A Bottom-Up Semantics for FLP

Giovanni Bacci, Marco Comini

6th October 2009

Outline

- Main features of Functional Logic Programs
- Operational Sematics
- Bottom-up Semantics Operator (examples)
- Abstract Semantics (examples)
- Conclusions & Future Works

Functional Logic Languages (FLP)

Are defined over a bipartite signature $\Sigma := \mathcal{C} \uplus \mathcal{D}$

- **Overlapping Rules**

$$R_1 : x \mathbin{?} y \longrightarrow x$$
$$R_2 : x \mathbin{?} y \longrightarrow y$$

$$1 \mathbin{?} 2 \xrightarrow{R_1} A \qquad 1 \mathbin{?} 2 \xrightarrow{R_2} B$$

NON-DETERMINISTIC FUNCTIONS!!

Example:

$$R_3 : \; reach(x) \longrightarrow x \mathbin{?} reach(adj(x))$$
$$\text{where } \forall i, k \; (i, j_k) \in E. \; adj(i) \longrightarrow j_1 \mathbin{?} \dots \mathbin{?} j_n$$

Functional Logic Languages (FLP)

- **Logical variables**

$R_1: \ True \ \&\& \ y \to y$
$R_2: \ path(x) \to [x]$
$R_3: \ path(x) \to adj(x, y) \ \&\& \ (x : path(y))$
where $\forall (i,j) \in E. \ adj(i,j) \to True$

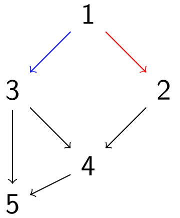
Narrowing:
$t \leadsto_\sigma s$ iff $\exists \sigma. \ \sigma(t) \to s$

$$path(1) \overset{\epsilon}{\leadsto} [1]$$

$$path(1) \overset{\epsilon}{\leadsto} adj(1, y) \ \&\& \ (1 : path(y))$$

$$adj(1, y) \ \&\& \ (1 : path(y)) \overset{\{y/3\}}{\leadsto} \vdots$$

$$adj(1, y) \ \&\& \ (1 : path(y)) \overset{\{y/2\}}{\leadsto} True \ \&\& \ (1 : path(2)) \overset{\epsilon}{\leadsto} (1 : path(2)) \overset{\epsilon}{\leadsto} [1, 2]$$



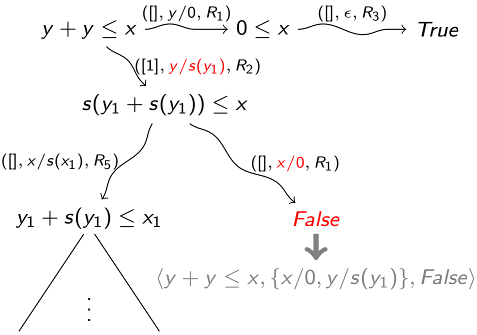
Needed Narrowing Strategy [Antoy & Hanus]

Example:

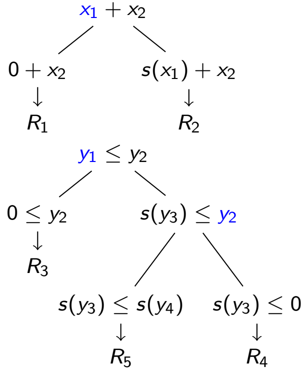
$R_1 : 0 + y \to y$
$R_2 : s(x) + y \to s(x + y)$

$R_3 : 0 \le y \to True$
$R_4 : s(x) \le 0 \to False$
$R_5 : s(x) \le s(y) \to x \le y$

... running a goal



Definitional Trees:



Toward a compositional Sementics

Definition (Compositionality)

$[\![s(t_1,\ldots,t_n)]\!]_P = op_s([\![t_1]\!]_P,\ldots,[\![t_n]\!]_P)$ where $s \in \Sigma = \mathcal{C} \uplus \mathcal{D}$

A natural notion of semantics:

$$[\![t]\!]_P := \{\langle t,\sigma,v\rangle \mid t \leadsto^*_\sigma v,\ \ v \text{ is a value}\}$$

$$[\![P]\!] := \bigcup\{[\![t]\!]_P \mid t \text{ is an expression}\}$$

IS NOT COMPOSITIONAL

since, the semantics of the previous program is:

$$\begin{aligned}[\![P]\!] = {} & \{\langle x_1+x_2,\{x_1/s^n(0)\},s^n(x_2)\rangle \mid n \geq 0\} \cup \\ & \{\langle x_1 \leq x_2,\{x_1/s^n(0),x_2/s^n(x_2')\},\mathit{True}\rangle \mid n \geq 0\} \\ & \{\langle x_1 \leq x_2,\{x_1/s^{n+1}(x_1'),x_2/s^n(0)\},\mathit{False}\rangle \mid n \geq 0\}\end{aligned}$$

$$[\![y+y]\!]_P = \{\langle y+y,\{y/s^n(0)\},s^{2n}(0)\rangle \mid n \geq 0\}$$

..but $y+y \leq x \leadsto^*_{\{y/s(y_1),x/0\}} \mathit{False}$

A Fix-point Operator over Narrowing Trees

$$T_P : \mathbb{C} \to \mathbb{C} \text{ where } \mathbb{C} := (PP \to NT)_{/\cong}$$
$$T_P(\mathcal{I}) := \lambda f(\vec{x}).\ \textstyle\sum_{f(\vec{t})\to r\in P} (\!| f(\vec{x}); \{\langle [], \{\vec{x}/\vec{t}\}, \mathcal{E}[\![r]\!]_{\mathcal{I}}\rangle\} |\!)$$

where $\mathcal{E}[\![]\!]_{\mathcal{I}}$ is the evaluation function:

$$\mathcal{E}[\![x]\!]_{\mathcal{I}} := (\!| x; \emptyset |\!)$$
$$\mathcal{E}[\![c(t_1,\dots,t_n)]\!]_{\mathcal{I}} := (\!| c(x_1,\dots,x_n); \emptyset |\!) \bullet \Phi[x/\mathcal{E}[\![t_i]\!]_{\mathcal{I}}]_{i=1,\dots,n}$$
$$\mathcal{E}[\![f(t_1,\dots,t_n)]\!]_{\mathcal{I}} := \mathcal{I}(f(x_1,\dots,x_n)) \bullet \Phi[x/\mathcal{E}[\![t_i]\!]_{\mathcal{I}}]_{i=1,\dots,n}$$

Theorem (Soundness & Completeness)

- $lfp(T_P)(f(\vec{x})) =_{\mathbb{C}} \{d \mid d \text{ is a derivation starting from } f(\vec{x})\}_{/\cong}$
- $\mathcal{E}[\![t]\!]_{lfp(T_P)} =_{\mathbb{C}} \{d \mid d \text{ is a derivation starting from } t\}_{/\cong}$

Toward a more abstract Semantics

The previous semantics is very precise ... even too much

We are able to distinguish too many programs:

$$
\begin{array}{l}
f \longrightarrow g \\
g \longrightarrow 0 \\
g \longrightarrow 1
\end{array}
\qquad
\llbracket P_1 \rrbracket = \left\{
\begin{array}{l}
\begin{tikzcd}
 & & 0 \\
f \mapsto f \arrow[r, "\epsilon"] & g \arrow[ur, "\epsilon"] \arrow[dr, "\epsilon"] & \\
 & & 1
\end{tikzcd} \\
\begin{tikzcd}
 & 0 \\
g \mapsto g \arrow[ur, "\epsilon"] \arrow[dr, "\epsilon"] & \\
 & 1
\end{tikzcd}
\end{array}
\right.
$$

$$
\begin{array}{l}
f \longrightarrow 0 \\
f \longrightarrow 1 \\
g \longrightarrow 0 \\
g \longrightarrow 1
\end{array}
\qquad
\llbracket P_1 \rrbracket = \left\{
\begin{array}{l}
\begin{tikzcd}
 & 0 \\
f \mapsto f \arrow[ur, "\epsilon"] \arrow[dr, "\epsilon"] & \\
 & 1
\end{tikzcd} \\
\begin{tikzcd}
 & 0 \\
g \mapsto g \arrow[ur, "\epsilon"] \arrow[dr, "\epsilon"] & \\
 & 1
\end{tikzcd}
\end{array}
\right.
$$

The Context Tree Abstraction

$$
\begin{aligned}
& f(g(x),x) \\
& \quad \rightsquigarrow_{\{x/B(x_1)\}} \\
& f(C(g(x_1)),B(x_1)) \\
& \quad \rightsquigarrow_{\{x_1/B(x_2)\}} \\
& f(C(C(g(x_2))),B(B(x_2))) \\
& \quad \rightsquigarrow_{\epsilon} \\
& g(x_2) \\
& \quad \rightsquigarrow_{\{x_2/B(x_3)\}} \\
& {\color{red}C(}g(x_3){\color{red})} \\
& \quad \rightsquigarrow_{\{x_3/B(x_4)\}} \\
& {\color{red}C(C(}g(x_4){\color{red}))}
\end{aligned}
$$

$$\xrightarrow{\ \Gamma\ }$$

$$\alpha(\mathcal{I}) := \lambda f(\vec{x}).\ \Gamma(f(\vec{x}))$$

Functions behaves as a black box

$$
\begin{aligned}
& \blacksquare \\
& \quad \rightsquigarrow_{\{x/B(x_1)\}} \\
& \quad \rightsquigarrow_{\{x_1/B(x_2)\}} \\
& \quad \rightsquigarrow_{\epsilon} \\
& \quad \rightsquigarrow_{\{x_2/B(x_3)\}} \\
& {\color{red}C(}\blacksquare{\color{red})} \\
& \quad \rightsquigarrow_{\{x_3/B(x_4)\}} \\
& {\color{red}C(C(}\blacksquare{\color{red}))}
\end{aligned}
$$

Inner steps are compressed

We just remember how the outermost context grows

The Abstract Fix-point Operator

Using standard techniques from Abstract Interpretation*, we defined it just by means of Γ and T_P:

$$T_P^\alpha \colon \mathbb{A} \to \mathbb{A} \text{ where } \mathbb{A} := (PP \to CT)_{/\cong}$$
$$T_P^\alpha(\mathcal{I}^\alpha) = \lambda f(\vec{x}).\ (\!|\blacksquare; S|\!)$$

where

$$\begin{aligned} S = \{\langle [], \{\vec{x}/\vec{t}\}, \widetilde{\mathcal{E}}[\![r]\!]_{\mathcal{I}^\alpha}\rangle \mid\ & f(\vec{x}) \to r \in P,\ r \text{ is } \mathcal{C} \text{ rooted}\} \\ \{\langle [], \{\vec{x}/\vec{t}\} \circ \sigma, T\rangle \mid\ & f(\vec{x}) \to r \in P,\ r \text{ is } \mathcal{D} \text{ rooted} \\ & \langle p, \sigma, T\rangle \in steps(\widetilde{\mathcal{E}}[\![r]\!]_{\mathcal{I}^\alpha})\} \end{aligned}$$

and $\widetilde{\mathcal{E}}[\![]\!]_{\mathcal{I}^\alpha}$ is the abstract evaluation function.

Lemma (Our abstraction is precise)

$$\forall \mathcal{I}.\ \forall t.\ \alpha(\mathcal{E}[\![t]\!]_{\mathcal{I}}) = \widetilde{\mathcal{E}}[\![t]\!]_{\alpha(\mathcal{I})}$$

It is able to characterize the **computed answer beavior**

Concusions

- we characterize FLP semantics using a bottom-up construction
- the semantics is goal-indipendent
- we found a more abstract semantics which precisely describe the computed answer behavior
- compositionality gives us the possibility to
  - define incremental and modular analysis tools
  - ... as well as verification tools.

Future works on CHR

Gabrielli & Meo defined a fully abstract fix-point semantics for CHR and they concluded saying:

> *[...] it would be desirable to introduce in the semantics the minimum amount of information needed to obtain compositionality, while preserving correctness. In other words, it would be desirable to obtain a fully abstract semantics for data sufficient answers. [...]*

[Gabrielli & Meo]