Unification by Narrowing

Mircea Marin

West University of Timisoara
mmarin@info.uvt.ro

May 20, 2014

What is narrowing?

- Narrowing
  - sound and complete method for solving E-unification problems in theories presented by complete term rewriting systems.
  - computational model for functional logic programming (FLP)

What is narrowing?

- Narrowing
  - sound and complete method for solving E-unification problems in theories presented by complete term rewriting systems.
  - computational model for functional logic programming (FLP)
- Functional logic programming = programming style resulted by the integration of two declarative programming styles: Functional Programming and Logic Programming

  FLP = FP + LP

What is narrowing?

- Narrowing
  - sound and complete method for solving E-unification problems in theories presented by complete term rewriting systems.
  - computational model for functional logic programming (FLP)
- Functional logic programming = programming style resulted by the integration of two declarative programming styles: Functional Programming and Logic Programming

$$\text{FLP} = \text{FP} + \text{LP}$$

- Functional Programming
  - Program = term rewriting system (usually terminating and confluent)
  - Computation = reduction to normal form $\Rightarrow$ value

What is narrowing?

- Narrowing
  - sound and complete method for solving E-unification problems in theories presented by complete term rewriting systems.
  - computational model for functional logic programming (FLP)
- Functional logic programming = programming style resulted by the integration of two declarative programming styles: Functional Programming and Logic Programming

  FLP = FP + LP
- Functional Programming
  - Program = term rewriting system (usually terminating and confluent)
  - Computation = reduction to normal form $\Rightarrow$ value
- Logic Programming
  - Program = set of Horn clauses (rules and facts)
  - Computation: SLD resolution of goals $\Rightarrow$ computed answers

DESIRE: inherit the best features from both logic programming and functional programming

- Advantages of logic programming:
  - Logical variables; sound and complete search strategy for answers to queries
- Advantages of functional programming:
  - More efficient operational behaviour: evaluation of function calls is more deterministic than computing answers to queries.

DESIRE: inherit the best features from both logic programming and functional programming

- Advantages of logic programming:
  - Logical variables; sound and complete search strategy for answers to queries
- Advantages of functional programming:
  - More efficient operational behaviour: evaluation of function calls is more deterministic than computing answers to queries.

Approaches to integrate FP with LP and define FLP=FP+LP

1. Integrate functions into LP.
2. Extend FP with equational queries involving function calls and logical variables.

Functional + logic programming
Characteristics

DESIRE: inherit the best features from both logic programming and functional programming

- Advantages of logic programming:
  - Logical variables; sound and complete search strategy for answers to queries
- Advantages of functional programming:
  - More efficient operational behaviour: evaluation of function calls is more deterministic than computing answers to queries.

Approaches to integrate FP with LP and define FLP=FP+LP

1. Integrate functions into LP.
2. Extend FP with equational queries involving function calls and logical variables.

Historically, both approaches resulted in languages with similar computational models.

Starting from

$f, g, h, \ldots \in \mathcal{F}$: ranked signature of function symbols; $ar(f) \in \mathbb{N}$ for all $f \in \mathcal{F}$

$x, y, z, \ldots \in \mathcal{V}$: countable set of variables

we build

- Terms: $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$: $t ::= x \mid f(t_1, \ldots, t_n)$ where $ar(f) = n$
  Convention: abbreviate $f()$ by f
- Equations: $e ::= s = t$ where $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$
- Rewrite rules: $l \to r$ where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $l \notin \mathcal{V}$, $vars(r) \subseteq vars(l)$. A TRS is a set of rewrite rules.
- Rewriting with a TRS $\mathcal{R}$ = replacing "equals by equals" in a directed manner: $s \to_{\mathcal{R}} t$ if there exist $p \in Pos(s)$, $(l \to r) \in \mathcal{R}$, and substitution $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$.

Equational reasoning

Equational reasoning = reasoning with equations in the quotient algebra $\mathcal{T}(\mathcal{F}, \mathcal{V})/_{=_E}$ where $=_E$ is the congruence relation induced on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ by a set of equations E (the equational axioms);

> $=_E$ is the least equivalence relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$, which satisfies the following two additional conditions:
>
> Substitution: if $s =_E t$ then $s\sigma =_E t\sigma$ for all substitutions σ
>
> Replacement: if $l =_E r$ and $s|_p = l$ then $s =_E s[r]_p$.

Equational reasoning

Equational reasoning = reasoning with equations in the quotient algebra $\mathcal{T}(\mathcal{F},\mathcal{V})/_{=_E}$ where $=_E$ is the congruence relation induced on $\mathcal{T}(\mathcal{F},\mathcal{V})$ by a set of equations E (the equational axioms);

$=_E$ is the least equivalence relation on $\mathcal{T}(\mathcal{F},\mathcal{V})$, which satisfies the following two additional conditions:

Substitution: if $s =_E t$ then $s\sigma =_E t\sigma$ for all substitutions σ

Replacement: if $l =_E r$ and $s|_p = l$ then $s =_E s[r]_p$.

E-unification problem

Given a set of equations E and a system of equations

$$\Gamma : s_1 = t_1, \ldots, s_n = t_n$$

Find a representation of the set of substitutions σ such that $s_i\sigma =_E t_i\sigma$ for all $i = 1..n$.

Γ is an E-unification problem, and a σ is a unifier of Γ.

The unification hierarchy (1)

ASSUMPTIONS:

E : set of equations
Γ : E-unification problem
$Sol(\Gamma)$: the set of all unifiers of Γ

- ▷ E induces an order on terms: $s \leq_E t$ if $s\sigma =_E t$ for some σ.
- ▷ A set S of substitutions is a complete set of unifiers (csu) of Γ if
  1. $S \subseteq Sol(\Gamma)$
  2. For any $\theta \in Sol(\Gamma)$ there is a $\sigma \in S$ such that $\sigma(x) \leq_E \theta(x)$ for all $x \in vars(\Gamma)$

  S is a minimal csu (mcsu) of Γ if it also satisfies the following condition:
  - If $\sigma_1, \sigma_2 \in S$ and $\sigma_1(x) \leq_E \sigma_2(x)$ for all $x \in vars(\Gamma)$, then $\sigma_1 = \sigma_2$.

The unification hierarchy (2)

mcsu of Γ may not exist!

Unification problem without mcsu [Schmidt-Schauss, 1986]

$E = \{f(f(x, y), z) = f(x, f(y, z)), f(x, x) = x\}$
$\Gamma : f(z, f(a, f(x, f(a, z)))) = f(z, f(a, z))$

[Siekmann, 1978] introduced the following hierarchy of unification problems:

- unitary: they have a mcsu with 0 or 1 elements.
- finitary: they have a mcsu with finite number of elements.
- infinitary: they have a mcsu with infinite number of elements.
- nullary: they do not have mcsu.

The unification hierarchy (2)

mcsu of Γ may not exist!

Unification problem without mcsu [Schmidt-Schauss, 1986]

$E = \{f(f(x,y),z) = f(x,f(y,z)), f(x,x) = x\}$
$\Gamma : f(z, f(a, f(x, f(a, z)))) = f(z, f(a, z))$

[Siekmann, 1978] introduced the following hierarchy of unification problems:

- unitary: they have a mcsu with 0 or 1 elements.
- finitary: they have a mcsu with finite number of elements.
- infinitary: they have a mcsu with infinite number of elements.
- nullary: they do not have mcsu.

[Nutt, 1991] proved that the unification hierarchy is undecidable.

Unification in theories presented by TRSs

IMPLICIT ASSUMPTIONS: $\mathcal{R}$ is a TRS, and

- $=_{\mathcal{R}}$ is the congruence relation induced by $\mathcal{R}$, viewed as system of equations.
- $s \downarrow_{\mathcal{R}} t :\stackrel{\text{def}}{\Longleftrightarrow}$ there exists u s.t. $s \to_{\mathcal{R}}^{*} u$ and $t \to_{\mathcal{R}}^{*} u$.

From now on we will consider systems of equations (also known as goals)

$$\Gamma : s_1 = t_1, \ldots, s_n = t_n$$

interpreted in equational theories presented by term rewriting systems. This means that:

- We interpret the equality $=$ as $=_{\mathcal{R}}$. If $\mathcal{R}$ is confluent then $=_{\mathcal{R}}$ coincides with $\downarrow_{\mathcal{R}}$.
- We wish to compute a compute set of $\mathcal{R}$-unifiers of Γ. These $\mathcal{R}$-unifiers are also known as solutions of Γ.

A TRS $\mathcal{R}$ is

- terminating (or normalizing) if very sequence of rewrite steps will eventually terminate: $t \rightarrow_{\mathcal{R}} t_1 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t_n \not\rightarrow_{\mathcal{R}}$ t_n is called a normal form of t.
- weakly-normalizing if for any term t there exists a rewrite termination that ends with a normal form: $t = t_0 \rightarrow_{\mathcal{R}} t_1 \rightarrow_{\mathcal{R}} \ldots \rightarrow_{\mathcal{R}} t_n \not\rightarrow_{\mathcal{R}}$
- confluent if $t_1 \downarrow_{\mathcal{R}} t_2$ whenever $t \rightarrow^*_{\mathcal{R}} t_1$ and $t \rightarrow^*_{\mathcal{R}} t_2$.
- semi-complete if it is weakly-normalizing and confluent.
- complete if it is terminating and confluent.

A TRS $\mathcal{R}$ is

- terminating (or normalizing) if very sequence of rewrite steps will eventually terminate: $t \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} t_n \not\to_{\mathcal{R}}$ t_n is called a normal form of t.
- weakly-normalizing if for any term t there exists a rewrite termination that ends with a normal form: $t = t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} t_n \not\to_{\mathcal{R}}$
- confluent if $t_1 \downarrow_{\mathcal{R}} t_2$ whenever $t \to^*_{\mathcal{R}} t_1$ and $t \to^*_{\mathcal{R}} t_2$.
- semi-complete if it is weakly-normalizing and confluent.
- complete if it is terminating and confluent.

Remarks

- If $\mathcal{R}$ is confluent then $s =_{\mathcal{R}} t$ iff $s \downarrow_{\mathcal{R}} t$.
- If $\mathcal{R}$ is complete then $=_{\mathcal{R}}$ is decidable.

Computations in FP and FLP

- Program = complete TRS defined over a signature $\mathcal{F} = \mathcal{F}_d \uplus \mathcal{F}_c$ where
  - $\mathcal{F}_d$: set of defined function symbols
  - $\mathcal{F}_c$: set of constructors
- Rewrite rules are of the form $f(s_1, \ldots, s_n) \to t$ where $f \in \mathcal{F}_d$ and $s_1, \ldots, s_n \in \mathcal{T}(\mathcal{F}_c, \mathcal{V})$.
- Computation in FP: computes the (unique) normal form of a term t
  - Strict languages: terms are reduced by leftmost innermost rewriting.
  - Lazy languages: terms are reduced by leftmost outermost rewriting.
- Computation in FLP: find a csu (preferably mcsu) of

$$\Gamma : s_1 = t_1, \ldots, s_n = t_n$$

ASSUMPTION: $\mathcal{R}$ is a TRS.

Definition (Fresh variant)

A fresh variant of a rewrite rule $l \to r$ is a bijective substitution σ with $dom(\sigma) = vars(l)$ and $\sigma(x)$ is a fresh new variable for each $x \in dom(\sigma)$.

Definition (Narrowing [Slagle, 1974])

s is narrowable into t, notation $s \rightsquigarrow_{\sigma,\mathcal{R}} t$, if there exist

- a narrowing position $p \in Pos(s)$ such that $s|_p \notin \mathcal{V}$
- a fresh variant $l \to r$ of a rewrite rule of $\mathcal{R}$

such that $\sigma = mgu(s|_p, l)$ and $t = s[r]_p\sigma$.

ASSUMPTION: $\mathcal{R}$ is a TRS.

Definition (Fresh variant)

A fresh variant of a rewrite rule $l \to r$ is a bijective substitution σ with $dom(\sigma) = vars(l)$ and $\sigma(x)$ is a fresh new variable for each $x \in dom(\sigma)$.

Definition (Narrowing [Slagle, 1974])

s is narrowable into t, notation $s \rightsquigarrow_{\sigma,\mathcal{R}} t$, if there exist

- a narrowing position $p \in Pos(s)$ such that $s|_p \notin \mathcal{V}$
- a fresh variant $l \to r$ of a rewrite rule of $\mathcal{R}$

such that $\sigma = mgu(s|_p, l)$ and $t = s[r]_p\sigma$.

NOTATION: A derivation $t_0 \rightsquigarrow_{\sigma_1,\mathcal{R}} t_1 \rightsquigarrow_{\sigma_2,\mathcal{R}} \ldots \rightsquigarrow_{\sigma_n,\mathcal{R}} t_n$ is abbreviated $t_0 \rightsquigarrow^*_{\sigma,\mathcal{R}} t_n$, or simply $t_0 \rightsquigarrow^*_{\sigma} t_n$, where $\sigma = \sigma_1 \ldots \sigma_n$.

Theorem ([Hullot, 1985])

If $\mathcal{R}$ is complete then

Soundness: *If $s = t \leadsto_\sigma s' = t'$ and $\theta = mgu(s', t')$ then*
$(s\sigma\theta) =_\mathcal{R} (t\sigma\theta)$

Completeness: *If $s\theta =_\mathcal{R} t\theta$ then there exist*

- $s = t \leadsto^*_\sigma s' = t'$ *and*
- $\sigma' \in mgu(s', t')$

such that $\sigma\sigma' \leq_\mathcal{R} \theta\ [vars(s, t)]$.

Theorem ([Hullot, 1985])

If $\mathcal{R}$ is complete then

Soundness: *If* $s = t \leadsto_\sigma s' = t'$ *and* $\theta = mgu(s', t')$ *then* $(s\sigma\theta) =_\mathcal{R} (t\sigma\theta)$

Completeness: *If* $s\theta =_\mathcal{R} t\theta$ *then there exist*

- $s = t \leadsto_\sigma^* s' = t'$ *and*
- $\sigma' \in mgu(s', t')$

such that $\sigma\sigma' \leq_\mathcal{R} \theta\ [vars(s, t)]$.

Question: Can we drop the condition of termination of $\mathcal{R}$, and still have soundness and completeness?

Answer: Yes, if we restrict ourselves to normalized unifiers: $NSol(\Gamma) = \{\theta \in Sol(\Gamma) \mid x\theta$ is normal form, for all $x \in dom(\theta)\}$.

$\mathcal{R} = \{0 + x \to x, s(x) + y \to s(x + y), x = x \to \texttt{true}\}$.
Let's solve $z + z = s(s(0))$:

$$\underline{z + z} = s(s(0)) \rightsquigarrow_{\color{red}\{y_1 \mapsto s(x_1), z \mapsto s(x_1)\}, s(x_1)+y_1 \to s(x_1+y_1)}$$
$$s(\underline{x_1 + s(x_1)}) = (s(0)) \rightsquigarrow_{\color{red}\{x_1 \mapsto 0, x_2 \mapsto s(0)\}, 0+x_2 \mapsto x_2}$$
$$s(s(0)) = s(s(0)) \rightsquigarrow_{\color{red}\{x_3 \mapsto s(s(0))\}, x_3+x_3 \to \texttt{true}} \texttt{true}$$

Solution: $\{y_1 \mapsto s(x_1), z \mapsto s(x_1)\}\{x_1 \mapsto 0, x_2 \mapsto s(0)\}\{x_3 \mapsto s(s(0))\}$ $= \{y_1 \mapsto s(0), {\color{red}z \mapsto s(0)}, x_1 \mapsto 0, x_2 \mapsto s(0), x_3 \mapsto s(s(0))\}$ restricted to $vars(z + z = 0) = \{z\}$, is ${\color{red}\theta = \{z \mapsto s(0)\}}$
There are also several failed attempts to compute $\mathcal{R}$-unifiers:

$$\underline{z + z} = s(s(0)) \rightsquigarrow_{\{z \mapsto 0, x_1 \mapsto 0\}, 0+x_1 \to x_1} 0 = s(s(0)) \not\rightsquigarrow$$

Let $\mathcal{R}$ be a confluent TRS, $\Gamma : s_1 = t_1, \ldots, s_n = t_n$, and

- $\mathcal{R}_+ := \mathcal{R} \cup \{(x = x) \to \mathtt{true}\}$
- $\top :=$ generic notation for system containing only `true`-s

Definition

$\leadsto_{\mathcal{R}}$ is extended to act on systems of equations as follows:

$$\Gamma_1, e, \Gamma_2 \leadsto_{\sigma,\mathcal{R}} (\Gamma_1, e', \Gamma_2)\sigma$$

if $e \leadsto_{\sigma,\mathcal{R}} e'$ where e is a non-`true` equation.

NOTATION: Like before, we abbreviate $\Gamma_0 \leadsto_{\sigma_1} \ldots \leadsto_{\sigma_n} \Gamma_n$ with $\Gamma_0 \leadsto^*_{\sigma} \Gamma_n$, where $\sigma = \sigma_1 \ldots \sigma_n$. Also, we define the set of answers computed by narrowing: $Ans(\Gamma) = \{\sigma \mid \Gamma \leadsto^*_{\sigma} \top\}$

Corollary

$Ans(\Gamma)$ is a csu of Γ.

The computation of $Ans(\Gamma)$ is highly nondeterministic, due to the selection of

1. the narrowing position
2. the rewrite rule to be applied at the narrowing position

A more deterministic version of narrowing, still sound and complete w.r.t. normalized unifiers, is basic narrowing ([Hullot, 1987], [Middeldorp *et al*, 1996])

Definition (Position constraint)

A position constraint for Γ is a mapping that assigns to every equation $e \in \Gamma$ a subset of $Pos_{\mathcal{F}}(e) = \{p \in Pos(e) \mid e|_p \notin \mathcal{V}\}$. The position constraint that assigns to every $e \in \Gamma$ the set $Pos_{\mathcal{F}}(e)$ is denoted by $\overline{\Gamma}$.

Definition (Basic derivation)

$\Gamma_1 \rightsquigarrow_{\sigma_1,e_1,p_1,l_1\to r_1} \cdots \rightsquigarrow_{\sigma_{n-1},e_{n-1},p_{n-1},l_{n-1}\to r_{n-1}} \Gamma_n$ is based on a position constraint B_1 for Γ_1 if $p_i \in B_i(e_i)$ for $1 \leq i \leq n-1$, where

$$B_{i+1}(e) := \begin{cases} B_i(e') & \text{if } e' \in \Gamma_i \setminus \{e_i\} \\ \mathcal{B}(B_i(e_i), p_i, r_i) & \text{if } e' = e_i[r_i]_{p_i} \end{cases}$$

for all $1 \leq i < n-1$ and $e = e'\sigma_i \in \Gamma_{i+1}$, with $\mathcal{B}(B_i(e_i), p_i, r_i)$ abbreviating the set of positions

$$B_i(e_i) \setminus \{q \in B_i(e_i) \mid q \geq p_i\} \cup \{p_i \cdot q \in Pos_{\mathcal{F}}(e) \mid q \in Pos_{\mathcal{F}}(r_i)\}.$$

Such a narrowing derivation of Γ_1 is basic if $B_1 = \overline{\Gamma}_1$.

REMARK: In a basic narrowing derivation, narrowing is never applied to a subterm introduced by a previous narrowing substitution.

Theorem ([Hullot, 1987], [Middeldorp and Hamoen, 1994])

Let $\mathcal{R}$ be a confluent TRS and Γ a system of equations. For every normalized unifier θ of G there exists a basic narrowing refutation $\Gamma \rightsquigarrow_{\sigma}^{*} \top$ such that $\sigma \leq_{\mathcal{R}} \theta$ [vars(Γ)] provided one of the following conditions is satisfied:

1. *$\mathcal{R}$ is terminating*
2. *$\mathcal{R}$ is orthogonal and $\Gamma\theta$ has an $\mathcal{R}$-normal form*
3. *$\mathcal{R}$ is right-linear*

$\mathcal{R} = \{rev(rev(x)) \to x\}$ specifies a property of the reverse operation on lists.

- An infinite non-basic narrowing derivation

$$\Gamma : \underline{rev(x)} = x \leadsto_{\{x \mapsto rev(x_1)\},1,rev(rev(x_1)) \to x_1} x_1 = \underline{rev(x_1)}$$
$$\leadsto_{\{x_1 \mapsto rev(x_2),2,rev(rev(x_2)) \to x_2\}} \underline{rev(x_2)} = x_2$$
$$\leadsto_{\{x_2 \mapsto rev(x_3),1,rev(rev(x_3)) \to x_3\}} \cdots$$

- The only basic narrowing derivation of the same Γ is

$$\Gamma : \underline{rev(x)} = x \leadsto_{\{x \mapsto rev(x_1)\},1,rev(rev(x_1)) \to x_1} x_1 = rev(x_1)$$

Basic narrowing prohibits any further narrowing steps $\Rightarrow \Gamma$ has no unifiers.

Theorem ([Hullot, 1980])

If $\mathcal{R} = \{l_i \to r_i \mid 1 \leq i \leq n\}$ is a complete TRS, and any basic narrowing derivation starting from r_i terminates, then all basic narrowing derivations starting from any term terminate.

Theorem ([Hullot, 1980])

If $\mathcal{R} = \{l_i \to r_i \mid 1 \leq i \leq n\}$ is a complete TRS, and any basic narrowing derivation starting from r_i terminates, then all basic narrowing derivations starting from any term terminate.

Corollary

Basic narrowing becomes a decision procedure for E-unification if the conditions of the previous theorem hold.

Narrowing calculi

- Computational model of several functional logic programming languages.
- Narrowing is a complicated operation $\Rightarrow$ various narrowing calculi consisting of more elementary inference rules that simulate narrowing have been proposed
- Properties of narrowing calculi
  - Easier to analyse than the narrowing operation
  - Three sources of nondeterminism, due to the choice of
    1. the equation of the system
    2. the inference rule to be applied
    3. the rewrite rule of the TRS (for certain inference rules)
- Several criteria have been proposed to reduce these sources of nondeterminism under reasonable assumptions.

[o] outermost narrowing: $\dfrac{\Gamma_1, f(s_1,\ldots,s_n) \simeq t, \Gamma_2}{\Gamma_1, s_1 = l_1, \ldots, s_n = l_n, r = t, \Gamma_2}$
if $f(l_1,\ldots,l_n) \to r$ is a fresh variant of a rule from $\mathcal{R}$

[i] imitation: $\dfrac{\Gamma_1, f(s_1,\ldots,s_n) \simeq x, \Gamma_2}{(\Gamma_1, s_1 = x_1, \ldots, s_n = x_n, \Gamma_2)\theta}$
if $\theta = \{x \mapsto f(x_1,\ldots,x_n)\}$ with $x_1,\ldots,x_n$ fresh variables.

[d] decomposition: $\dfrac{\Gamma_1, f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n), \Gamma_2}{\Gamma_1, s_1 = t_1, \ldots, s_n = t_n, \Gamma_2}$

[v] variable elimination: $\dfrac{\Gamma_1, x \simeq t, \Gamma_2}{(\Gamma_1, \Gamma_2)\sigma}$
if $x \notin vars(t)$ and $\sigma = \{x \mapsto t\}$

[t] removal of trivial equations: $\dfrac{\Gamma_1, x = x, \Gamma_2}{\Gamma_1, \Gamma_2}$

The red equations produced by [o] are called parameter-passing equations.

NOTATION:

- $\Gamma \Rightarrow_{[\alpha],\sigma} \Gamma'$ if Γ and Γ' are the upper and lower parts of an inference rule $[\alpha]$ ($\alpha \in \{o, i, d, v, t\}$) and σ is the substitution computed by that inference rule.
- $\Box$ denotes the system with no equations.
- An LNC-derivation $\Gamma_0 \Rightarrow_{[\alpha_1],\sigma_1} \ldots \Rightarrow_{[\alpha_n],\sigma_n} \Gamma_n$ is abbreviated $\Gamma_0 \Rightarrow^*_{\sigma} \Box$ where $\sigma = \sigma_1 \ldots \sigma_n$.

Theorem

If $\mathcal{R}$ is confluent and θ is a normalized $\mathcal{R}$-unifier of Γ then there exists $\Gamma \Rightarrow_{\sigma}^{*} \square$ respecting leftmost equation selection strategy such that $\sigma \leq \theta$ [vars(Γ)].

Theorem

If $\mathcal{R}$ is confluent and θ is a normalized $\mathcal{R}$-unifier of Γ then there exists $\Gamma \Rightarrow^*_\sigma \Box$ respecting leftmost equation selection strategy such that $\sigma \leq \theta$ [vars(Γ)].

Theorem

Let $\mathcal{R}$ be a confluent TRS, Γ a system of equations, and $\mathcal{S}$ any selection function for equations from a system. For every normalized solution θ of Γ there exists an LNC-refutation $\Gamma \Rightarrow^*_\sigma \Box$ respecting $\mathcal{S}$ such that $\sigma \leq \theta$ [vars(Γ)] provided one of the following conditions holds:

1. *$\mathcal{R}$ is terminating*
2. *$\mathcal{R}$ is orthogonal and $\Gamma\theta$ has an $\mathcal{R}$-normal form*
3. *$\mathcal{R}$ is right-linear*

Refinements of LNC

LNC with eager variable elimination [Middeldorp and Okui, 1996]

Refinement of LNC which performs eager variable elimination for descendants of parameter-passing equations:

- Whenever we select an equation $x \simeq t$ with $x \notin vars(t)$, which is descendant of a parameter-passing equation, we apply inference rule [v].

Theorem

Let $\mathcal{R}$ be an orthogonal TRS and Γ a system of equations. For every $\mathcal{R}$-normalized unifier θ of Γ there exists an eager LNC-refutation $G \Rightarrow_{\sigma}^{*} \square$ respecting leftmost equation selection strategy, such that $\sigma \leq \theta$ [$vars(\Gamma)$].

Note that a TRS $\mathcal{R}$ is orthogonal if

1. It is left-linear, i.e., no equation appears twice in any lhs of some rewrite rule
2. It's rewrite rules are non-overlapping

Designed for strict solving of systems of equations

Definition

Let $\mathcal{R}$ be a TRS. A substitution σ is a strict solution of a system Γ if for every equation $s = t$ in Γ there exists a constructor term u such that $s\sigma \to^*_{\mathcal{R}} u$ and $t\sigma \to^*_{\mathcal{R}} u$.

LNC_d is a refinement of calculus LNC which distinguishes:

- $\mathcal{F} = \mathcal{F}_c \uplus \mathcal{F}_d$, where $\mathcal{F}_d := \{f \in \mathcal{F} \mid \exists (f(s_1, \ldots, s_n) \to r) \in \mathcal{R}\}$ and $\mathcal{F}_c = \mathcal{F} \setminus \mathcal{F}_d$
- Descendants of initial equations (written as $s \equiv t$) from descendants of parameter-passing equations (written as $s \rhd t$). We write $s \cong t$ if $s \equiv t$ or $t \equiv s$

$[o]_{\equiv}$ outermost narrowing: $\dfrac{f(s_1,\ldots,s_n) \simeq t, \Gamma}{s_1 \rhd l_1,\ldots,s_n \rhd l_n, r \equiv t, \Gamma}$
if $root(t) \notin \mathcal{F}_d$ and $f(s_1,\ldots,s_n) \to r$ is a fresh variant of a rewrite rule from $\mathcal{R}$

$[i]_{\equiv}$ imitation: $\dfrac{f(s_1,\ldots,s_n) \simeq x, \Gamma}{(s_1 \equiv x_1,\ldots,s_n \equiv x_n, \Gamma)\sigma}$
if $f \in \mathcal{F}_c$, $c \notin vars_c(f(s_1,\ldots,s_n))$, $f(s_1,\ldots,s_n) \notin \mathcal{T}(\mathcal{F}_c,\mathcal{V})$, and $\sigma = \{x \mapsto f(x_1,\ldots,x_n)\}$ with $x_1,\ldots,x_n$ fresh variables.

$[d]_{\equiv}$ decomposition: $\dfrac{f(s_1,\ldots,s_n) \equiv f(t_1,\ldots,t_n), \Gamma}{s_1 \equiv t_1,\ldots,s_n \equiv t_n, \Gamma}$

$[v]_{\equiv}$ variable elimination: $\dfrac{s \simeq x, \Gamma}{\Gamma\sigma}$
if $x \notin vars(s)$ and $\sigma = \{x \mapsto s\}$

$[t]_{\equiv}$ removal of trivial equations: $\dfrac{x \equiv x, \Gamma}{\Gamma}$

$[o]_{\rhd}$ outermost narrowing:

$$\frac{f(s_1,\ldots,s_n) \rhd t, \Gamma}{s_1 \rhd l_1,\ldots,s_n \rhd l_n, r \rhd t, \Gamma}$$

if $root(t) \notin \mathcal{F}_d$ and $f(s_1,\ldots,s_n) \to r$ is a fresh variant of a rewrite rule from $\mathcal{R}$

$[d]_{\rhd}$ decomposition: $\dfrac{f(s_1,\ldots,s_n) \rhd f(t_1,\ldots,t_n), \Gamma}{s_1 \rhd t_1,\ldots,s_n \rhd t_n, \Gamma}$

$[v]_{\rhd}$ variable elimination: $\dfrac{s \rhd x, \Gamma}{\Gamma\sigma}$ $\dfrac{x \rhd s, \Gamma}{\Gamma\sigma}$

if $x \notin vars(s)$ and $\sigma = \{x \mapsto s\}$

Theorem

Let $\mathcal{R}$ be a left-linear confluent constructor system and Γ a system of equations. For every strict and normalized solution θ of Γ there exists an LNC_d-refutation $G \Rightarrow_\sigma^* \square$ such that $\sigma \leq_{\mathcal{R}} \theta$ [vars(Γ)]

Theorem

Let $\mathcal{R}$ be a left-linear confluent constructor system and Γ a system of equations. For every strict and normalized solution θ of Γ there exists an LNC_d-refutation $G \Rightarrow_\sigma^* \square$ such that $\sigma \leq_\mathcal{R} \theta\ [vars(\Gamma)]$

Remark

The only source of nondeterminism of the lazy narrowing calculus LNC_d is the choice of the rewrite rule when applying inference rules $[o]_\equiv$ and $[o]_\triangleright$.

The other sources of nondeterminism disappeared:

1. The selected equation is always the leftmost
2. There is only at most one applicable inference rule.

Extensions to larger classes of TRSs

A conditional TRS (CTRS) consists of conditional rewrite rules $l \to r \Leftarrow c$ where the conditional part is a (possibly empty) sequence $s_1 = t_1, \ldots, s_n = t_n$ of equations. We require $l \notin \mathcal{V}$.

$$evars(l \to r \Leftarrow c) := vars(r, c) \setminus vars(l)$$

Extensions to larger classes of TRSs

A conditional TRS (CTRS) consists of conditional rewrite rules $l \to r \Leftarrow c$ where the conditional part is a (possibly empty) sequence $s_1 = t_1, \ldots, s_n = t_n$ of equations. We require $l \notin \mathcal{V}$.

$$evars(l \to r \Leftarrow c) := vars(r, c) \setminus vars(l)$$

CTRSs are classified according to the distribution of variables in rewrite rules, into:

1-CTRS	$vars(r, c) \subseteq vars(l)$	for all rules $l \to r \Leftarrow c$
2-CTRS	$vars(r) \subseteq vars(l)$	for all rules $l \to r \Leftarrow c$
3-CTRS	$vars(r) \subseteq vars(l, c)$	for all rules $l \to r \Leftarrow c$

REMARK: Extra variables enable a note natural style of writing program specifications

Example (Fibonacci numbers)

$0 + y \to y,\ s(x) + y \to s(x + y),$
$fib(0) \to \langle 0, s(0) \rangle,$
$fib(s(x)) \to \langle z, y + z \rangle \Leftarrow fib(x) = \langle y, z \rangle$

Rewriting with conditional TRSs

ASSUMPTIONS:

- every CTRS $\mathcal{R}$ contains the rewrite rule $x = x \to \mathtt{true}$
- $\mathtt{true}$ and $=$ do not occur in other rewrite rules of $\mathcal{R}$
- $\top$ denotes any sequence of $\mathtt{true}$s

We define inductively the unconditional TRSs $\mathcal{R}_n$ for $n \geq 0$:

$$\mathcal{R}_0 := \{x = x \to \mathtt{true}\}$$
$$\mathcal{R}_{n+1} := \{l\sigma \to r\sigma \mid l \to r \Leftarrow c \in \mathcal{R} \text{ and } c\sigma \to^*_{\mathcal{R}_n} \top\}$$

and abbreviate $\to_{\mathcal{R}_n}$ by $\to_n$

Rewriting with conditional TRSs

ASSUMPTIONS:

- every CTRS $\mathcal{R}$ contains the rewrite rule $x = x \to \mathtt{true}$
- $\mathtt{true}$ and $=$ do not occur in other rewrite rules of $\mathcal{R}$
- $\top$ denotes any sequence of $\mathtt{true}$s

We define inductively the unconditional TRSs $\mathcal{R}_n$ for $n \geq 0$:

$$\mathcal{R}_0 := \{x = x \to \mathtt{true}\}$$
$$\mathcal{R}_{n+1} := \{l\sigma \to r\sigma \mid l \to r \Leftarrow c \in \mathcal{R} \text{ and } c\sigma \to^*_{\mathcal{R}_n} \top\}$$

and abbreviate $\to_{\mathcal{R}_n}$ by $\to_n$

Remarks

We interpret equality as joinability; such kind of CTRSs are known as join CTRSs in the literature.

Level confluence: $\mathcal{R}$ is level-confluent if every $\mathcal{R}_n$ is confluent.

Shallow-confluence: $\mathcal{R}$ is shallow-confluent if

$$ {}_m^{*}\!\leftarrow \circ \rightarrow_n^{*} \subseteq {}_n^{*}\!\leftarrow \circ \rightarrow_m^{*} \text{ for all } m, n \geq 0. $$

Decreasingness: $\mathcal{R}$ is decreasing if there exists a well-founded $\succ$ order on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ with the following properties:

- $\succ$ contains $\rightarrow_{\mathcal{R}}$
- $\succ$ has the subterm property (i.e., $\lhd \subseteq \succ$ where $s \rhd t$ iff t is a proper subterm of s)
- $t\sigma \succ s\sigma$ and $s\sigma \succ t\sigma$ for every $l \rightarrow r \Leftarrow c \in \mathcal{R}$, every $s = t$ from c, and every substitution σ.

Remark

Shallow-confluent CTRSs are level-confluent, but the reverse is not true.

ASSUMPTION: $\mathcal{R}$ is a CTRS

$$\frac{\Gamma', e, \Gamma''}{(\Gamma', e[r]_p, c, \Gamma'')\sigma}$$

if there exist a fresh variant $l \to r \Leftarrow c$ of a rewrite rule in $\mathcal{R}$, a non-variable position p in e, and $\sigma = mgu(e|_p, l)$.

Remarks

- The previous inference rule is also written as $(\Gamma', e, \Gamma'') \rightsquigarrow_{\sigma,p,l\to r\Leftarrow c} (\Gamma', e[r]_p, c, \Gamma'')\sigma$ or simply $(\Gamma', e, \Gamma'') \rightsquigarrow_{\sigma} (\Gamma', e[r]_p, c, \Gamma'')\sigma$.
- CNC is sound: If $\Gamma \rightsquigarrow^*_\sigma \top$ then $\sigma|_{vars(G)}$ is an $\mathcal{R}$-unifier of Γ.
- We can define basic conditional narrowing, similar to basic narrowing:
  - Main idea: no narrowing steps should take place at positions introduced by previous narrowing substitutions.

Basic conditional narrowing (1)

A CNC-derivation

$$\Gamma_1 \rightsquigarrow_{\theta_1, p_1, l_1 \to r_1 \Leftarrow c_1} \cdots \rightsquigarrow_{\theta_{n-1}, p_{n-1}, l_{n-1} \to r_{n-1} \Leftarrow c_{n-1}} \Gamma_n$$

is based on a position constraint B_1 for Γ_1 if $p_i \in B_i(e_i)$ for $1 \leq i \leq n-1$, where the position constraints $B_2, \ldots, B_{n-1}$ for $\Gamma_2, \ldots, \Gamma_{n-1}$ are defined inductively by

$$B_{i+1}(e) = \begin{cases} B_i(e') & \text{if } e' \in \Gamma_i \setminus \{e_i\} \\ \mathcal{B}(B_i(e_i), p_i, r_i) & \text{if } e' = e_i[r_i]_{p_i} \\ Pos_{\mathcal{F}}(e') & \text{if } e' \in c_i \end{cases}$$

for all $1 \leq i < n-1$ and $e = e'\theta_i \in \Gamma_{i+1}$, with $\mathcal{B}(B_i(e_i), p_i, r_i)$ abbreviating the set of positions

$$(B_i(e_i) \setminus \{q \in B_i(e_i) \mid q \geq p_i\}) \cup \{p_i \cdot q \in Pos_{\mathcal{F}}(e) \mid q \in Pos_{\mathcal{F}}(r_i)\}$$

Basic conditional narrowing (2)

- ▷ The position constraint on Γ that assigns the set of positions $Pos_{\mathcal{F}}(e)$ to every e in G is denoted by $\overline{G}$
- ▷ A CNC derivation is basic it it is based on $\overline{G}$
  - Basic CNC has much smaller search space than CNC

CNC is complete for

- semi-complete 1-CTRSs
- semi-confluent 1-CTRSs w.r.t. normalizable substitutions
- level-semi-complete 2-CTRSs
- level-complete 3-CTRSs

Basic conditional narrowing is complete for

- decreasing and confluent 1-CTRSs
- semi-complete orthogonal 1-CTRSs

REFERENCE: [Middeldorp and Hamoen, 1994]

Extending LNC to work with 3-CTRS

LCNC = lazy conditional narrowing calculus
The only change is inference rule [o]:

[o] outermost narrowing $\dfrac{\Gamma', f(s_1, \ldots, s_n) \simeq t, \Gamma''}{\Gamma', s_1 = l_1, s_n = l_n, r = t, c, \Gamma''}$
if $l \to r \Leftarrow c$ is a fresh variant of a rewrite rule in $\mathcal{R}$.

The other inference rules ([i], [d], [v], [t]) are like those of LNC.

$$\mathcal{R} = \{\ 0 + y = y, s(x) + y \to s(x+y), fib(0) = \langle 0, s(0)\rangle, \\ fib(s(x)) \to \langle z, y+z\rangle \Leftarrow fib(x) = \langle y, z\rangle \}$$

$$\begin{array}{rl}
\underline{fib(x) = \langle x, x\rangle} \Rightarrow_{[o]} & x = s(x_1), \underline{\langle z_1, y_1 + z_1\rangle = \langle x, x\rangle}, fib(x_1) = \langle y_1, z_1\rangle \\
\Rightarrow_{[d]} & x = s(x_1), \underline{z_1 = x}, y_1 + z_1 = x, fib(x_1) = \langle y_1, z_1\rangle \\
\Rightarrow_{[v],\{z_1 \mapsto x\}} & x = s(x_1), y_1 + x = x, \underline{fib(x_1) = \langle y_1, x\rangle} \\
\Rightarrow_{[o]} & x = s(x_1), y_1 + x = x, \underline{x_1 = 0}, \langle 0, s(0)\rangle = \langle y_1, x\rangle \\
\Rightarrow_{[v],\{x_1 \mapsto 0\}} & x = s(0), y_1 + x = x, \underline{\langle 0, s(0)\rangle = \langle y_1, x\rangle} \\
\Rightarrow_{[d]} & x = s(0), y_1 + x = x, \underline{0 = y_1}, s(0) = x \\
\Rightarrow_{[v],\{y_1 \mapsto 0\}} & \underline{x = s(0)}, 0 + x = x, s(0) = x \\
\Rightarrow_{[v],\{x \mapsto s(0)\}} & 0 + s(0) = s(0), \underline{s(0) = s(0)} \\
\Rightarrow_{[d]} & 0 + s(0) = s(0), \underline{0 = 0} \Rightarrow_{[d]} \underline{0 + s(0) = s(0)} \\
\Rightarrow_{[o]} & 0 = 0, s(0) = y_2, \underline{y_2 = s(0)} \\
\Rightarrow_{[v],\{y_1 \mapsto s(0)\}} & 0 = 0, \underline{s(0) = s(0)} \\
\Rightarrow_{[d]} & 0 = 0, 0 = 0 \\
\Rightarrow_{[d]} \Rightarrow_{[d]} & \square
\end{array}$$

LCNC

Example

$$\mathcal{R} = \{\ 0 + y = y, s(x) + y \to s(x+y), fib(0) = \langle 0, s(0)\rangle,$$
$$fib(s(x)) \to \langle z, y+z\rangle \Leftarrow fib(x) = \langle y, z\rangle \}$$

$$\begin{aligned}
\underline{fib(x) = \langle x, x\rangle} &\Rightarrow_{[o]} && x = s(x_1), \underline{\langle z_1, y_1 + z_1\rangle = \langle x, x\rangle}, fib(x_1) = \langle y_1, z_1\rangle \\
&\Rightarrow_{[d]} && x = s(x_1), \underline{z_1 = x}, y_1 + z_1 = x, fib(x_1) = \langle y_1, z_1\rangle \\
&\Rightarrow_{[v],\{z_1 \mapsto x\}} && x = s(x_1), y_1 + x = x, \underline{fib(x_1) = \langle y_1, x\rangle} \\
&\Rightarrow_{[o]} && x = s(x_1), y_1 + x = x, \underline{x_1 = 0}, \langle 0, s(0)\rangle = \langle y_1, x\rangle \\
&\Rightarrow_{[v],\{x_1 \mapsto 0\}} && x = s(0), y_1 + x = x, \underline{\langle 0, s(0)\rangle = \langle y_1, x\rangle} \\
&\Rightarrow_{[d]} && x = s(0), y_1 + x = x, \underline{0 = y_1}, s(0) = x \\
&\Rightarrow_{[v],\{y_1 \mapsto 0\}} && \underline{x = s(0)}, 0 + x = x, s(0) = x \\
&\Rightarrow_{[v],\{x \mapsto s(0)\}} && 0 + s(0) = s(0), \underline{s(0) = s(0)} \\
&\Rightarrow_{[d]} && 0 + s(0) = s(0), \underline{0 = 0} \Rightarrow_{[d]} \underline{0 + s(0) = s(0)} \\
&\Rightarrow_{[o]} && 0 = 0, s(0) = y_2, \underline{y_2 = s(0)} \\
&\Rightarrow_{[v],\{y_1 \mapsto s(0)\}} && 0 = 0, \underline{s(0) = s(0)} \\
&\Rightarrow_{[d]} && 0 = 0, 0 = 0 \\
&\Rightarrow_{[d]}\Rightarrow_{[d]} && \square
\end{aligned}$$

Computed substitution: $\{x \mapsto s(0)\}$

Properties of LCNC

Theorem

Let $\mathcal{R}$ be a confluent 1-CTRS and Γ a system of equations. For every normalized unifier θ of Γ there exists an LCNC-refutation $\Gamma \Rightarrow^*_\sigma \Box$ respecting leftmost equation selection strategy such that $\sigma \leq \theta$ [vars(Γ)]

Theorem

Let $\mathcal{R}$ be an arbitrary CTRS and $\Gamma \rightsquigarrow^*_\theta \top$ be a basic CNC-refutation. For every selection function $\mathcal{S}$ there exists an LCNC-refutation respecting $\mathcal{S}$ such that $\sigma \leq \theta$ [vars(Γ)].

Theorem

Let $\mathcal{R}$ be a terminating and level-confluent CTRS. For every $\mathcal{R}$-unifier θ of a system Γ there exists an LCNC-refutation $\Gamma \Rightarrow^*_\sigma \Box$ such that $\sigma \leq \theta$ [vars(Γ)].

Unification for deterministic CTRSs (1)

We will consider equations of two kinds: $s = t$ and $s \rhd t$

Definition

Let X be a set of variables. A system of equations $\Gamma : e_1, \ldots, e_n$ is *X-deterministic* if

- $vars(s_i) \subseteq X \cup \bigcup_{j=1}^{i-1} vars(e_j)$ when e_i is $s_i \rhd t_i$
- $vars(e_i) \subseteq X \cup \bigcup_{j=1}^{i-1} vars(e_j)$ when e_i is $s_i = t_i$

A CTRS $\mathcal{R}$ is *deterministic* if it is made of rewrite rules of the form $l \to r \Leftarrow c$ where c is an $vars(l)$-deterministic system of equations. $\mathcal{R}$ is *fresh* if $vars(t) \cap vars(l) = \emptyset$ for every $s \rhd t$ in the condition c of any rewrite rule $l \to r \Leftarrow c$ from $\mathcal{R}$.

When rewriting with deterministic CTRSs, = is interpreted as joinability ($\downarrow_{\mathcal{R}}$), and $\rhd$ as reducibility ($\to_R$):

- If $\mathcal{R}$ is deterministic then $s \to_{\mathcal{R}} t$ if there exist $l \to r \Leftarrow c \in \mathcal{R}$, $p \in Pos_{\mathcal{F}}(s)$ and θ such that $s|_p = l\theta$, $t = s[r\theta]_p$, and for all $e_i \in c$: $s_i\theta \downarrow_{\mathcal{R}} t_i\theta$ if e_i is $s_i = t_i$; $s_i\theta \to^*_{\mathcal{R}} t_i\theta$ if e_i is $s_i \rhd t_i$.

$$\mathcal{R} = \{ \quad \begin{array}{ll} 0 + y \to y, & fst(\langle x, y \rangle) \to x, \\ s(x) + y \to s(x + y), & snd(\langle x, y \rangle) \to y, \\ fib(0) \to \langle 0, s(0) \rangle, & \\ fib(s(x)) \to \langle y, y + z \rangle \Leftarrow fib(x) \rhd \langle y, z \rangle & \end{array}$$

The goal

$$\Gamma : fib(s(x)) \rhd \langle s(s(s(0))), y \rangle, y \rhd s(z)$$

is $\{x\}$-deterministic.

Unification for deterministic CTRSs (2)

The calculus $\mathrm{LCNC}_\ell^\dagger$ [Marin and Middeldorp, 2004]

Given an X-deterministic goal Γ and a deterministic CTRS $\mathcal{R}$

Compute a complete set of X-normalized $\mathcal{R}$-unifiers of Γ. A substitution θ is X-normalized if $\theta(x)$ is an $\mathcal{R}$-normal form for all $x \in X$.

$\mathrm{LCNC}_\ell^\dagger$: refinement of LCNC adjusted to resolve this problem

- Works on terms from $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\dagger, \mathcal{V} \cup \mathcal{V}^\dagger)$ where $\mathcal{F}^\dagger$ (resp. $\mathcal{V}^\dagger$) is the set of marked function symbols (resp. marked variables). The purpose of marking is to avoid computing many non-normalised solutions.
- We write $t^\dagger$ for the term obtained from t by marking its root symbol, if not already marked.
- We write $\mathtt{u}(t)$ for the term obtained by removing all markers from t. E.g., $\mathtt{u}(f^\dagger(x, g(y^\dagger))) = f(x, g(y))$

Same as LCNC, except that the leftmost equation is always selected, and the inference rules $[i], [d], [v], [t]$ are adjusted as follows:

$$[i]\quad \frac{f(s_1,\ldots,s_n) \rhd x, \Gamma}{(s_1 \rhd x_1,\ldots,s_n \rhd x_n,\Gamma)\sigma} \qquad \frac{f(s_1,\ldots,s_n) \simeq x^\dagger, \Gamma}{(s_1 \rhd x_1^\dagger,\ldots,s_n \rhd x_n^\dagger,\Gamma)\sigma'}$$

with $(s \simeq t) \in \{s = t, t = s, s \rhd t\}$, $\sigma = \{x \mapsto f(x_1,\ldots,x_n), x^\dagger \mapsto f^\dagger(x_1,\ldots,x_n)\}$, $\sigma' = \{x, x^\dagger \mapsto f^\dagger(x_1,\ldots,x_n)\}$, and $x_1,\ldots,x_n$ fresh variables

$$[d]\quad \frac{f(s_1,\ldots,s_n) \simeq f(t_1,\ldots,t_n), \Gamma}{s_1 \simeq t_1,\ldots,s_n \simeq t_n, \Gamma} \qquad \frac{f^\dagger(s_1,\ldots,s_n) \simeq f(t_1,\ldots,t_n), \Gamma}{s_1^\dagger \simeq t_1,\ldots,s_n^\dagger \simeq t_n, \Gamma}$$

$$\frac{f(s_1,\ldots,s_n) \simeq f^\dagger(t_1,\ldots,t_n), \Gamma}{s_1 \simeq t_1^\dagger,\ldots,s_n \simeq t_n^\dagger, \Gamma} \qquad \frac{f^\dagger(s_1,\ldots,s_n) \simeq f^\dagger(t_1,\ldots,t_n), \Gamma}{s_1^\dagger \simeq t_1^\dagger,\ldots,s_n^\dagger \simeq t_n^\dagger, \Gamma}$$

with $\simeq \in \{=, \rhd\}$

$$[v]\quad \frac{x^\dagger \simeq s, \Gamma}{\Gamma\theta'} \text{ if } s \notin \mathcal{V} \cup \mathcal{V}^\dagger \qquad \frac{s \simeq x^\dagger, \Gamma}{\Gamma\theta'} \qquad \frac{s \rhd x, \Gamma}{\Gamma\theta}$$

with $x \notin vars(\mathrm{u}(s))$, $\simeq \in \{=, \rhd\}$,
$\theta = \{x \mapsto s, x^\dagger \mapsto s^\dagger\}$, and
$\theta' = \{x, x^\dagger \mapsto s^\dagger\} \cup \{y \mapsto y^\dagger \mid y \in vars(\mathrm{u}(s))\}$

$$[t]\quad \frac{s \simeq t, \Gamma}{\Gamma} \text{ if } \mathrm{u}(s) = \mathrm{u}(t) \text{ and } \simeq \in \{=, \rhd\}$$

NOTATION: $\Gamma^\dagger$ is the result of replacing all variables x with $x^\dagger$ in Γ

Theorem

Let $\mathcal{R}$ be a deterministic CTRS and θ a normalized $\mathcal{R}$-unifier of Γ. There exists an $\mathrm{LCNC}_\ell^\dagger$-refutation $G^\dagger \Rightarrow_\sigma^* \square$ such that $\mathrm{u}(\sigma) \leq \theta\ [vars(\Gamma)]$

The calculus $\mathrm{LCNC}_\ell^\dagger$

Nondeterminism due to selection of inference rules

- For unoriented equation $s = t$

$\mathrm{root}(s) \backslash \mathrm{root}(t)$	$\mathcal{F}^\dagger$	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$	$\mathcal{V}^\dagger$	$\mathcal{V}$
$\mathcal{F}^\dagger$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_2$	$[\mathtt{v}]$	$\times$
$\mathcal{F}_\mathcal{C}$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{o}]_2$	$[\mathtt{i}],[\mathtt{v}]$	$\times$
$\mathcal{F}_\mathcal{D}$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_1$	$[\mathtt{o}]_1$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_1,[\mathtt{o}]_2$	$[\mathtt{o}]_1,[\mathtt{i}],[\mathtt{v}]$	$\times$
$\mathcal{V}^\dagger$	$[\mathtt{v}]$	$[\mathtt{i}],[\mathtt{v}]$	$[\mathtt{o}]_2,[\mathtt{i}],[\mathtt{v}]$	$[\mathtt{t}];[\mathtt{v}]$	$\times$
$\mathcal{V}$	$\times$	$\times$	$\times$	$\times$	$\times$

- For oriented equation $s \rhd t$

$\mathrm{root}(s) \backslash \mathrm{root}(t)$	$\mathcal{F}^\dagger$	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$	$\mathcal{V} \cup \mathcal{V}^\dagger$
$\mathcal{F}^\dagger$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{v}]$
$\mathcal{F}_\mathcal{C}$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$\times$	$[\mathtt{i}],[\mathtt{v}]$
$\mathcal{F}_\mathcal{D}$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_1$	$[\mathtt{o}]_1$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_1$	$[\mathtt{o}]_1,[\mathtt{i}],[\mathtt{v}]$
$\mathcal{V}^\dagger$	$[\mathtt{v}]$	$[\mathtt{v}]$	$[\mathtt{v}]$	$[\mathtt{t}];[\mathtt{v}]$
$\mathcal{V}$	$\times$	$\times$	$\times$	$\times$

MAIN IDEA: Like LNC, the calculus LCNC can apply eagerly variable elimination for descendants of parameter-passing equations without losing completeness

- Descendants of parameter-passing equations are defined in exactly the same way as for LNC; we write $s \blacktriangleright t$ to distinguish them from other kinds of equations.
- $\mathrm{LCNC}_\ell^{\mathsf{eve}}$: adjustment of $\mathrm{LCNC}_\ell^{\dagger}$ with the following strategy to solve parameter-passing equations:

root(s) \ root(t)	$\mathcal{F_C}$	$\mathcal{F_D}$	$\mathcal{V}$	$\mathcal{F}^\dagger \cup \mathcal{V}^\dagger$
$\mathcal{F}^\dagger$	[t]; [d]	[t]; [d]	[v]	×
$\mathcal{F_C}$	[t]; [d]	×	[v]	×
$\mathcal{F_D}$	$[\mathrm{o}]_1$	[t]; [d], $[\mathrm{o}]_1$	[v]	×
$\mathcal{V}^\dagger$	[v]	[v]	[v]	×
$\mathcal{V}$	×	×	×	×

BAD NEWS: $\mathrm{LCNC}_\ell^{\mathsf{eve}}$ is incomplete for left-linear deterministic CTRSs.

Example

- $\mathcal{R} = \{f(x) \to x, g(x,y) \to x \Leftarrow x \rhd y\}$
- $\Gamma : g(x, f(y)) = a$

$\mathcal{R}$ is left-linear and deterministic, but not fresh; $\theta = \{x \mapsto a, y \mapsto a\}$ is a normalized solution of Γ. θ can not be computed with $\mathrm{LCNC}_\ell^{\mathsf{eve}}$, because the only maximal $\mathrm{LCNC}_\ell^{\mathsf{eve}}$-derivation is

$$\begin{aligned}
\Gamma^\dagger \Rightarrow_{[o]} \quad & x^\dagger \blacktriangleright x_1, f(y^\dagger) \blacktriangleright y_1, x_1 \rhd y_1, x_1 = a \\
\Rightarrow_{[v],\{x_1 \mapsto x^\dagger, x_1^\dagger \mapsto x^\dagger\}} \quad & f(y^\dagger) \blacktriangleright y_1, x^\dagger \rhd y_1, x^\dagger = a \\
\Rightarrow_{[v],\{y_1 \mapsto f(y^\dagger), y_1^\dagger \mapsto f^\dagger(y^\dagger)\}} \quad & x^\dagger \rhd f(y^\dagger), x^\dagger = a \\
\Rightarrow_{[v],\{x \mapsto f^\dagger(y^\dagger), x^\dagger \mapsto f^\dagger(y^\dagger)\}} \quad & f^\dagger(y^\dagger) = a
\end{aligned}$$

BAD NEWS: $\mathrm{LCNC}_\ell^{\mathsf{eve}}$ is incomplete for left-linear deterministic CTRSs.

Example

- $\mathcal{R} = \{f(x) \to x, g(x,y) \to x \Leftarrow x \rhd y\}$
- $\Gamma : g(x, f(y)) = a$

$\mathcal{R}$ is left-linear and deterministic, but not fresh; $\theta = \{x \mapsto a, y \mapsto a\}$ is a normalized solution of Γ. θ can not be computed with $\mathrm{LCNC}_\ell^{\mathsf{eve}}$, because the only maximal $\mathrm{LCNC}_\ell^{\mathsf{eve}}$-derivation is

$$\begin{aligned}
\Gamma^\dagger &\Rightarrow_{[o]} && x^\dagger \blacktriangleright x_1, f(y^\dagger) \blacktriangleright y_1, x_1 \rhd y_1, x_1 = a \\
&\Rightarrow_{[v],\{x_1 \mapsto x^\dagger, x_1^\dagger \mapsto x^\dagger\}} && f(y^\dagger) \blacktriangleright y_1, x^\dagger \rhd y_1, x^\dagger = a \\
&\Rightarrow_{[v],\{y_1 \mapsto f(y^\dagger), y_1^\dagger \mapsto f^\dagger(y^\dagger)\}} && x^\dagger \rhd f(y^\dagger), x^\dagger = a \\
&\Rightarrow_{[v],\{x \mapsto f^\dagger(y^\dagger), x^\dagger \mapsto f^\dagger(y^\dagger)\}} && f^\dagger(y^\dagger) = a
\end{aligned}$$

GOOD NEWS: $\mathrm{LCNC}_\ell^{\mathsf{eve}}$ is complete for left-linear fresh deterministic CTRSs

Unification of strict equations for deterministic CTRSs

Definition

A substitution θ is a strict solution of an equation

- $s \rhd t$ if $s\theta \to_{\mathcal{R}}^{*} t\theta$ and $t\theta \in \mathcal{T}(\mathcal{F}_c, \mathcal{V})$ (that is, $t\theta$ is a term without defined function symbols)
- $s = t$ if there exists a term $u \in \mathcal{T}(\mathcal{F}_c, \mathcal{V})$ such that $s\theta \to_{\mathcal{R}}^{*} u$ and $t\theta \to_{\mathcal{R}}^{*} u$.

θ is a strict solution of a goal Γ if it is a strict solution of every equation from Γ.

CHALLENGE: Find a refinement of LCNC which computes a complete set of strict solutions of a given goal Γ.

Unification of strict equations for deterministic CTRSs

Definition

A substitution θ is a strict solution of an equation

- $s \rhd t$ if $s\theta \to^*_{\mathcal{R}} t\theta$ and $t\theta \in \mathcal{T}(\mathcal{F}_c, \mathcal{V})$ (that is, $t\theta$ is a term without defined function symbols)
- $s = t$ if there exists a term $u \in \mathcal{T}(\mathcal{F}_c, \mathcal{V})$ such that $s\theta \to^*_{\mathcal{R}} u$ and $t\theta \to^*_{\mathcal{R}} u$.

θ is a strict solution of a goal Γ if it is a strict solution of every equation from Γ.

CHALLENGE: Find a refinement of LCNC which computes a complete set of strict solutions of a given goal Γ.

- The calculus LCNC^s_ℓ [Marin and Middeldorp, 2004] was designed for this purpose.

[o] $\dfrac{f(s_1,\ldots,s_n) \simeq t, \Gamma}{s_1 \blacktriangleright l_1, \ldots, s_n \blacktriangleright l_n, c, \Gamma}$ where $\simeq \in \{=, =^{-1}, \rhd, \blacktriangleright\}$
if $f(l_1,\ldots,l_n) \to r \Leftarrow c$ is a fresh variant of a rewrite rule in $\mathcal{R}$

[i] $\dfrac{g(s_1,\ldots,s_n) \simeq x, \Gamma}{(s_1 \simeq x_1, \ldots, s_n \simeq x_n, \Gamma)\theta}$ where $\simeq \in \{=, =^{-1}, \rhd\}$, $g \in \mathcal{F}_c$
if $g(s_1,\ldots,s_n) \notin \mathcal{T}(\mathcal{F}_c, \mathcal{V})$, $\theta = \{x \mapsto g(x_1,\ldots,x_n)\}$.

$\dfrac{g(s_1,\ldots,s_n) \blacktriangleright x, \Gamma}{(s_1 \blacktriangleright x_1, \ldots, s_n \blacktriangleright x_n, \Gamma)\theta'}$
if $\theta' = \{x \mapsto f(x_1,\ldots,x_n)\}$

[d] $\dfrac{g(s_1,\ldots,s_n) \simeq g(t_1,\ldots,t_n), \Gamma}{s_1 \simeq t_1, s_n \simeq t_n, \Gamma}$ where $\simeq \in \{=, \rhd\}$ and $g \in \mathcal{F}_c$

$\dfrac{f(s_1,\ldots,s_n) \simeq f(t_1,\ldots,t_n), \Gamma}{s_1 \rhd t_1, \ldots, s_n \rhd t_n, \Gamma}$

[v] $\dfrac{x \blacktriangleright s, \Gamma}{\Gamma\theta}$ were $s \notin \mathcal{V}$ $\qquad \dfrac{s \simeq x, \Gamma}{\Gamma\theta}$ where $s \in \mathcal{T}(\mathcal{F}_c, \mathcal{V})$

$\dfrac{s \blacktriangleright x, \Gamma}{\Gamma\theta} \qquad \dfrac{x \simeq s, \Gamma}{\Gamma\theta}$ where $s \in \mathcal{T}(\mathcal{F}_c, \mathcal{V}) \setminus \mathcal{V}$

where $x \notin vars(s)$, $\simeq \in \{=, \rhd\}$, and $\theta = \{x \mapsto s\}$

[t] $\dfrac{s \rhd s, \Gamma}{\Gamma} \qquad \dfrac{s \simeq s, \Gamma}{\Gamma}$ where $\simeq \in \{=, \rhd\}$ and $s \in \mathcal{T}(\mathcal{F}_c, \mathcal{V})$

$s \approx t$

	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$	$\mathcal{V}$
$\mathcal{F}_\mathcal{C}$	[t]; [d]	$[\mathtt{o}]_2$	[v]; [i]
$\mathcal{F}_\mathcal{D}$	$[\mathtt{o}]_1$	$[\mathtt{o}]_1$	$[\mathtt{o}]_1$
$\mathcal{V}$	[v]; [i]	$[\mathtt{o}]_2$	[t]; [v]

$s \rhd t$

	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$	$\mathcal{V}$
$\mathcal{F}_\mathcal{C}$	[t]; [d]	$\times$	[v]; [i]
$\mathcal{F}_\mathcal{D}$	$[\mathtt{o}]_1$	[t]; $[\mathtt{o}]_1$	$[\mathtt{o}]_1$
$\mathcal{V}$	[v]	$\times$	[t]; [v]

$s \blacktriangleright t$

	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$	$\mathcal{V}$
$\mathcal{F}_\mathcal{C}$	[t]; [d]	$\times$	[v]; [i]
$\mathcal{F}_\mathcal{D}$	$[\mathtt{o}]_1$	[t]; ($[\mathtt{o}]_1$, [d])	$[\mathtt{o}]_1$, [i], [v]
$\mathcal{V}$	[v]	[v]	[t]; [v]

Theorem

Let $\mathcal{R}$ be a deterministic CTRS and θ an $\mathcal{R}$-normalized strict solution of Γ. Then there exists an LCNC_ℓ^s-refutation $\Gamma \Rightarrow_\sigma^* \Box$ such that $\sigma \leq \theta$ [vars(Γ)].

Higher-order extensions of the narrowing calculus

- Functional programming operates with functions as values
- The lambda calculus is suitable to express functional computations (function abstractions and function calls)
  $\Rightarrow$ it is natural to try to extend narrowing to solve systems of equations between λ-terms

$$\begin{array}{lll} t ::= & x & \text{variable} \\ & \lambda x.t & \text{abstraction} \\ & (t\ t) & \text{application} \end{array}$$

where x ranges over a countably infinite set of variables.

Conversion rules for λ-terms

λ-terms are identified modulo the following conversion rules:

- $\lambda x.t \to \lambda y.(t\{x \mapsto y\})$ if $y \in \mathcal{V} \setminus vars(t)$ (α-conversion)
- $(\lambda x.s)\ t = s\{x \mapsto t\}$ (β-conversion)
- $(\lambda x.(t\ x) = t$ if $x \in \mathcal{V} \setminus vars(t)$ (η-conversion)

Rewriting systems for λ-terms

- Usually, higher-order *E*-unification is performed between simply-typed λ-terms, which can be represented in a standard form called long $\beta\eta$-normal form
- TRSs have been generalised to pattern rewrite systems (PRS), and CTRSs to conditional PRSs
- Narrowing has been generalised to higher-order lazy narrowing with PRSs.
  - 1998: Prehofer proposed lazy narrowing calculus for PRS $\mathcal{R}$, called LN. LN performs higher-order $\mathcal{R}$-preunification.
  - Main challenge: reduce the search space for solutions
- since 2000: several refinements of LN which reduce nondeterminism have been proposed.