A Rho calculus for graph rewriting

Clara Bertolissi

joint work with

Horatiu Cirstea, and Claude Kirchner (LORIA)

&

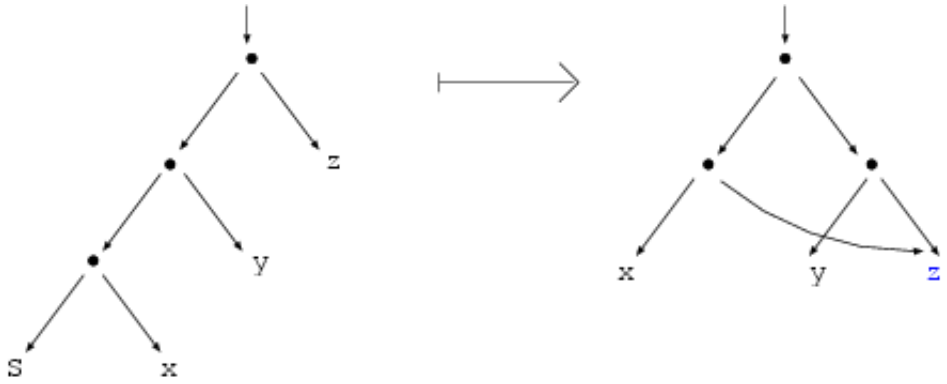
Paolo Baldan, and Furio Honsell (Univ.UDINE/VENEZIA)

Term Graph rewriting: directed acyclic graphs

Rewrite rules often ask for duplication of subterms:

$$Sxyz \;\rightarrow\; xz(yz)$$

How to save space in actual implementations?
Working with pointers to share subterms.



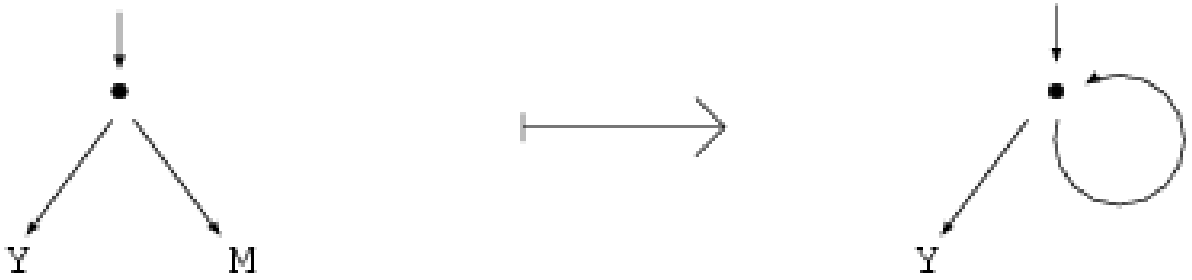
One step further: cyclic graphs

Cycles arise naturally in recursive structures.

The optimized representation of the fixed point combinator

$$Y\ M\ \rightarrow\ M\ (Y\ M)$$

is using a cyclic graph



Graph rewriting: different approaches

- categorical approach *(push-out diagrams)*
  [Corradini,Montanari,Gadducci95], . . .

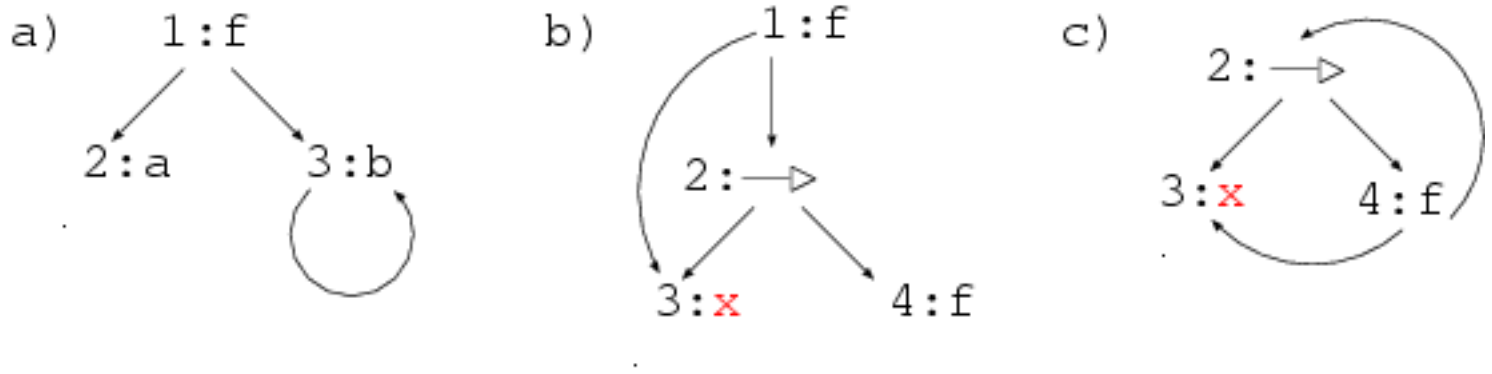
- implementation oriented systems *(pointers, redirections)*
  [Barendregt et all87],[Plump98],[Kennaway94],. . .

- equational representation *(set of recursive equations)*
  [Ariola,Klop97], . . .

1) GRHO-$\mathcal{L}$: labels as pointers

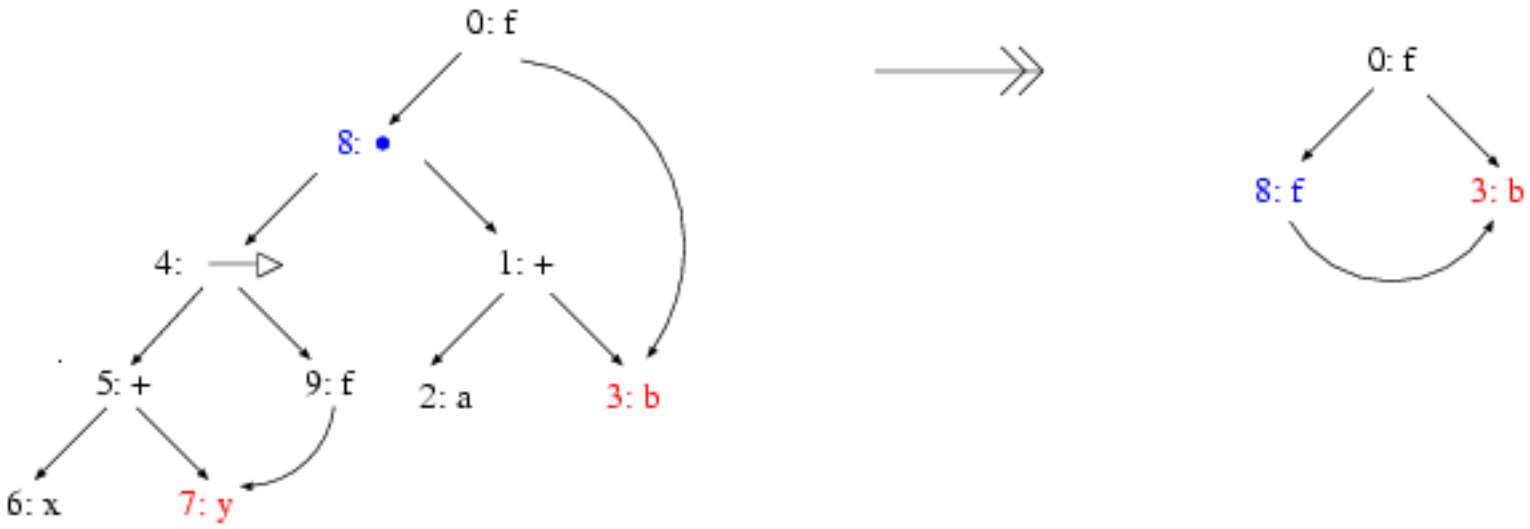
$$
\begin{array}{rcll}
\mathcal{G} & ::= & \langle \mathcal{L} \mid \quad \rangle & labels \\
 & | & \langle \mathcal{L} \mid \mathcal{L} = \mathcal{X} \rangle & \\
 & | & \langle \mathcal{L} \mid \mathcal{L} = \mathcal{K} \rangle & \\
 & | & \langle \mathcal{L} \mid \mathcal{L} = \mathcal{G} \rightarrow \mathcal{G} \rangle & \\
 & | & \langle \mathcal{L} \mid \mathcal{L} = [\mathcal{G} \ll \mathcal{G}]\mathcal{G} \rangle & \\
 & | & \langle \mathcal{L} \mid \mathcal{L} = \mathcal{G}\ \mathcal{G} \rangle & \\
 & | & \langle \mathcal{L} \mid \mathcal{L} = \mathcal{G}; \mathcal{G} \rangle & \\
 & | & \langle \mathcal{L} \mid \mathcal{L} = \mathcal{G}, \ldots, \mathcal{L} = \mathcal{G} \rangle & declarations
\end{array}
$$

Examples



- Every node has its label.
- The graph b) is not well-formed (x is free and bound)
- The graph c): how to define its scope ($i.e.$ for performing unwinding)?

Example of reduction



- Solution of the matching $\sigma = \{6/2, 7/3\}$
- Instantiation of the rhs by edge redirection
- Redirection of the head of the redex to the rhs of the rule
- Garbage collection

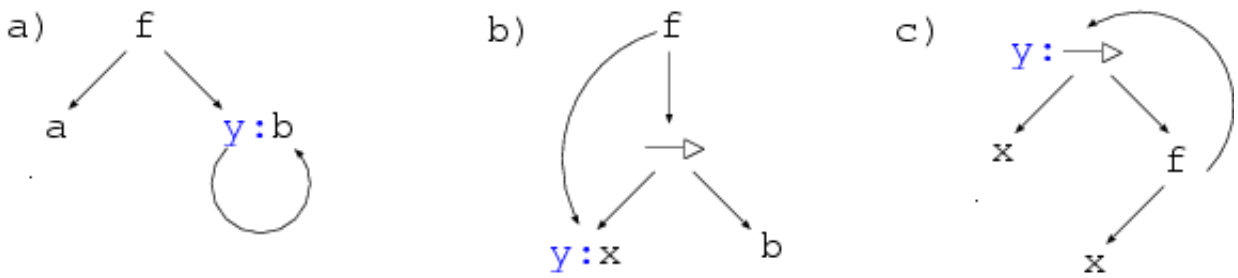
Advantages/Drawbacks of GRHO-$\mathcal{L}$

- *Pro*:
  - Variables are maximally shared
  - Possibility of dealing with cycles in rewrite rules and matching
  - Matching algorithm suitable for different theories
- *Anti*:
  - Problems in the correspondence between term rewriting and graph rewriting.
  - Evaluation rules difficult to write in a compact way
  - Matching needs non local contexts

2) GRHO-$\mathcal{G}$: generalising cyclic lambda graphs

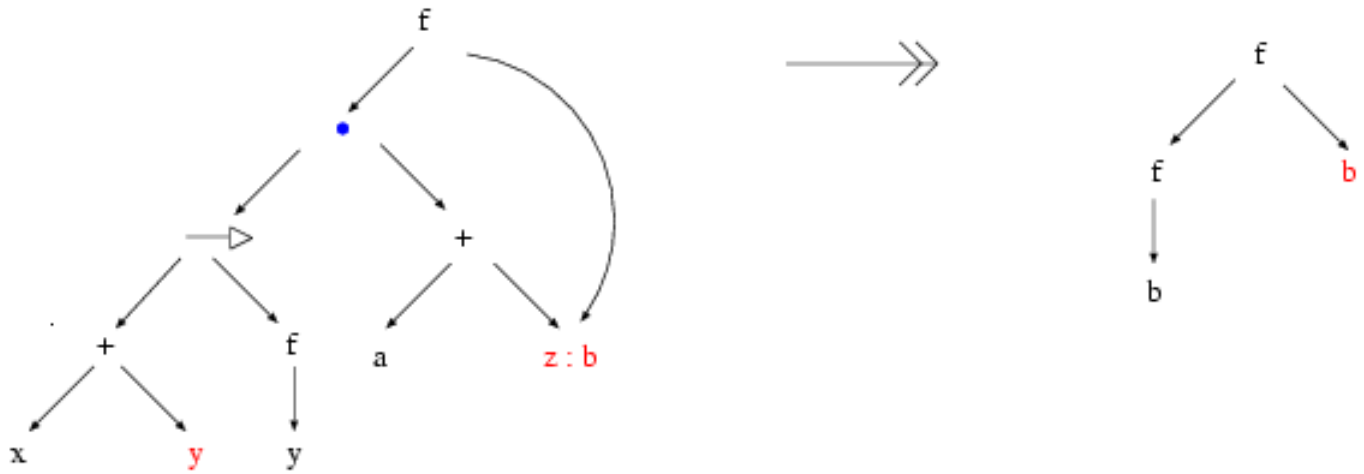
$$
\begin{array}{rcll}
\mathcal{G} & ::= & \mathcal{X} & \\
 & | & \mathcal{K} & \\
 & | & \mathcal{G} \rightarrowtriangle \mathcal{G} & \\
 & | & [\mathcal{G} \ll \mathcal{G}]\mathcal{G} & \\
 & | & \mathcal{G}\ \mathcal{G} & \\
 & | & \mathcal{G};\mathcal{G} & \\
 & | & \langle \mathcal{G} \mid \mathcal{X} = \mathcal{G}, \ldots, \mathcal{X} = \mathcal{G} \rangle & letrec
\end{array}
$$

Examples



- Only shared objects have a name
- Two kind of bound variables: by the arrow and by the letrec construction
- The graph b) $\langle f(\langle y \mid y = x\rangle \rightarrowtriangle b, y)\rangle$ is not allowed in this syntax: y cannot be seen outside its letrec construction.
- The scope of the graph c) $\langle y \mid y = x \rightarrowtriangle f(x, y)\rangle$ is clear: y is bound by the letrec construction, not by the arrow.

Example of reduction



- Solution of the matching: we add the substitution $x = a, y = b$ in the list of equations of the rhs.
- Instantiation of the rhs by applying the substitution: $\langle f(y) \mid x = a, y = b \rangle \rightarrow_s \langle f(b) \mid x = a, y = b \rangle$
- Garbage collection: $\langle f(b) \mid x = a, y = b \rangle \rightarrow_{gc} f(b)$

Advantages/Drawbacks of GRHO-$\mathcal{G}$

- *Pro*:
  - Matching can be defined at the calculus level
  - Problems of scope are partially solved by the local definition of names as variables
- *Anti*:
  - Lost of some sharing
  - The natural notion of substitution causes non confluence
  - Matching: not easy generalisation to cyclic graphs

Comparing the two choises

- GRHO-$\mathcal{G}$
  - Using variables instead of labels allows a better control on the scope
  - Evaluation rules are easier to describe syntactically
- GRHO-$\mathcal{L}$
  - Redirections of pointers avoid copying when applying substitution
  - The matching algorithm can be easily adapted according to the graph relation we choose: homomorphism, bisimulation, . . .

Open problems

- Analyse different matching theories, $i.e.$ corresponding to functional and relational graph bisimulation.
- Adequacy property: some relation between ρ-graph reductions end ρ-reductions on correspondent terms.
- Study the confluence problem: which notion of substitution is the more convenient?

Higher-order term rewriting

- Representation of the CRS (term-rewrite system + abstractor) in the RHO
  - Definition of the $RHO_{\P}$ (HO matching theory)
  - Translation of CRS-components into RHO-terms
  - Correction and completeness of the translation

- Corollary: indirect representation of HRS in the RHO (using the equivalence of CRS and HRS in [vOvR93])

Translation of the CRS in the RHO

$$CRS \longrightarrow RHO$$

$$\mathcal{R} \longrightarrow \overline{\mathcal{R}}$$

$$\mathcal{A} \longrightarrow \overline{\mathcal{A}}$$

Translation of the HRS in the RHO

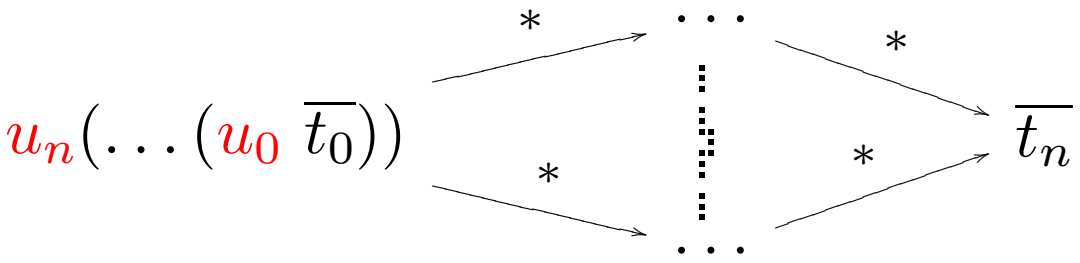
$$HRS \longrightarrow CRS + Beta \longrightarrow RHO$$

$$\mathcal{R} \longrightarrow \mathcal{R}' + Beta \longrightarrow \overline{\mathcal{R}' + Beta}$$

$$\mathcal{A} \longrightarrow \mathcal{A}' \longrightarrow \overline{\mathcal{A}'}$$

Correction and completeness of the translation

Given a CRS-derivation $t_0 \mapsto\!\!\!\!\rightarrow_{CRS} t_n$
There exist n ρ-terms $u_n \ldots u_0$ such that every correspondent RHO-derivation terminates and converges to $\overline{t_n}$



- Preservation of properties:

- Termination
- Confluence
- Orthogonality