \text{Dom}(\phi) = \emptyset \), Proposition 5.6 could be false. For instance, let \( \sigma(x) = a, \zeta(x) = a, \phi(x) = a, \) and \( \phi'(x) = b \) for a given variable \( x \) and arbitrary constants \( a \) and \( b \). Then, \( \phi(\sigma(x)) = \phi'(\sigma(x)) = \zeta(x) = a, \) i.e., \( \{\phi \circ \sigma, \phi' \circ \sigma\} \) is bounded by \( \zeta \), but \( \{\phi(x), \phi'(x)\} = \{a, b\} \) is not bounded.
Lemma 5.8 Let \( t \) be a finite term and \( \sigma, \sigma' \) be substitutions such that \( \sigma \subseteq \sigma' \). Then, \( \| \sigma(t) \|_{FL} \subseteq \| \sigma'(t) \|_{FL} \).

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

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

Proof. Let

\[
A : \langle \xi_{\text{var}(t)}, t \rangle \leadsto \langle \sigma_{1}, t_{1} \rangle \leadsto \cdots \leadsto \langle \sigma_{n-1}, t_{n-1} \rangle \leadsto \langle \sigma_{n}, t_{n} \rangle
\]

We abbreviate \( N\text{arr}^{A}(t) \) by \( \text{m} \). 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 \( \text{A} \) starting from \( t \),

\[
N\text{arr}^{A}(t) = \{ \langle \zeta, \delta \rangle \mid \exists \varphi \in T(C_{\bot})^{V}, (\varphi \circ \xi_{\text{var}(t)}) \|_{FL} \subseteq \zeta \land \delta \subseteq \| \varphi(t) \|_{FL} \subseteq m \}.
\]

We have that \( (\varphi \circ \xi_{\text{var}(t)}) \|_{FL} = \varphi \circ \xi_{\text{var}(t)} \|_{F} = \varphi \circ \xi_{\text{var}(t)} = \varphi_{\text{var}(t)} \). In particular, by choosing \( \varphi = \bot_{\text{Valuat}} \), \( \zeta = \bot_{\text{Valuat}} \) (note that \( \text{Dom}(\bot_{\text{Valuat}}) = \emptyset \)), and \( \delta = \bot \), we obtain \( \xi_{\text{var}(t)} = (\bot_{\text{Valuat}} \circ \xi_{\text{var}(t)}) \|_{FL} \subseteq \bot_{\text{Valuat}} \) and \( \delta = \bot \subseteq \| \xi_{\text{var}(t)} \|_{FL} \), i.e., \( \bot_{\text{Valuat}} \) and \( \bot \).

2. Let \( \zeta, m, \delta \) and \( \zeta, m, \delta' \) be. By definition of \( m \), there are \( \phi_{i}, \phi_{j} \in T(C_{\bot})^{V} \) such that \( \phi_{i} \circ (t_{1}) \|_{F} \subseteq \zeta \) and \( \phi_{j} \circ (t_{j}) \|_{F} \subseteq \zeta \) for some \( 0 \leq i < j < n \). Since \( i < j \), there exists an idempotent substitution \( \theta : \text{Var}(t_{i}) \rightarrow T(\Sigma, V) \) 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 \( \phi_{i} \circ (t_{1}) \|_{F} \subseteq \zeta \) and \( \phi_{j} \circ (t_{j}) \|_{F} = \phi_{j} \circ \theta \circ \sigma_{i} \|_{F} \subseteq \zeta \). Let us show that \( \phi_{i} \circ (t_{j}) \|_{F} \subseteq \zeta \). Let \( x \in \text{Var}(t) \) be such that \( x \not\in \text{Dom}(\| t_{j} \|_{F}) \). Since \( \text{Dom}(\| t_{i} \|_{F}) \subseteq \text{Dom}(\| t_{j} \|_{F}) \), it follows that \( x \not\in \text{Dom}(\| t_{j} \|_{F}) \); thus, \( \phi_{i}(\| t_{j}(x) \|_{F}) = \phi_{i}(x) = \phi_{i}(\| t_{i}(x) \|_{F}) \subseteq \zeta(x) \). If \( x \in \text{Dom}(\| t_{j} \|_{F}) \), then, by using the fact that \( \sigma_{j} = \theta \circ \sigma_{i} \), we distinguish two cases:

   (a) If \( \text{Var}(\| t_{i}(x) \|_{F}) \cap \text{Dom}(\theta) = \emptyset \), then \( \| t_{i}(x) \|_{F} = \| t_{j}(x) \|_{F} \); hence, \( \phi_{i}(\| t_{j}(x) \|_{F}) \subseteq \zeta(x) \).

   (b) If \( \text{Var}(\| t_{i}(x) \|_{F}) \cap \text{Dom}(\theta) \neq \emptyset \), then \( \| t_{j}(x) \|_{F} = \theta(\| t_{i}(x) \|_{F}) \) can contain variables which are already present in \( \| t_{i}(x) \|_{F} \) (i.e., variables \( y \) with \( y \not\in \text{Dom}(\theta) \)) for which condition \( \phi_{i} \circ (t_{i}) \|_{F} \subseteq \zeta \) ensures the desired result. For the other variables, we can assume, w.l.o.g., that \( \phi_{i} \) does not essentially modify anything (i.e., we can assume that \( \text{Rng}(\theta) \cap \text{Dom}(\phi_{i}) = \emptyset \)). Hence, the condition \( \bot \circ \text{Valuat} \circ (t_{j}) \|_{F} \subseteq \zeta \) (an easy consequence of \( \phi_{j} \circ (t_{j}) \|_{F} \subseteq \zeta \)) ensures the desired result.
Thus, since \( \phi_i \circ \{\tau_j\}_F \subseteq \varsigma \) and \( \phi_j \circ \{\tau_j\}_F \subseteq \varsigma \), by Proposition 5.6 (note that \( \text{Dom}(\{\tau_j\}_F) \cap \text{Dom}(\phi_j) = \emptyset \)), \( \{\phi_i, \phi_j\} \) is bounded by \( \phi = \phi_i \cup \phi_j \) and \( \phi \circ \{\tau_j\}_F \subseteq \varsigma \). By definition of \( m \), we also have \( \delta \subseteq (\phi_i(t_i))_F \subseteq \varsigma \) and \( \delta' \subseteq (\phi_j(t_j))_F \subseteq \varsigma \). By Proposition 4.7, \( (\phi_i)_F \subseteq (\phi_i(t_i))_F \subseteq \varsigma \). By using the fact that both \( \phi_i \circ \{\tau_i\}_F \) and \( \phi_j \circ \{\tau_j\}_F \) are bounded by \( \varsigma \), we conclude that \( (\phi_i(t_i))_F \subseteq (\phi_i(t_i))_F \subseteq \varsigma \). By Lemma 5.8, we obtain

\[
\delta \subseteq (\phi_i(t_i))_F \subseteq (\phi_i(t_j))_F \subseteq (\phi_j(t_j))_F \subseteq (\phi(t_j))_F
\]

By Lemma 5.8 again,

\[
\delta' \subseteq (\phi_j(t_j))_F \subseteq (\phi(t_j))_F
\]

Thus, \( \{\delta, \delta'\} \) is bounded, and \( \delta \uplus \delta' \subseteq (\phi(t_j))_F \). Since \( \phi \circ (\tau_j)_F \subseteq \varsigma \), by definition of \( m \), we have \( \varsigma \subseteq (\delta \uplus \delta') \).

3. Let \( \varsigma \) be \( \delta, \varsigma \subseteq \varsigma' \), and \( \delta' \subseteq \delta \). Thus, there is \( \phi \in \mathcal{T}(C_{\parallel}) \) and \( \sigma_i \), \( 0 \leq i \leq n \) such that \( \phi \circ \{\sigma_i\}_F \subseteq \varsigma \) and \( \delta \subseteq (\phi(t_i))_F \). Since \( \phi \circ \{\sigma_i\}_F \subseteq \varsigma \subseteq \varsigma' \) and \( \delta' \subseteq \delta \subseteq (\phi(t_i))_F \), \( \varsigma \) \( \delta' \) holds by definition of \( m \).

\[\Box\]

**Definition 5.10** Given a term \( t \in \mathcal{T}_v \), we define the relation \( \text{Narr}(t) \subseteq \mathcal{T}_v \times \mathcal{T}_v \) to be \( \text{Narr}(t) = \bigcup_{\alpha \in \text{NDeriv}(t)} \text{Narr}^\alpha(t) \).

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

**Example 5.11** Consider the TRS:

\[
\begin{array}{ccl}
f(f(x)) & \rightarrow & a \\
c & \rightarrow & b
\end{array}
\]

and \( A : \langle \varepsilon_{[x]}, t \rangle = \langle \{ x \mapsto x, f(x) \} \sim \{ x \mapsto f(x) \}, a \rangle \). If \( m = \text{Narr}^\alpha(t) \), then \( \{ x \mapsto a \} \subseteq \{ x \mapsto f(x) \} \subseteq \{ x \mapsto a \} \). Thus, \( \text{Narr}^\alpha(t) \subseteq \{ x \mapsto a \} \subseteq \{ x \mapsto a \} \). However, \( \{ x \mapsto a \}(t) = f(a) \not\in a \).

The problem here is that \( \| \_ \|_F \) identifies (as \( \bot \)) parts of the bindings \( \sigma(x) \) of a computed substitution \( \sigma \) 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) \( mgv \)'s as unifiers for the narrowing process, the substitutions which we deal with are linear constructor substitutions, i.e., for all narrowing derivations \( \langle \varepsilon_{[x]}, t \rangle \sim^* \langle \sigma, a \rangle \) and \( x \in \text{Var}(t), \sigma(x) \) is a constructor term and \( \{ \sigma(x) | x \in \text{Dom}(\sigma) \} \) 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, \( (\sigma)_F = \sigma, (\sigma)_F = (\sigma)_F = \bigcup_{\text{Valuat} \circ (\sigma)_F = \bigcup_{\text{Valuat} \circ \sigma} \), and we have the following result.
Proposition 5.12 Let $\sigma$ be a linear constructor substitution and $\phi, \zeta \in T(C_\bot)^V$ be such that $\text{Dom}(\zeta) \cap \text{Rng}(\sigma) = \emptyset$. If $\phi \circ \sigma \subseteq \zeta$, then there exists $\phi' \in T(C_\bot)^V$ such that $\phi' \subseteq \phi$ and $\phi \circ \sigma = \zeta$.

Proof. Let $x \in V$. Since $\phi \circ \sigma \subseteq \zeta$, we have that $\phi(\sigma(x)) \subseteq \zeta(x)$. We consider two cases:

1. If $\sigma(x) \neq x$ (i.e., $x \notin \text{Dom}(\sigma)$), then we distinguish, again, two cases:
   
   (a) If $\text{Var}(\sigma(x)) = \emptyset$, then, since $\sigma$ is a constructor substitution, we have that $\sigma(x) \in T(C)$. Hence, $\phi(\sigma(x)) = \sigma(x) \subseteq \zeta(x)$ which, in fact, means that $\sigma(x) = \zeta(x)$. Thus, for all $\phi' \in T(C_\bot)^V$, $\phi'(\sigma(x)) = \sigma(x) = \zeta(x)$.

   (b) If $\text{Var}(\sigma(x)) \neq \emptyset$, then there exists a context $C[\ldots \zeta[\ldots]$ such that $\phi(\sigma(x)) = C[\phi(y_1), \ldots, \phi(y_n)] \subseteq C[\delta_1, \ldots, \delta_n] = \zeta(x)$ where $\text{Var}(\sigma(x)) = \ldots \zeta$ has been replaced by $\emptyset$; this is correct because $\sigma$ is a constructor substitution (and hence $\sigma(x)$ is a constructor term without $\bot$’s) and $\sigma$ is linear (thus there are $n$ different variables $y_1, \ldots, y_n$ in $\text{Var}(\sigma(x))$ 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 $\text{Dom}(\zeta) \cap \text{Rng}(\sigma) = \emptyset$ ensures that no further collisions would arise between these assignments for $\phi'$ and those given in case 2 below.

2. If $\sigma(x) = x$ (i.e., $x \notin \text{Dom}(\sigma)$), then we can just let $\phi'(x) = \zeta(x)$.

\[ \square \]

Note that linearity of $\sigma$ is necessary for ensuring this result.

Example 5.13 Let $\sigma = \{ u \mapsto f(x, y), v \mapsto f(x, z) \}$, $\phi = \bot_{\text{val}}$, and $\zeta = \{ u \mapsto f(\bot, \bot), v \mapsto f(\bot, \bot) \}$. Clearly, $\phi \circ \sigma = \{ u \mapsto f(\bot, \bot), v \mapsto f(\bot, \bot) \} \subseteq \zeta$. However, there is no $\phi'$ such that $\phi' \circ \sigma = \zeta$ because it would be necessary that, simultaneously, $\phi'(x) = \bot$ and $\phi'(x) = \zeta$.

Moreover, the condition $\text{Dom}(\zeta) \cap \text{Rng}(\sigma) = \emptyset$ is also necessary for ensuring the result\(^5\).

Example 5.14 Let $\sigma = \{ x \mapsto s(y), y \mapsto y, z \mapsto z \}$, $\phi = \bot_{\text{val}}$, and $\zeta = \{ x \mapsto s(\bot), y \mapsto \bot, z \mapsto \bot \}$. Note that $\text{Dom}(\zeta) \cap \text{Rng}(\sigma) = \{ y \}$. Clearly, $\phi \circ \sigma = \{ x \mapsto s(\bot), y \mapsto \bot, z \mapsto \bot \} \subseteq \zeta$. However, there is no $\phi'$ such that $\phi' \circ \sigma = \zeta$ because it would be necessary that, simultaneously, $\phi'(y) = \bot$ and $\phi'(y) = \bot$.

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

\[ \text{Narr}^L_t = \{ \langle \phi, \delta \rangle \mid \exists \phi' \in T(C_\bot)^V. (\phi \circ \sigma_t \subseteq \phi' \land \delta \subseteq \{ \phi(t_i) \}_{i=1}^n) \}
\]

\[ = \{ \langle \phi, \delta \rangle \mid \exists \phi' \in T(C_\bot)^V. (\phi \circ \sigma_t = \phi' \land \delta \subseteq \{ \phi(t_i) \}_{i=1}^n) \} \]

\(^5\)Example 5.14 was suggested by a referee.
This is easily proved correct when considering the special properties of (partial) computed substitutions \( \sigma \) 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 \( R \) be a left-linear, confluent CB-TRS and \( t \) be a term. Then, \( \text{Narr}(t) \) is an approximable mapping.

**Proof.** We abbreviate \( \text{Narr}(t) \) by \( m \). Then, we check the conditions of Definition 4.8. Again, we use Lemma 5.4 to simplify the expressions.

1. Since \( \langle \varepsilon|_{\text{Var}(t)}, t \rangle \sim^* \langle \varepsilon|_{\text{Var}(t)}, t \rangle \), we have that \( \bot_{\text{Valuat}} \circ \varepsilon|_{\text{Var}(t)} = \bot_{\text{Valuat}} \subseteq \langle \varepsilon|_{\text{Var}(t)} \rangle_{FL} \), i.e., \( \bot_{\text{Valuat}} m \bot \).

2. Let \( \varsigma, m, \delta \), and \( \varsigma, m, \delta' \). By definition of \( m \), there are narrowing derivations \( \langle \varepsilon|_{\text{Var}(t)}, t \rangle \sim^* \langle \sigma_1, s_1 \rangle, \langle \varepsilon|_{\text{Var}(t)}, t \rangle \sim^* \langle \sigma_2, s_2 \rangle \) and \( \phi_1, \phi_2 \in \mathcal{T}(C_\downarrow)^V \) such that \( \phi_1 \circ \langle \sigma_1 \rangle \subseteq \varsigma \) and \( \phi_2 \circ \langle \sigma_2 \rangle \subseteq \varsigma \). By Proposition 5.12, there exist \( \theta_1, \theta_2 \in \mathcal{T}(C_\downarrow)^V \) such that \( \phi_1 \subseteq \theta_1, \phi_2 \subseteq \theta_2 \), and \( \varsigma = \theta_1 \circ \langle \sigma_1 \rangle \subseteq \theta_2 \circ \langle \sigma_2 \rangle \). By confluence, there is a term \( s \) such that \( \theta_1(s_1) \rightarrow^* \theta_2(s_2) \). By Proposition 3.1 and Lemma 4.4, \( \langle \theta_1(s_1) \rangle_{FL}, \langle \theta_2(s_2) \rangle_{FL} \subseteq \langle s \rangle_{FL} \). By Hullot’s Theorem \( [\text{Hul80}] \), there is \( \sigma \leq \varsigma \) such that \( \langle \varepsilon|_{\text{Var}(t)}, t \rangle \sim^* \langle \sigma, s' \rangle \) and \( s' \subseteq s \), i.e., there exists a substitution \( \phi \) such that \( \varsigma = \phi \circ \sigma \) and \( s = \phi(s') \).

By hypothesis and by Lemma 5.8, \( \delta \subseteq \langle \phi_1(s_1) \rangle_{FL} \subseteq \langle \theta_1(s_1) \rangle_{FL} \) and \( \delta' \subseteq \langle \phi_2(s_2) \rangle_{FL} \subseteq \langle \theta_2(s_2) \rangle_{FL} \). Since \( \langle \theta_1(s_1) \rangle_{FL}, \langle \theta_2(s_2) \rangle_{FL} \subseteq \langle s \rangle_{FL} \), it follows that \( \langle \delta, \delta' \rangle \) is bounded by \( \langle s \rangle_{FL} \), i.e., \( \langle \delta, \delta' \rangle \) is consistent. Since \( \mathcal{T}(C_\downarrow) \) is a causal, \( \delta \cup \delta' \) is the lub of \( \delta \) and \( \delta' \). Hence, since \( \phi \circ \sigma \subseteq \phi(s) \), \( \langle \sigma \rangle_{FL} = \langle s \rangle_{FL} \subseteq \langle \phi(s) \rangle_{FL} \), by Definition 5.10, \( \varsigma, m, \delta \cup \delta' \).

3. We need to prove that, if \( \varsigma, m, \delta \), \( \varsigma, m, \delta' \), then \( \varsigma, m, \delta \).

Since \( \varsigma, m, \delta \), there is a narrowing derivation \( \langle \varepsilon|_{\text{Var}(t)}, t \rangle \sim^* \langle \sigma, s \rangle \) and a substitution \( \phi \in \mathcal{T}(C_\downarrow)^V \) such that \( \phi \circ \langle \sigma \rangle \subseteq \varsigma \). If \( \varsigma \subseteq \varsigma' \), then \( \phi \circ \langle \sigma \rangle_{FL} \subseteq \varsigma' \). On the other hand, \( \delta \subseteq \delta \subseteq \langle \phi(s) \rangle_{FL} \). Hence, by definition of \( m \), \( \varsigma, m, \delta \).
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 \( \text{Narr}(t) = \bigsqcup_{A \in N\text{Deriv}(t)} \text{Narr}^A(t) \).

**Proof.** Proposition 5.15, Proposition 5.9, and Proposition 4.13. \( \square \)

Thus, we define the semantic function

\[
\text{CNarr}^\infty : \mathcal{T}(\Sigma, V) \to [\mathcal{T}^\infty(\mathcal{C})^V \to \mathcal{T}^\infty(\mathcal{C})]
\]

as follows:

\[
\text{CNarr}^\infty(t) = \overline{\text{Narr}(t)}
\]

i.e., \( \text{CNarr}^\infty(t) \) is the continuous mapping associated to the approximable mapping \( \text{Narr}(t) \) 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 \( t \in \mathcal{T}(\Sigma, V) \), \( \zeta \in \mathcal{T}(\mathcal{C})^V \), \( \text{CNarr}^\infty(t) \zeta = \text{Crew}^\infty(\zeta(t)) \).

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

\[
\text{CNarr}^\infty(t)[\zeta] = \{ \delta \mid \zeta \text{ Narr}(t)[\delta] \} = \bigcup_{A \in N\text{Deriv}(t)} \{ \delta \mid \zeta \text{ Narr}^A(t)[\delta] \}
\]

For each narrowing derivation

\[
\mathcal{A} : \langle \psi_{\text{Narr}(t)}, t \rangle \rightsquigarrow \langle \sigma_0, t_0 \rangle \rightsquigarrow \langle \sigma_1, t_1 \rangle \rightsquigarrow \cdots \rightsquigarrow \langle \sigma_n, t_n \rangle
\]

such that \( \zeta = \phi \circ \sigma_i \) for some \( 1 \leq i \leq n \) and \( \delta \subseteq \{ \psi(t_i) \}_{FL} \), by Hullot’s Theorem, we have \( \sigma_i(t_i) \rightarrow \phi(t_i) \). By stability \( \zeta \rightarrow \phi(t_i) \). Thus, since \( \phi \in \mathcal{T}(\mathcal{C})^V \), we have that \( \{ \psi(t_i) \}_{FL} \subseteq \{ \psi(t_i) \}_{FL} \in \text{Rew}(\zeta(t)) \) and, in fact, \( \text{CNarr}^\infty(t)[\zeta] \subseteq \text{Rew}(\zeta(t)) \). In order to prove that \( \text{Rew}(\zeta(t)) \subseteq \text{CNarr}^\infty(t)[\zeta] \), let us consider \( \delta \in \text{Rew}(\zeta(t)) \). Then there exists \( \{ \delta \}_{FL} \in \text{Rew}(\zeta(t)) \) such that \( \delta \subseteq \{ \delta \}_{FL} \). Hence, \( \zeta(t) \rightarrow s \) and there is a narrowing derivation \( \langle \psi_{\text{Narr}(t)}, t \rangle \rightsquigarrow \langle \sigma, s' \rangle \) with \( \zeta = \phi \circ \sigma \) for some \( \phi \in \mathcal{T}(\mathcal{C})^V \) and \( s = \phi(s') \). Therefore, since \( \zeta(t), s \in \mathcal{T}(\Sigma) \), we have that \( \{ \delta \}_{FL} = \{ \psi \}_{FL} \subseteq \{ \psi(s') \}_{FL} = \phi(s') \). Thus, \( \text{Narr}(t)[\phi(s')] \) and, since \( \text{Narr}(t) \) is an approximable mapping and \( \delta \subseteq \{ \delta \}_{FL} \), we have \( \zeta \text{ Narr}(t)[\delta] \), i.e., \( \text{Rew}(\zeta(t)) \subseteq \text{CNarr}^\infty(t)[\zeta] \). \( \square \)
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 \( NDeriv_\mathcal{N}(t) \subseteq NDeriv(t) \) of derivations which start from \( t \) and conform to \( \mathcal{N} \). By Proposition 5.9, each \( A \in NDeriv_\mathcal{N}(t) \) defines an approximable mapping \( \text{Narr}^A(t) \) which is obviously contained in \( \text{Narr}(t) \). By Proposition 4.9 (when we consider left-linear, confluent CB-TRSs), \( \text{Narr}^A(t) \subseteq \text{Narr}(t) = CNarr^\infty(t) \). Therefore, \( \{\text{Narr}^A(t) \mid A \in NDeriv_\mathcal{N}(t)\} \) is bounded by \( CNarr^\infty(t) \). Since \( [\mathcal{T}^\infty(\mathcal{C}_\bot), V] \sim [\mathcal{T}^\infty(\mathcal{C}_\bot), V] \) 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

\[
CNarr^\infty(t) = \bigsqcup \{\text{Narr}^A(t) \mid A \in NDeriv_\mathcal{N}(t)\}
\]

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

**Remark 5.18** Narrowing is able to yield the graph of a function \( f \) by computing \( CNarr^\infty(f(\mathcal{X})) \), where \( x_1, \ldots, x_m \) 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 \( \text{Rew}(t) \) and \( \text{Narr}(t) \):

**Rewriting**
- \( \text{Rew}(t) \) is a set of partial constructor terms \( \delta \in \mathcal{T}(\mathcal{C}_\bot, V) \).
- \( \text{Rew}(t) \) is a directed set.
- The limit \( CRew^\infty(t) \) of \( \text{Rew}(t) \) within the domain \( \mathcal{T}^\infty(\mathcal{C}_\bot, V) \) is a (possibly infinite) value.

**Narrowing**
- \( \text{Narr}(t) \) is a set of pairs \( (\varsigma, \delta) \), where \( \varsigma \) is a valuation on \( \mathcal{T}(\mathcal{C}_\bot) \) and \( \delta \in \mathcal{T}(\mathcal{C}_\bot) \).
- \( \text{Narr}(t) \) is an approximable mapping.
- The ‘limit’ of \( \text{Narr}(t) \) is a continuous mapping \( CNarr^\infty(t) \) from valuations to (finite) 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 \( R \) be a confluent TRS, \( t \in T(\Sigma, V) \), and \( \delta \in T(\mathcal{C}, V) \). Then, \( \delta = \text{CRew}^\infty(t) \) if and only if \( t \rightarrow^* \delta \).

**Proof.** If \( t \rightarrow^* \delta \), then \( \|\delta\|_F = \delta \in \text{Rew}(t) \). Since \( \delta \) is maximal and, by Proposition 3.2, \( \text{Rew}(t) \) is directed, it follows that \( \delta = \text{CRew}^\infty(t) \). The opposite statement follows a similar reasoning. \( \square \)

**Proposition 6.2** Let \( R \) be a TRS, \( t \in T(\Sigma, V) \), \( F \) be a rewriting strategy, and \( \delta \in T(\mathcal{C}, V) \). Then, \( \delta = \text{CRew}_F^\infty(t) \) if and only if \( t \rightarrow^F \delta \).

**Proof.** Similar to Proposition 6.1. \( \square \)

Concerning narrowing computations, we have the following result.

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

1. For every narrowing derivation \( \langle \vartheta[\varsigma(t)], t \rangle \rightarrow^* \langle \sigma, s \rangle \) such that \( \phi \circ \sigma = \varsigma \) for some \( \phi \in T(\mathcal{C})^V \), we have \( \|\phi(s)\|_F \subseteq \delta \).

2. If \( \delta \in T(\mathcal{C})^V \), there exists a narrowing derivation \( \langle \vartheta[\varsigma(t)], t \rangle \rightarrow^* \langle \sigma, s \rangle \) and \( \phi \in T(\mathcal{C})^V \) such that \( \phi \circ \sigma = \varsigma \) and \( \delta = \|\phi(s)\|_F \).

3. If \( \delta \in T(\mathcal{C})^V \), then there exists a narrowing derivation \( \langle \vartheta[\varsigma(t)], t \rangle \rightarrow^* \langle \sigma, s \rangle \) and \( \phi \in T(\mathcal{C})^V \) such that \( s \in T(\mathcal{C}, V) \), \( \phi \circ \sigma = \varsigma \), and \( \delta = \phi(s) \).

**Proof.**

1. If \( A : \langle \vartheta[\varsigma(t)], t \rangle \rightarrow^* \langle \sigma, s \rangle \) is such that \( \phi \circ \sigma = \varsigma \), by definition of \( \text{Narr}^A(t) \), we have \( \varsigma \in \text{Narr}^A(t) \{\phi(s)\}_F \), i.e., \( \varsigma \in \text{Narr}(t) \{\phi(s)\}_F \). By Proposition 4.11, the conclusion follows.

2. By Proposition 4.12, we have that \( \varsigma \in \text{Narr}(t) \delta \). Thus, by definition of \( \text{Narr}(t) \), there is a narrowing derivation \( A : \langle \vartheta[\varsigma(t)], t \rangle \rightarrow^* \langle \sigma, s \rangle \) such that \( \varsigma \in \text{Narr}^A(t) \delta \). Hence, there exists \( \phi \in T(\mathcal{C})^V \) such that \( \phi \circ \sigma = \varsigma \) and \( \delta \subseteq \|\phi(s)\|_F \). Using (1), we conclude \( \delta = \|\phi(s)\|_F \).
3. By using (2), we conclude that there exists a narrowing derivation $\langle \varepsilon, \sigma, s \rangle \rightarrow^* \langle \sigma, s \rangle$ and $\delta \in T(C_V)$ such that $\vdash \sigma = \varepsilon$ and $\delta = \varphi(s)$. Assume that $s \notin T(C_V)$. Then, there exists a defined symbol $f \in F$ in $s$. Then, $\bot$ occurs in $\delta \rightarrow \varphi(s) F_L$ thus contradicting the fact that $\delta \in T(C_V)$.

\[ \square \]

Proposition 6.3(1) expresses that, given a (finite) valuation $\varsigma$, we can use any narrowing derivation starting from a term $t$ that computes a substitution more general than $\varsigma$ to approximate the value $\delta$ that, according to the semantic interpretation of $t$ as a non-ground value, corresponds to $\varsigma$. 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 conveyed by the narrowing semantics by introducing a small modification on it.

**Definition 6.4** Given a term $t \in T(S, V)$, and a narrowing derivation

$$A : \langle \varepsilon, \sigma(t), t \rangle \rightarrow^* \langle \sigma_0, t_0 \rangle \rightarrow \cdots \rightarrow \langle \sigma_{n-1}, t_{n-1} \rangle \rightarrow \langle \sigma_n, t_n \rangle$$

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

$$BNavr^A(t) = \{ \langle \varsigma, \delta \rangle \in T(C_V)^* \times T(C_V) \mid \varphi(t) F_L \subseteq \varsigma \wedge \delta \subseteq \varphi(t) F_L \}$$

**Proposition 6.5** Let $R$ 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

$$A : \langle \varepsilon, \sigma(t), t \rangle \rightarrow^* \langle \sigma_0, t_0 \rangle \rightarrow \cdots \rightarrow \langle \sigma_{n-1}, t_{n-1} \rangle \rightarrow \langle \sigma_n, t_n \rangle$$

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 $t$,

$$BNarr^A = \{ \langle \varsigma, \delta \rangle \mid \varphi(t) F_L \subseteq \varsigma \wedge \delta \subseteq \varphi(t) F_L \} \subseteq m.$$ 

We have that $\varphi(t) F_L = \mathcal{V}_{\text{Var} \text{t}}$ and we obtain $\mathcal{V}_{\text{Var} \text{t}} \mathcal{V}_{\text{Val} \text{t}} = \varsigma$, and $\delta = \bot \subseteq \varphi(t) F_L$, i.e., $\bot \mathcal{V}_{\text{Val} \text{t}} m \bot$.

2. Let $\varsigma$, $m$, $\delta$ and $\varsigma$, $m$, $\delta$. By definition of $m$, there are $\sigma_i, \sigma_j$, such that $\varphi(t) F_L \subseteq \varsigma$, $\delta \subseteq \varphi(t) F_L$, $\sigma_j F_L \subseteq \varsigma$, and $\delta \subseteq \varphi(t) F_L$ for some $0 \leq i \leq j \leq n$. By Proposition 4.7, $\varphi(t) F_L \subseteq \varphi(t) F_L$, i.e., $\delta, \delta'$ is bounded by $\varphi(t) F_L$. Thus, $\delta \cup \delta' \subseteq \varphi(t) F_L$ and, by definition of $m$, $\delta \cup \delta' = \bot$.

3. Let $\varsigma$, $m$, $\delta$, $\varsigma'$, and $\delta'$. Thus, there is $\sigma_i$, $0 \leq i \leq n$ such that $\varphi(t) F_L \subseteq \varsigma$ and $\delta \subseteq \varphi(t) F_L$. Since $\varphi(t) F_L \subseteq \varsigma$, $\delta'$ and $\delta' \subseteq \varphi(t) F_L$, by definition of $m$, we also have that $\varsigma'$ $m$, $\delta'$.  22
Since each $BNarr_1^4(t)$ is a special case of $Narr_1^4(t)$, in which only $\phi = \bot_{\text{valsat}}$ is allowed, we have that $BNarr^4(t) \subseteq Narr^4(t)$. Therefore, by Propositions 5.9 and 6.5, and using Proposition 4.9, we have that, for all terms $t$, $BNarr^4(t) \subseteq Narr^4(t)$. Whenever we consider left-linear, confluent CB-TRSs, Proposition 5.15 and Proposition 4.9 ensure that $\{BNarr^4(t) \mid A \in NDerr(t)\}$ is bounded by $CNarr^\infty(t)$. Thus, for left-linear, confluent CB-TRSs, we fix

$$BNarr^\infty(t) = \bigcup \{BNarr^4(t) \mid A \in NDerr(t)\}$$

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

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

$$A1: \ (\{ x \mapsto x, y \mapsto y \}, x + y) \rightsquigarrow (\{ x \mapsto 0, y \mapsto y' \}, y')$$

and

$$A2: \ (\{ x \mapsto x, y \mapsto y \}, x + y) \rightsquigarrow (\{ x \mapsto s(x'), y \mapsto y' \}, s(x' + y'))$$

$$\rightsquigarrow (\{ x \mapsto s(0), y \mapsto y'' \}, s(y'''))$$

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):

$$BNarr^{A1}_0(t) = \{ \langle \cdot, \bot \rangle \mid \cdot \in \mathcal{T}^\infty(\mathcal{C}_{\bot})^{[x,y]} \}$$

$$BNarr^{A1}_1(t) = \{ \langle x \mapsto 0, y \mapsto \bot \rangle, \bot, \langle x \mapsto 0, y \mapsto 0 \rangle, \bot, \langle x \mapsto 0, y \mapsto s(\bot) \rangle, \bot \}$$

$$BNarr^{A2}_0(t) = \{ \langle \cdot, \bot \rangle \mid \cdot \in \mathcal{T}^\infty(\mathcal{C}_{\bot})^{[x,y]} \}$$

$$BNarr^{A2}_1(t) = \{ \langle x \mapsto s(\bot), y \mapsto \bot \rangle, \bot, \langle x \mapsto s(\bot), y \mapsto 0 \rangle, s(\bot), \langle x \mapsto s(\bot), y \mapsto 0 \rangle, s(\bot), \langle x \mapsto s(0), y \mapsto 0 \rangle, \bot \}$$

$$BNarr^{A2}_2(t) = \{ \langle x \mapsto s(\bot), y \mapsto \bot \rangle, \bot, \langle x \mapsto s(\bot), y \mapsto 0 \rangle, s(\bot), \langle x \mapsto s(\bot), y \mapsto 0 \rangle, s(\bot), \langle x \mapsto s(0), y \mapsto 0 \rangle, \bot \}$$

$$BNarr^{A2}_3(t) = \{ \langle x \mapsto s(\bot), y \mapsto \bot \rangle, \bot, \langle x \mapsto s(\bot), y \mapsto 0 \rangle, s(\bot), \langle x \mapsto s(\bot), y \mapsto 0 \rangle, s(\bot), \langle x \mapsto s(0), y \mapsto 0 \rangle, \bot \}$$

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, $\varsigma \in \mathcal{T}(\downarrow)$, $m = BNarr^\infty(t)$, and $\delta = m(\varsigma)$.

1. For every narrowing derivation $\langle \epsilon_{\mathcal{P}}(t), t \rangle \leadsto^* \langle \sigma, s \rangle$ such that $\|s\|_{FL} \subseteq \varsigma$, it is $(\|s\|_{FL} \subseteq \delta$.

2. If $\delta \in \mathcal{T}(\downarrow)$, there exists a narrowing derivation $\langle \epsilon_{\mathcal{P}}(t), t \rangle \leadsto^* \langle \sigma, s \rangle$ such that $\phi \circ \sigma = \varsigma$ and $\delta = (\|s\|_{FL}$ for some $\phi \in \mathcal{T}(\downarrow)$.

**Proof.**

1. If $A : \langle \epsilon_{\mathcal{P}}(t), t \rangle \leadsto^* \langle \sigma, s \rangle$ is such that $\|s\|_{FL} \subseteq \varsigma$, then, by definition of $BNarr^\mathcal{A}(t)$, we have that $\varsigma$ $BNarr^\mathcal{A}(t)$ $\|s\|_{FL}$. Therefore, $(\|s\|_{FL} \subseteq BNarr^\mathcal{A}(t) \subseteq m(\varsigma) = \delta$.

2. By Proposition 4.14, there is a narrowing derivation $A : \langle \epsilon_{\mathcal{P}}(t), t \rangle \leadsto^* \langle \sigma, s \rangle$ such that $\delta = BNarr^\mathcal{A}(t) \subseteq \varsigma$. By Proposition 4.12, $\varsigma$ $BNarr^\mathcal{A}(t) \delta$. Since $(\|s\|_{FL} \subseteq \varsigma$, by using (1), we conclude $(\|s\|_{FL} \subseteq \delta$. By definition of $BNarr^\mathcal{A}(t)$, $\delta \subseteq (\|s\|_{FL} and the conclusion follows.

\[ \Box \]

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

**Proof.** Let $\delta' = m(\|\sigma\|_{FL})$. By Proposition 6.7(1), $(\|\sigma\|_{FL} \subseteq \delta'$. Since $\delta \in \mathcal{T}(\downarrow)$, $(\|\delta\|_{FL} = \delta; moreover, since $\delta$ is maximal, $\delta \nsubseteq \delta'$. 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

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

However, according to Example 5.3, we have that

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

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 \( \pi \) of a program \( P \) whose denotation \([P]\) is taken from a domain \( D \) (i.e., \([P] \in D\)) can be identified with a predicate \( \pi : D \to 2 \), where \( 2 \) is the two point domain \( 2 = \{\bot, \top\} \) ordered by \( \bot \subseteq \top \) (where \( \bot \) can be thought of as false and \( \top \) as true). A program \( P \) satisfies \( \pi \) if \( \pi([P]) = \top \) (alternatively, if \([P] \in \pi^{-1}(\top)\)). As usual in domain theory, we require continuity of \( \pi \) for achieving computability (or observability, see [Smy83, Vic89]). The set \([D \to 2]\) 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 \((X, \tau)\) 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]: \( X, \emptyset \in \tau \); 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 \( \bigcup A \in U \), then \( \exists x \in A. x \in U \) [SLG94].

Note that, when considering the Scott’s topology \((D, \tau_D)\) of a domain \( D \), the open set \( D \) denotes a trivial property which every program satisfies; \( \emptyset \), the least element of lattice \( \tau_D \), 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 \((D, \tau_D)\) and \((A, \tau_A)\) 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 \( \alpha^{-1} \), i.e., \( \alpha^{-1} : \tau_A \to \tau_D \) is a mapping from abstract properties (open sets of \( \tau_A \)) to concrete properties (open sets of \( \tau_D \)). 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 \( \tau_A \) plays the role of an abstract domain in the usual, lattice-based, abstract interpretation approaches. For instance, the Scott’s topology of \( 2 \) is given by \( \tau_2 = \{\emptyset, \{\top\}, 2\} \). 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 \( D = \mathcal{T}^\infty(\mathcal{L}_\bot) \). Observable properties of functional logic pro-
grams 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 \(\alpha_F\).

If \(\alpha_F : D \to A\) is strict (surjective, continuous), then \(\alpha_L\) is strict (surjective, continuous). Whenever \(\alpha_F\) is a continuous mapping from a domain \(D\) to \(2\), \(\alpha_F\) expresses, in fact, a single observable property \(\alpha^{-1}(\{1\})\) of \(D\). We can thought of \(\alpha_F\) as a functional property. Thus, Definition 7.1 associates an abstraction \(\alpha_L\) of \(D^V\) to a given property identified by \(\alpha_F\). Thus, each functional property induces a related set of logic properties which is a subtopology of \(\tau_{D^V}\). 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 \(\alpha_{D \to E}^S, \alpha_{D \to E}^L : (D \to E) \to (A \to B)\) of continuous mappings by [Abr90]

\[
\alpha_{D \to E}^S(f)(d) = \cup \{ (\alpha_E \circ f)(d') | \alpha_D(d') \sqsubseteq d \}, \quad \text{and} \quad \alpha_{D \to E}^L(f)(d) = \cap \{ (\alpha_E \circ f)(d') | \alpha_D(d') \sqsupseteq d \}
\]

where the following correctness result holds:

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

Consider an abstraction \(\alpha_E : E \to 2\) 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 \(f^S \circ \alpha_D \sqsubseteq \alpha_E \circ f\) ensures that, for all \(x \in D\), whenever the abstract computation \(f^S(\alpha_D(x))\) yields \(\top\), the concrete computation \(f(x)\) does not satisfy the property \(\alpha_E\), i.e., \(\alpha_E(f(x)) = \bot\). 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 \(\bot\), the concrete computation \(f(x)\) actually satisfies \(\alpha_E\), 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 $\text{ngw}$'s reveals connections between apparently disconnected analyses. Consider $h_t : T^\infty(C_\bot) \rightarrow 2$ defined by

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

and let $h_g : T^\infty(C_\bot)^V \rightarrow 2^V$ 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 2 is $\{T\}$. By continuity of $h_t$, $h_t^{-1}(\{T\})$ 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:

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

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

On the other hand, each open set of $2^V$ is (isomorphic to) an upward closed collection of sets of variables ordered by inclusion. In this case, $h_t^{-1}(U)$ 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, $2^V$ is a well-known abstract domain for groundness analysis in logic programming [JS87].

If $C$ has constructors with positive arity, then $h_t^{-1}(\{T\})$ 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:

\[
\begin{align*}
0 + x & \to x \\
\mathit{s}(x) + y & \to \mathit{s}(x+y) \\
\text{even}(0) & \to \text{true} \\
\text{even}(\mathit{s}(x)) & \to \text{even}(x)
\end{align*}
\]

For an initial (conditional) expression “if even\( (x) \) then \( x+y \) else \( s(x+y) \)” (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 \( x+y \) 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: 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 \( x \in \mathcal{Var}(t) \), we use the safety abstraction \( m^S \in 2^\mathcal{V} \to 2 \) of \( m = \mathsf{BNarr}^\infty(t) \) (based on \( h_t \) and \( h_y \)).

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

\[
\mathsf{BNarr}^\infty(t) = \{ \begin{align*}
\{ x & \mapsto \bot \} \mapsto \bot, & \{ x \mapsto 0 \} \mapsto \text{true}, \\
\{ x \mapsto \mathit{s}(\bot) \} \mapsto \bot, & \{ x \mapsto \mathit{s}(0) \} \mapsto \text{false}, \\
\{ x \mapsto \mathit{s}(\mathit{s}(\bot)) \} \mapsto \bot, & \{ x \mapsto \mathit{s}(\mathit{s}(0)) \} \mapsto \text{true}, \\
\ldots \}
\end{align*}
\]

In general, if we can prove that, for all abstract substitutions \( \phi^\# \in 2^\mathcal{V} \) 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 \( \langle \epsilon, t \rangle \vdash^* \langle \sigma, \delta \rangle \) such that \( \delta \in \mathcal{T}(C) \) and \( \sigma(x) \not\in \mathcal{T}(C) \), i.e., \( x \) is not grounded. By Proposition 6.8, \( m(\{ x \mapsto \bot \}_{FL}) = \delta \). By definition of \( m^S \), \( m^S(h_y(\{ x \mapsto \bot \}_{FL})) = \top \). Since \( \{ x \mapsto \bot \}_{FL} \not\in \mathcal{T}(C) \), we have \( h_y(\{ x \mapsto \bot \}_{FL})(x) = h_y(\{ x \mapsto \bot \}_{FL}) = \bot \), thus contradicting (a particularization of) our initial assumption, \( m^S(h_y(\{ x \mapsto \bot \}_{FL})) = \bot \).

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

The previous considerations show that the semantic dependency expressed by the \textit{nge}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 fixedpoint 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 $T^{\infty}(C_\perp)$ 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 these 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


