Rewriting calculus:

an introduction

Horatiu Cirstea and Claude Kirchner

in close collaboration with
Gilles Barthe, Clara Bertolissi, Germain Faure, Luigi Liquori,
Benjamin Wack

Nancy March 11, 2004

Rewrite Systems and Lambda Calculus: some dates

1930 Lambda Calculus
1970 Formal definition of rewrite rule and rewrite relation
1975 Rewriting Modulo
1980 CRS
1980 OBJ (rewriting modulo, local strategies)
1990 Combination of lambda calculus and rewriting
1990 Higher-order rewriting systems
1990 Rewriting logic
1995 Deduction Modulo
1997 Rewriting calculus
2000 Combination of CC and rewriting

Pre-history

1980 Jouannaud-Lescanne: Nancy rewriting school
1984 Reve workshop in Nancy: implicit ideas about computation and deduction by rewriting
1984-86 OBJ
1990 Ecolog Nancy-Orsay
1992 ELAN-00 (deduction and computation, user defined strategies) MV
1995 Study of strategies PB
1996 First ideas about the rewriting calculus
2000 Formal definition of the rewriting calculus HC

History

- **Term rewriting from a functional point of view**
  - ➥ rewrite rules as functions applied using an explicit application function
- **The ρ-calculus**
  - ➥ confluence (evaluation strategies) and termination (simply typed calculus)
  - ➥ operational semantics of ELAN
- **Rho-calculus**
  - ➥ representation of object calculi (ObjCal and LambdaObjCal)
  - ➥ sophisticated type systems - the Rho-cube
- **Exceptional Rho-calculus**
  - ➥ delayed matching constraints
  - ➥ exception handling mechanism

The untyped rewriting calculus - ρ-calculus

The Untyped Syntax

$$\mathcal{P} \quad ::= \quad \mathcal{T}$$

$$\mathcal{T} \quad ::= \quad \mathcal{X} \mid \mathcal{K} \mid \mathcal{P} \twoheadrightarrow \mathcal{T} \mid \mathcal{T} \bullet \mathcal{T} \mid [\mathcal{P} \ll \mathcal{T}]\mathcal{T} \mid \mathcal{T}; \mathcal{T}$$

1. $T_1 \twoheadrightarrow T_2$ denotes a *rule abstraction* with pattern T_1 and body T_2
   ... the free variables of T_1 are bound in T_2

2. $[T_1 \ll T_2]T_3$ denotes a *delayed matching constraint*
   ... the free variables of T_1 are bound in T_3 but not in T_2

3. The terms can be also *structures* built using the symbol ";"

4. We work modulo the *α-convention* and the *hygiene-convention*

Some ρ-terms

$(\mathcal{X} \to \mathcal{X})\bullet a$ similar to the λ-term $(\lambda x.x)\ a$

$(\mathcal{X} \to \mathcal{X}\bullet\mathcal{X})\bullet(\mathcal{X} \to \mathcal{X}\bullet\mathcal{X})$ the well-known λ-term $(\omega\omega)$

Some ρ-terms

$(\mathcal{X} \twoheadrightarrow \mathcal{X})\bullet a$ similar to the λ-term $(\lambda x.x)\ a$

$(\mathcal{X} \twoheadrightarrow \mathcal{X}\bullet\mathcal{X})\bullet(\mathcal{X} \twoheadrightarrow \mathcal{X}\bullet\mathcal{X})$ the well-known λ-term $(\omega\omega)$

$(a \twoheadrightarrow b)\bullet a$ the application of the rule $a \twoheadrightarrow b$ to the term a

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet f(a,b)$ a classical rewrite rule application

Some ρ-terms

$(\mathcal{X} \twoheadrightarrow \mathcal{X})\bullet a$ similar to the λ-term $(\lambda x.x)\ a$

$(\mathcal{X} \twoheadrightarrow \mathcal{X}\bullet\mathcal{X})\bullet(\mathcal{X} \twoheadrightarrow \mathcal{X}\bullet\mathcal{X})$ the well-known λ-term $(\omega\omega)$

$(a \twoheadrightarrow b)\bullet a$ the application of the rule $a \twoheadrightarrow b$ to the term a

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet f(a,b)$ a classical rewrite rule application

$(a \twoheadrightarrow b; a \twoheadrightarrow c)\bullet a$ "non-deterministic" application

Some abbreviations

$T(T_1, \ldots, T_n) \triangleq T \bullet T_1 \ldots \bullet T_n$ function-application (

$(T_i)^{i=1\ldots n} \triangleq T_1; \ldots; T_n$ structure/object ($n \in$

$T_1.T_2 \triangleq T_1 \bullet T_2 \bullet T_1$ Kamin's self-applicati

Operator	Associate	Priority
$\_ \bullet \_$	Left	$>$
$[\_ \ll \_]\_$	Right	$>$
$\_ \twoheadrightarrow \_$	Right	$>$
$\_ ; \_$	Right	$>$

Matching Equations and Solutions

1. A *match equation* is a formula of the form $T_1 \prec\!\!\prec T_2$.

2. A *matching system* $\mathsf{T} \stackrel{\triangle}{=} \bigwedge_{i=0\ldots n} T_i \prec\!\!\prec T_i'$ is a conjunction of

   match equations, where $\wedge$ is associative, commutative, and idempotent.

3. A matching system T is *successful* if it is empty or:

   (a) has the shape $\bigwedge_{i=0\ldots n} \mathcal{X}_i \prec\!\!\prec T_i \bigwedge_{j=0\ldots m} \mathcal{K}_j \prec\!\!\prec \mathcal{K}_j$;

   (b) for all $h, k = 0 \ldots n$, such that $h \neq k$, we have $\mathcal{X}_h \neq \mathcal{X}_k$;

4. A substitution $\sigma_{\mathsf{T}} = \{T_1/\mathcal{X}_1 \cdots T_n/\mathcal{X}_n\}$ is the solution of a successful matching system T. The set of solutions of T is denoted by $\mathcal{S}ol(\mathsf{T})$.

Reduction relies on matching power

Matching is parametrized over a theory $\mathbb{T}$ and an order $\prec$ on substitutions

$$\mathcal{S}ol(T_1 \prec\!\!\prec_{\mathbb{T}} T_2) = \sigma_1, \ldots, \sigma_n, \ldots$$

$$\sigma \in \mathcal{S}ol(T_1 \prec\!\!\prec_{\mathbb{T}} T_2) \;\Leftrightarrow \models_{\mathbb{T}} \sigma(T_1) = T_2$$

$$\sigma_1 \prec \ldots \prec \sigma_n \qquad (n \leq \infty)$$

THEORIES ALGORITHM Semantics

The Small-step Reduction Semantics

$$(P \twoheadrightarrow A)\bullet B \;\to_{\rho}\; [P \ll B]A$$

$$[P \ll B]A \;\to_{\sigma}\; A\theta_1; \ldots; A\theta_n; \ldots$$

with $\{\theta_1, \ldots, \theta_n, \ldots\} = \mathcal{S}ol($

$$(A; B)\bullet C \;\to_{\delta}\; A\bullet C; B\bullet C$$

Intuition on the small-step Semantics

$$(P \twoheadrightarrow A)\bullet B \;\to_\rho\; [P \ll B]A$$
$$\to_\sigma\; A\theta$$
$$\text{if } P\theta =_{\mathbb{T}} B$$

$$(P \twoheadrightarrow A)\bullet B \;\to_\rho\; [P \ll B]A$$
$$\text{STOP!}$$
$$\text{if } \nexists\theta.\ P\theta =_{\mathbb{T}} B$$

Matching Examples Confluence

Some ρ-reductions

$(\mathcal{X} \twoheadrightarrow \mathcal{X})\bullet a$

$(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X}))\bullet(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X}))$

$(a \twoheadrightarrow b)\bullet a$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(f(a,b))$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(g(a,b))$

$(a \twoheadrightarrow b; a \twoheadrightarrow c)\bullet a$

Some ρ-reductions

$(\mathcal{X} \twoheadrightarrow \mathcal{X})\bullet a \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} a$

$(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X}))\bullet(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X}))$

$(a \twoheadrightarrow b)\bullet a$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(f(a,b))$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(g(a,b))$

$(a \twoheadrightarrow b; a \twoheadrightarrow c)\bullet a$

Some ρ-reductions

$(\mathcal{X} \twoheadrightarrow \mathcal{X})\bullet a \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; a$

$(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X}))\bullet(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X})) \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \{\omega\bullet\omega\} \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \ldots$

$(a \twoheadrightarrow b)\bullet a$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(f(a,b))$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(g(a,b))$

$(a \twoheadrightarrow b; a \twoheadrightarrow c)\bullet a$

Some ρ-reductions

$(\mathcal{X} \twoheadrightarrow \mathcal{X})\bullet a \qquad \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; a$

$(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X}))\bullet(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X})) \qquad \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; \{\omega\bullet\omega\} \; \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; \ldots$

$(a \twoheadrightarrow b)\bullet a \qquad \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; b$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(f(a,b))$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(g(a,b))$

$(a \twoheadrightarrow b; a \twoheadrightarrow c)\bullet a$

Some ρ-reductions

$(\mathcal{X} \twoheadrightarrow \mathcal{X})\bullet a \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; a$

$(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X}))\bullet(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X})) \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; \{\omega\bullet\omega\} \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; \ldots$

$(a \twoheadrightarrow b)\bullet a \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; b$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(f(a,b)) \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; [f(\mathcal{X},\mathcal{Y}) \ll f(a,b)]g(\mathcal{X}$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(g(a,b))$

$(a \twoheadrightarrow b; a \twoheadrightarrow c)\bullet a$

Some ρ-reductions

$(\mathcal{X} \twoheadrightarrow \mathcal{X})\bullet a \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; a$

$(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X}))\bullet(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X})) \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; \{\omega\bullet\omega\} \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; \ldots$

$(a \twoheadrightarrow b)\bullet a \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; b$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(f(a,b))$
$\mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; [f(\mathcal{X},\mathcal{Y}) \ll f(a,b)]g(\mathcal{X},\mathcal{Y}) \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; g(a,b)$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(g(a,b)) \qquad \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} \; [f(\mathcal{X},\mathcal{Y}) \ll g(a,b)]g(\mathcal{X},$

$(a \twoheadrightarrow b; a \twoheadrightarrow c)\bullet a$

Some ρ-reductions

$(\mathcal{X} \twoheadrightarrow \mathcal{X})\bullet a \qquad \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; a$

$(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X}))\bullet(\mathcal{X} \twoheadrightarrow (\mathcal{X}\bullet\mathcal{X})) \qquad \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; \{\omega\bullet\omega\} \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; \ldots$

$(a \twoheadrightarrow b)\bullet a \qquad \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; b$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(f(a,b))$
$\mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; [f(\mathcal{X},\mathcal{Y}) \ll f(a,b)]g(\mathcal{X},\mathcal{Y}) \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; g(a,b)$

$(f(\mathcal{X},\mathcal{Y}) \twoheadrightarrow g(\mathcal{X},\mathcal{Y}))\bullet(g(a,b))$
$\mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; [f(\mathcal{X},\mathcal{Y}) \ll g(a,b)]g(\mathcal{X},\mathcal{Y})$

$(a \twoheadrightarrow b; a \twoheadrightarrow c)\bullet a \qquad \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; (a \twoheadrightarrow b)\bullet a; (a \twoheadrightarrow c)\bullet a \mapsto\!\!\!\!\rightarrow_{\rho\alpha\delta} \; b; c$

Simple Success Reduction

$$
\begin{aligned}
\underline{(f(\mathcal{X}) \twoheadrightarrow (3 \twoheadrightarrow 3)\mathcal{X})\bullet f(3)} &\mapsto_{\rho} \underline{[f(\mathcal{X}) \ll f(3)]((3 \twoheadrightarrow 3} \\
&\mapsto_{\sigma} \underline{(3 \twoheadrightarrow 3)\bullet 3} \\
&\mapsto_{\rho} \underline{[3 \ll 3]3} \\
&\mapsto_{\sigma} 3
\end{aligned}
$$

$$
\begin{aligned}
(f(\mathcal{X}) \twoheadrightarrow \underline{(3 \twoheadrightarrow 3)\mathcal{X}})\bullet f(3) &\mapsto_{\rho} \underline{(f(\mathcal{X}) \twoheadrightarrow [3 \ll \mathcal{X}]3)\bullet f} \\
&\mapsto_{\rho} \underline{[f(\mathcal{X}) \ll f(3)]([3 \ll \mathcal{X}} \\
&\mapsto_{\sigma} \underline{[3 \ll 3]3} \\
&\mapsto_{\sigma} 3
\end{aligned}
$$

Simple Failure Reduction

$$\underline{(f(\mathcal{X}) \twoheadrightarrow (3 \twoheadrightarrow 3)\mathcal{X})\bullet f(4)} \mapsto_\rho \underline{[f(\mathcal{X}) \ll f(4)]((3 \twoheadrightarrow 3}$$

$$\mapsto_\sigma \underline{(3 \twoheadrightarrow 3)\bullet 4}$$

$$\mapsto_\rho \underline{[3 \ll 4]3}$$

$$(f(\mathcal{X}) \twoheadrightarrow \underline{(3 \twoheadrightarrow 3)\mathcal{X}})\bullet f(4) \mapsto_\rho \underline{(f(\mathcal{X}) \twoheadrightarrow [3 \ll \mathcal{X}]3)\bullet f}$$

$$\mapsto_\rho \underline{[f(\mathcal{X}) \ll f(4)]([3 \ll \mathcal{X}}$$

$$\mapsto_\sigma \underline{[3 \ll 4]3}$$

Rewriting: The relation, the logic and the calculus

Given a set $\mathcal{R}$ of rewrite rules $(l_i \twoheadrightarrow r_i)$, we can define:

Rewriting: The relation, the logic and the calculus

Given a set $\mathcal{R}$ of rewrite rules $(l_i \twoheadrightarrow r_i)$, we can define:

The rewrite relation : $t \longrightarrow_{\mathcal{R}} t'$
i.e. the smallest relation containing the rewrite rules and stable by context et substitution.

Rewriting: The relation, the logic and the calculus

Given a set $\mathcal{R}$ of labeled rewrite rules $([\ell_i]\ l_i \twoheadrightarrow r_i)$, we can define:

Rewriting: The relation, the logic and the calculus

Given a set $\mathcal{R}$ of labeled rewrite rules $([\ell_i]\ l_i \twoheadrightarrow r_i)$, we can define:

The rewriting logic: $\mathcal{R} \vdash t \Rightarrow t'$ [José Meseguer, TCS1992]

Rewriting: The relation, the logic and the calculus

Given a set $\mathcal{R}$ of labeled rewrite rules $([\ell_i]\ l_i \twoheadrightarrow r_i)$, we can define:

The rewriting logic : $\mathcal{R} \vdash t \Rightarrow t'$ [José Meseguer, TCS1992]

Formulas are sequents of the form

$$\pi : t \Rightarrow t'$$

where π is a proof term, built on $\mathcal{F} \cup L \cup \{;\}$
recording the proof of the sequent.
Models are computation spaces (quotiented set of proof terms).

$$\mathcal{R} \ \vdash \ \pi : t \Rightarrow t'$$

if $\pi : t \Rightarrow t'$ can be obtained by finite application of the following rules.

Reflexivity For any $t \in \mathcal{T}(\mathcal{F})$:

$$t : t \Rightarrow t$$

Congruence For any $f \in \mathcal{F}$ with $arity(f) = n$:

$$\frac{\pi_1 : t_1 \Rightarrow t'_1 \quad \ldots \quad \pi_n : t_n \Rightarrow t'_n}{f(\pi_1, \ldots, \pi_n) : f(t_1, \ldots, t_n) \Rightarrow f(t'_1, \ldots, t'_n)}$$

Replacement For any $\ell : l(x_1, \ldots, x_n) \Rightarrow r(x_1, \ldots, x_n) \in R$,

$$\frac{\pi_1 : t_1 \Rightarrow t'_1 \quad \ldots \quad \pi_n : t_n \Rightarrow t'_n}{\ell(\pi_1, \ldots, \pi_n) : l(t_1, \ldots, t_n) \Rightarrow r(t'_1, \ldots, t'_n)}$$

Transitivity

$$\frac{\pi_1 : t_1 \Rightarrow t_2 \qquad \pi_2 : t_2 \Rightarrow t_3}{\pi_1;\pi_2 \; : \; t_1 \Rightarrow t_3}$$

Rewriting: The relation, the logic and the calculus

Given a set $\mathcal{R}$ of rewrite rules $(l_i \twoheadrightarrow r_i)$, we can define:

Rewriting: The relation, the logic and the calculus

Given a set $\mathcal{R}$ of rewrite rules $(l_i \twoheadrightarrow r_i)$, we can define:

The rewriting calculus : $(lmim(\mathcal{R})\ \ t) \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} t'$

Why a new calculus?

Rewriting is nice, but

- the rewrite relation or the rewriting logic are difficult to control
- non-reducibility is impossible to express
- it is too close to first-order

Why a new calculus?

Rewriting is nice, but

- the rewrite relation or the rewriting logic are difficult to control
- non-reducibility is impossible to express
- it is too close to first-order

Lambda-calculus is great, but

- lacks of discrimination capabilities
- difficult to control

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

In "basic" rewriting calculus,

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

In "basic" rewriting calculus,

- rules are first class object

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

In "basic" rewriting calculus,

- rules are first class object
- application is explicit

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

In "basic" rewriting calculus,

- rules are first class object
- application is explicit
- decision of redex reduction is explicit

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

In "basic" rewriting calculus,

- rules are first class object
- application is explicit
- decision of redex reduction is explicit
- matching is a main explicit parameter

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

In "basic" rewriting calculus,

- rules are first class object
- application is explicit
- decision of redex reduction is explicit
- matching is a main explicit parameter
- results are first class

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

In "basic" rewriting calculus,

- rules are first class object
- application is explicit
- decision of redex reduction is explicit
- matching is a main explicit parameter
- results are first class

This allows for advanced calculi,

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

In "basic" rewriting calculus,

- rules are first class object
- application is explicit
- decision of redex reduction is explicit
- matching is a main explicit parameter
- results are first class

This allows for advanced calculi,

- with constraints explicit substitutions

A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.

In "basic" rewriting calculus,

- rules are first class object
- application is explicit
- decision of redex reduction is explicit
- matching is a main explicit parameter
- results are first class

This allows for advanced calculi,

- with constraints explicit substitutions
- to even dissociate binding from matching when abstracting

About the expressiveness of the rewriting calculus in $\mathbb{T}_\emptyset$

1. **Compiling the Λ into the ρ-calculus**. $\varphi : \Lambda \Rightarrow \rho$-calculus

   (a) $\varphi(X) = X$
   (b) $\varphi(\lambda X.M) = X \twoheadrightarrow \varphi(M)$
   (c) $\varphi(M\ N) = \varphi(M) \bullet \varphi(N)$

   Theorem: If $M \mapsto_\beta N$, then $\varphi(M) \mapsto\!\!\!\!\twoheadrightarrow_{\rho\varpi\delta} \varphi(N)$

About the expressiveness of the rewriting calculus in $\mathbb{T}_\emptyset$

1. **Compiling the Λ into the ρ-calculus**. $\varphi : \Lambda \Rightarrow \rho$-calculus

   (a) $\varphi(X) = X$
   (b) $\varphi(\lambda X.M) = X \twoheadrightarrow \varphi(M)$
   (c) $\varphi(M\ N) = \varphi(M) \bullet \varphi(N)$

   Theorem: If $M \mapsto_\beta N$, then $\varphi(M) \mapsto\!\!\!\!\rightarrow_{\rho\partial} \varphi(N)$

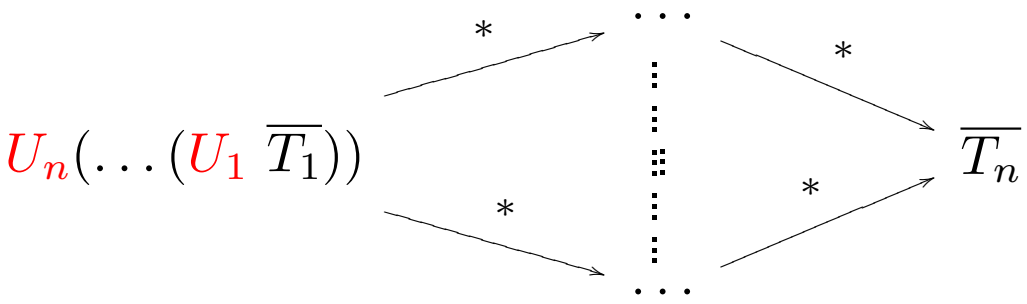
2. **Encoding Rewriting**

   (a) A rewrite system $\mathcal{R}$ can be represented as the structure containing all the rules
   (b) Reduction strategies can be encoded
   Theorem: If $T_1 \mapsto_{\mathcal{R}} T_2$, then $\exists\ T_{\mathcal{R}}$ such that $T_{\mathcal{R}} \bullet T_1 \mapsto\!\!\!\!\rightarrow_{\rho\partial} T_2$

About the expressiveness of the rewriting calculus

3. Representation of Higher-order term rewriting (CRS) (term-rewrite system + abstractor)
   - Definition of the $\rho_{\P}$-calculus (HO matching theory)
   - Translation of CRS-components into RHO-terms
   - Correction and completeness of the translation

Theorem: If $T_1 \mapsto_{\mathcal{R}} T_2 \ldots \mapsto_{\mathcal{R}} T_n$, then $\exists\ U_n \ldots U_1$ such that every correspondent RHO-derivation terminates and converges to $\overline{T_n}$



ρ-calculus and records

Record $=$ structure composed of rewriting rules, $i.e.$:

$$(T_i)^{i=1\ldots n} \quad \triangleq \quad T_1;\ldots;T_n \qquad (n \in \mathsf{Nat})$$

$$[m_i = T_i]^{i\in I} \quad \triangleq \quad (m_i \twoheadrightarrow T_i)^{i\in I}$$

$$[cx = 0, cy = 0] \quad \triangleq \quad (cx \twoheadrightarrow 0; cy \twoheadrightarrow 0)$$

Record selection $=$ the application of the record to the label, $i.e.$ $T_1.T_2$ as $T_1{\bullet}T_2$.

$$\begin{array}{lll} (cx \twoheadrightarrow 0; cy \twoheadrightarrow 0){\bullet}cx & \mapsto_{\delta} & (cx \twoheadrightarrow 0){\bullet}cx; (cy \twoheadrightarrow 0){\bullet}cx \\ & \mapsto\!\!\!\!\rightarrow_{\rho\varpi\delta} & 0; [cy \ll cx]0 \\ & =_{\mathbb{T}} & (0; \mathsf{stk}) =_{\mathbb{T}} 0 \quad (\mathbb{T}_{\varsigma\mathcal{O}bj}) \end{array}$$

. . . the matching theory

ρ-calculus and objects

Object = record with an explicit account of self, $i.e.$

$$[m_i = \varsigma(\mathcal{X}_i)T_i]^{i\in I} \quad \triangleq \quad (m_i \to \mathcal{X}_i \to T_i)^{i\in I}$$

Self-application = the application of an object to the object itself, $i.e.$

$$T_1.T_2 \quad \triangleq \quad T_1 \bullet T_2 \bullet T_1$$

Ex: $T \triangleq a \to S \to b$. Then:
$T.a \triangleq T \bullet a \bullet T \quad \mapsto\!\!\!\!\!\rightarrow_{\rho\partial} \quad (S \to b) \bullet T \quad \mapsto\!\!\!\!\!\rightarrow_{\rho\partial} \quad b$

Ex: $T \triangleq \omega \to S \to S.\omega$. Then:
$T.\omega \quad \mapsto\!\!\!\!\!\rightarrow_{\rho\partial} \quad (S \to S.\omega) \bullet T \quad \mapsto\!\!\!\!\!\rightarrow_{\rho\partial} \quad T.\omega \quad \mapsto\!\!\!\!\!\rightarrow_{\rho\partial} \quad \ldots$

A "ping-pong" object

Let $T \triangleq (ping \multimap S \multimap S.pong; pong \multimap S \multimap S.ping)$
Then:

$$
\begin{array}{rcl}
T.ping & \triangleq & T \bullet ping \bullet T \\
& \mapsto\!\!\!\!\rightarrow_{\rho\partial} & ((ping \multimap S \multimap S.pong) \bullet ping; \\
& & \ (pong \multimap S \multimap S.ping) \bullet ping) \bullet T \\
& \mapsto\!\!\!\!\rightarrow_{\rho\partial} & ((S \multimap S.pong); \mathsf{stk}) \bullet T \\
& \mapsto\!\!\!\!\rightarrow_{\rho\partial} & (S \multimap S.pong) \bullet T; \mathsf{stk} \bullet T \\
& \mapsto\!\!\!\!\rightarrow_{\rho\partial} & (S \multimap S.pong) \bullet T; \mathsf{stk} \\
& =_{\mathbb{T}} & (S \multimap S.pong) \bullet T \quad (\mathbb{T}_{\varsigma\mathcal{O}bj}) \\
& \mapsto\!\!\!\!\rightarrow_{\rho\partial} & T.pong \\
& \mapsto\!\!\!\!\rightarrow_{\rho\partial} & T.ping \\
& \mapsto\!\!\!\!\rightarrow_{\rho\partial} & \ldots
\end{array}
$$

Functional object update

Update $\quad (a.m := b) \triangleq (a; m \twoheadrightarrow b)$

$$
\begin{array}{rcl}
Point & \triangleq & val \twoheadrightarrow S \twoheadrightarrow v(1,1); \\
 & & get \twoheadrightarrow S \twoheadrightarrow S.val; \\
 & & set \twoheadrightarrow S \twoheadrightarrow v(X,Y) \twoheadrightarrow (S.val := S' \twoheadrightarrow v(X,Y))
\end{array}
$$

Then:

$$
\begin{array}{rcl}
Point.get & \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} & v(1,1) \\
Point.set(v(2,2)) & \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} & Point; (val \twoheadrightarrow S' \twoheadrightarrow v(2,2)) \\
Point.set(v(2,2)).get & \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} & v(1,1); v(2,2)
\end{array}
$$

Imperative object update

$Kill_m$ rule:

$$kill_m \quad \triangleq \quad (m \twoheadrightarrow X; Y) \twoheadrightarrow Y \qquad\qquad (\text{in } \mathbb{T}_{\varsigma\mathcal{O}bj})$$

Update $\qquad (a.m :=_I b) \triangleq (kill_m(a); m \twoheadrightarrow b)$

Imperative object update

$Kill_m$ rule:

$$kill_m \triangleq (m \twoheadrightarrow X; Y) \twoheadrightarrow Y \qquad \text{(in } \mathbb{T}_{\varsigma\mathcal{O}bj}\text{)}$$

Update $(a.m :=_I b) \triangleq (kill_m(a); m \twoheadrightarrow b)$

Then:

$$\begin{aligned} Point_I.get &\mapsto\!\!\!\!\rightarrow_{\rho\partial\delta} v(1,1) \\ Point_I.set(v(2,2)) &\mapsto\!\!\!\!\rightarrow_{\rho\partial\delta} val \twoheadrightarrow S' \twoheadrightarrow v(2,2); get \twoheadrightarrow \ldots, set \\ Point_I.set(v(2,2)).get &\mapsto\!\!\!\!\rightarrow_{\rho\partial\delta} v(2,2) \end{aligned}$$

More on objects in ρ-calculus

- Other examples
  - An object with "self-extension"
  - Inheritance in the ρ-calculus
- Encodings
  - The Lambda Calculus of Objects $\lambda\mathcal{O}bj$
  - The Object Calculus $\varsigma\mathcal{O}bj$
- More sophisticated examples
  - The Para object: labels as first-class entities
  - The object Daemon: methods as first-class entities
- Types

An object with "self-extension"

Let $t_1 \triangleq add_n \to S \to (S; n \to S' \to 1)$

$$(t_1.add_n).n \quad \mapsto\!\!\!\!\!\!\rightarrow_{\rho\infty\delta} \quad ((S \to (S; n \to S' \to 1))\bullet t_1).n$$

Back to objects

An object with "self-extension"

Let $t_1 \triangleq add_n \twoheadrightarrow S \twoheadrightarrow (S; n \twoheadrightarrow S' \twoheadrightarrow 1)$

$$
\begin{aligned}
(t_1.add_n).n \quad &\mapsto\!\!\!\!\!\rightarrow_{\rho\varpi\delta} \quad ((S \twoheadrightarrow (S; n \twoheadrightarrow S' \twoheadrightarrow 1))\bullet t_1).n \\
&\mapsto\!\!\!\!\!\rightarrow_{\rho\varpi\delta} \quad \underbrace{(t_1; n \twoheadrightarrow S' \twoheadrightarrow 1)}_{t_2}.n
\end{aligned}
$$

Back to objects

An object with "self-extension"

Let $t_1 \triangleq add_n \twoheadrightarrow S \twoheadrightarrow (S; n \twoheadrightarrow S' \twoheadrightarrow 1)$

$$
\begin{aligned}
(t_1.add_n).n \quad &\mapsto\!\!\!\!\!\!\rightarrow_{\rho\varpi\delta} \quad ((S \twoheadrightarrow (S; n \twoheadrightarrow S' \twoheadrightarrow 1))\bullet t_1).n \\
&\mapsto\!\!\!\!\!\!\rightarrow_{\rho\varpi\delta} \quad \underbrace{(t_1; n \twoheadrightarrow S' \twoheadrightarrow 1)}_{t_2}.n \\
&\triangleq \quad t_2\bullet n\bullet t_2
\end{aligned}
$$

Back to objects

An object with "self-extension"

Let $t_1 \triangleq add_n \twoheadrightarrow S \twoheadrightarrow (S; n \twoheadrightarrow S' \twoheadrightarrow 1)$

$$
\begin{array}{lll}
(t_1.add_n).n & \mapsto\!\!\!\!\!\rightarrow_{\rho\varpi\delta} & ((S \twoheadrightarrow (S; n \twoheadrightarrow S' \twoheadrightarrow 1))\bullet t_1).n \\
& \mapsto\!\!\!\!\!\rightarrow_{\rho\varpi\delta} & \underbrace{(t_1; n \twoheadrightarrow S' \twoheadrightarrow 1)}_{t_2}.n \\
& & \\
& \triangleq & t_2 \bullet n \bullet t_2 \\
& \mapsto\!\!\!\!\!\rightarrow_{\rho\varpi\delta} & ((add_n \twoheadrightarrow \ldots)\bullet n; (n \twoheadrightarrow S' \twoheadrightarrow 1)\bullet n)\bullet t_2
\end{array}
$$

Back to objects

An object with "self-extension"

Let $t_1 \triangleq add_n \twoheadrightarrow S \twoheadrightarrow (S; n \twoheadrightarrow S' \twoheadrightarrow 1)$

$$
\begin{array}{lll}
(t_1.add_n).n & \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & ((S \twoheadrightarrow (S; n \twoheadrightarrow S' \twoheadrightarrow 1))\bullet t_1).n \\
& \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & \underbrace{(t_1; n \twoheadrightarrow S' \twoheadrightarrow 1)}_{t_2}.n \\
& & \\
& \triangleq & t_2 \bullet n \bullet t_2 \\
& \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & ((add_n \twoheadrightarrow \ldots)\bullet n; (n \twoheadrightarrow S' \twoheadrightarrow 1)\bullet n)\bullet t_2 \\
& \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & \mathsf{stk}; (S' \twoheadrightarrow 1)\bullet t_2
\end{array}
$$

Back to objects

An object with "self-extension"

Let $t_1 \triangleq add_n \to S \to (S; n \to S' \to 1)$

$$
\begin{array}{lll}
(t_1.add_n).n & \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & ((S \to (S; n \to S' \to 1))\bullet t_1).n \\
& \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & \underbrace{(t_1; n \to S' \to 1)}_{t_2}.n
\end{array}
$$

$$
\begin{array}{lll}
& \triangleq & t_2 \bullet n \bullet t_2 \\
& \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & ((add_n \to \ldots)\bullet n; (n \to S' \to 1)\bullet n)\bullet t_2 \\
& \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & \mathsf{stk}; (S' \to 1)\bullet t_2 \\
& =_{\mathbb{T}} & (S' \to 1)\bullet t_2 \quad (\mathbb{T}_{\varsigma\mathcal{O}bj})
\end{array}
$$

Back to objects

An object with "self-extension"

Let $t_1 \triangleq add_n \to S \to (S; n \to S' \to 1)$

$$
\begin{array}{lll}
(t_1.add_n).n & \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & ((S \to (S; n \to S' \to 1))\bullet t_1).n \\
& \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & \underbrace{(t_1; n \to S' \to 1)}_{t_2}.n
\end{array}
$$

$$
\begin{array}{lll}
 & \triangleq & t_2 \bullet n \bullet t_2 \\
 & \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & ((add_n \to \ldots)\bullet n; (n \to S' \to 1)\bullet n)\bullet t_2 \\
 & \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & \mathsf{stk}; (S' \to 1)\bullet t_2 \\
 & =_{\mathbb{T}} & (S' \to 1)\bullet t_2 \quad (\mathbb{T}_{\varsigma \mathcal{O}bj}) \\
 & \mapsto\!\!\!\!\!\rightarrow_{\rho\sigma\delta} & 1
\end{array}
$$

Back to objects

Inheritance in the ρ-calculus (Abadi & Cardelli encoding of classes-as-objects)

$$
\begin{aligned}
PClass \triangleq\ & new \to S \to (val \to S' \to (S.preval)\bullet S';\\
& \qquad\qquad get \to S' \to (S.preget)\bullet S';\\
& \qquad\qquad set \to S' \to (S.preset)\bullet S');\\
& preval \to S \to S' \to v(1,1);\\
& preget \to S \to S' \to S'.val;\\
& preset \to S \to S' \to v(X,Y) \to (S'.val := S'' \to v
\end{aligned}
$$

Then:

$$PClass.new \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} Point$$

Back to objects

The Lambda Calculus of Objects $\lambda\mathcal{O}bj$

Abstract syntax

$$M, N \quad ::= \quad c \mid X \mid \lambda X.M \mid MN \mid$$
$$\langle\,\rangle \mid \langle M \longleftarrow n = N\rangle \mid \langle M \longleftarrow\!\!+\, n = N\rangle \mid M \Leftarrow n \mid$$
$$Sel(M, m, N)$$

Small-step semantics $(\leftrightarrow\!\!* \;=\; \longleftarrow$ or $add)$

$$(Beta) \qquad (\lambda X.M)\, N \;\rightsquigarrow\; \{X/N\}M$$
$$(Sel) \qquad M \Leftarrow m \;\rightsquigarrow\; Sel(M, m, M)$$
$$(Next) \qquad Sel(\langle M \leftrightarrow\!\!* n = N\rangle, m, P) \;\rightsquigarrow\; Sel(M, m, P)$$
$$(Succ) \qquad Sel(\langle M \leftrightarrow\!\!* n = N\rangle, n, P) \;\rightsquigarrow\; NP$$

Compiling $\lambda\mathcal{O}bj$ in ρ-calculus

$$
\begin{array}{lcl}
[\![c]\!] & \triangleq & c \\
[\![X]\!] & \triangleq & X \\
[\![\lambda X.M]\!] & \triangleq & X \rightarrowtriangle [\![M]\!] \\
[\![MN]\!] & \triangleq & [\![M]\!]\bullet[\![N]\!] \\
[\![< >]\!] & \triangleq & \mathsf{stk} \\
[\![< M \longleftarrow n = N >]\!] & \triangleq & kill_n([\![M]\!]); n \rightarrowtriangle [\![N]\!] \\
[\![< M \longleftarrow\!\!+ n = N >]\!] & \triangleq & [\![M]\!]; n \rightarrowtriangle [\![N]\!] \\
[\![M\text{''} <= \text{''}m]\!] & \triangleq & [\![M]\!].m \triangleq [\![M]\!]\bullet m\bullet[\![M]\!] \\
[\![Sel(M,m,N)]\!] & \triangleq & [\![M]\!]\bullet m\bullet[\![N]\!]
\end{array}
$$

Theorem: If $M \rightsquigarrow_{\lambda\mathcal{O}bj} N$, then $[\![M]\!] \mapsto\!\!\!\!\twoheadrightarrow_{\rho\partial\mathbb{S}_{\mathbb{T}_{\lambda\mathcal{O}bj}}} [\![N]\!]$.

The Object Calculus $\varsigma\mathcal{O}bj$

Abstract syntax

$$a, b \quad ::= \quad X \;\mid\; [m_i = \varsigma(X)b_i]^{i\in I} \;\mid\; a.m \;\mid\; a.m := \varsigma(X)b$$

Small-step semantics

Let $a \triangleq [m_i = \varsigma(X)b_i]^{i\in I}$

$$(Select) \qquad a.m_j \rightsquigarrow \{X/a\}b_j$$

$$(Update) \qquad a.m_j := \varsigma(X)b \rightsquigarrow [m_i = \varsigma(X)b_i, m_j = \varsigma(X)b]^{i\in I\setminus\{}$$

$$(Extend) \qquad a.m_j := \varsigma(X)b \rightsquigarrow [m_i = \varsigma(X)b_i, m_j = \varsigma(X)b]^{i\in I}$$

Compiling $\varsigma\mathcal{O}bj$ in ρ-calculus

$$
\begin{array}{lcl}
[\![X]\!] & \triangleq & X \\
[\![a.m_j]\!] & \triangleq & [\![a]\!].m_j \\
[\![\ [m_i = \varsigma(X)b_i]^{i\in I}\ ]\!] & \triangleq & (m_i \twoheadrightarrow X \twoheadrightarrow [\![b_i]\!])^{i\in I} \\
[\![a.m := \varsigma(X)b]\!] & \triangleq & [\![a]\!].m := X \twoheadrightarrow [\![b]\!]
\end{array}
$$

Theorem:

If $a \rightsquigarrow_{\varsigma\mathcal{O}bj} b$, then $[\![a]\!] \longmapsto\!\!\!\!\!\rightarrow_{\rho\partial\mathcal{S}_{\mathbb{T}_{\varsigma\mathcal{O}bj}}} [\![b]\!]$.

... back to other examples

The Para object: labels as first-class entities

$$Para \triangleq (ten \twoheadrightarrow S \twoheadrightarrow 10; par(X) \twoheadrightarrow S \twoheadrightarrow S.X)$$

This object has a method $par(X)$ which seeks for a method name that is assigned to the variable X and then sends this method to the object itself.

... back to other examples

The Para object: labels as first-class entities

$$Para \triangleq (ten \twoheadrightarrow S \twoheadrightarrow 10; par(X) \twoheadrightarrow S \twoheadrightarrow S.X)$$

This object has a method $par(X)$ which seeks for a method name that is assigned to the variable X and then sends this method to the object itself.

$$\begin{array}{lll} Para.(par(ten)) & \triangleq & Para\bullet(par(ten))\bullet Para \\ & \mapsto_{\rho\alpha\delta} & ((ten \twoheadrightarrow S \twoheadrightarrow 10)\bullet(par(ten)); \\ & & (par(X) \twoheadrightarrow S \twoheadrightarrow S.X)\bullet(par(ten)))\bullet Pa \\ & \mapsto_{\rho\alpha\delta} & \mathsf{stk}; (S \twoheadrightarrow S.ten)\bullet Para \\ & =_{\mathbb{T}} & (S \twoheadrightarrow S.ten)\bullet Para \quad (\mathbb{T}_{\varsigma\mathcal{O}bj}) \\ & \mapsto_{\rho\alpha\delta} & Para.ten \\ & \mapsto_{\rho\alpha\delta} & 10 \end{array}$$

... back to other examples

The object Daemon: methods as first-class entities

$$Daemon \triangleq set \rightarrowtriangle S \rightarrowtriangle X \rightarrowtriangle (X; set \rightarrowtriangle S' \rightarrowtriangle Y \rightarrowtriangle (Y; S'))$$

... back to other examples

The object Daemon: methods as first-class entities

$$Daemon \triangleq set \to S \to X \to (X; set \to S' \to Y \to (Y; S'))$$

$$Daemon.set(x \to S \to 3)$$

$$\begin{array}{ll} \triangleq & Daemon{\bullet}set{\bullet}Daemon{\bullet}(x \to S \to 3) \\ \mapsto\!\!\!\!\!\rightarrow_{\rho\varpi\delta} & (S \to X \to (X; set \to S' \to Y \to (Y; S')))\bullet Daemon\bullet(x - \\ \mapsto\!\!\!\!\!\rightarrow_{\rho\varpi\delta} & (X \to (X; set \to S' \to Y \to (Y; S')))\bullet(x \to S \to 3) \\ \mapsto\!\!\!\!\!\rightarrow_{\rho\varpi\delta} & \underbrace{x \to S \to 3; set \to S' \to Y \to (Y; S')}_{obj} \end{array}$$

... back to other examples

The object Daemon: methods as first-class entities

$$Daemon \triangleq set \to S \to X \to (X; set \to S' \to Y \to (Y; S'))$$

$$Daemon.set(x \to S \to 3)$$

$$\begin{array}{ll} \triangleq & Daemon{\bullet}set{\bullet}Daemon{\bullet}(x \to S \to 3) \\ \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} & (S \to X \to (X; set \to S' \to Y \to (Y; S')))\bullet Daemon\bullet(x - \\ \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} & (X \to (X; set \to S' \to Y \to (Y; S')))\bullet(x \to S \to 3) \\ \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} & \underbrace{x \to S \to 3; set \to S' \to Y \to (Y; S')}_{obj} \end{array}$$

$$obj.set(y \to S \to 4) \mapsto\!\!\!\!\!\rightarrow_{\rho\delta} (y \to S \to 4; x \to S \to 3; set \to S' \to Y \to (Y; S'))$$

... back to other examples

Overview

- **Integration of first-order rewriting and higher-order theories**
  - ➥ *rewrite rules* considered as *first-class* objects
  - ➥ explicit concept of *rewrite rule application*
  - ➥ *matching theories* as the basic parameter passing mechanism
  - ➥ sophisticated type systems

- **Significant expressive power**
  - ➥ first order rewriting (with (non-deterministic) strategies) ⇨ rewrite based languages semantics (ELAN, Maude, OBJ,. . .)
  - ➥ lambda calculus
  - ➥ object calculi: *Object Calculus* and *Lambda Calculus of Objects*

"Untyped" Matching theories I

Empty theory $\mathbb{T}_\emptyset$ of equality (up to α-conversion)

$$\frac{\Vdash T_1 = T_2 \quad \Vdash T_2 = T_3}{\Vdash T_1 = T_3}\ (Trans) \qquad \frac{\Vdash T_1 = T_2}{\Vdash T_2 = T_1}\ (Syr$$

$$\frac{\Vdash T_1 = T_2}{\Vdash T_{3\lceil T_1 \rceil_p} = T_{3\lceil T_2 \rceil_p}}\ (Cont) \qquad \frac{}{\Vdash t = t}\ (Refl)$$

$T_{1\lceil T_2 \rceil_p}$: term T_1 with term T_2 at position p BACK

"Untyped" Matching theories II

Theory of Associativity $\mathbb{T}_{A(f)}$ (resp. Commutativity $\mathbb{T}_{C(f)}$) is defined as $\mathbb{T}_{\emptyset}$ plus:

$$\frac{}{\Vdash f(f(T_1, T_2), T_3) = f(T_1, f(T_2, T_3))} \ (Assoc)$$

$$\frac{}{\Vdash f(T_1, T_2) = f(T_2, T_1)} \ (Comm)$$

BACK

"Untyped" Matching theories III

Theory of Idempotency $\mathbb{T}_{I(f)}$ is defined as $\mathbb{T}_{\emptyset}$ plus the axiom

$$\frac{}{\Vdash f(T,T) = t}\ (Idem)$$

Theory of Neutral Element $\mathbb{T}_{N(f^0)}$ is defined as $\mathbb{T}_{\emptyset}$ plus

$$\frac{}{\Vdash f(0,T) = T}\ (0{-}Left) \qquad \frac{}{\Vdash f(T,0) = T}\ (0{-}Right)$$

BACK

"Untyped" Matching theories IV

The Theory of Stuck $\mathbb{T}_{\mathsf{stk}}$ is defined as $\mathbb{T}_{N(,^{\mathsf{stk}})}$ plus the axioms

$$\frac{\forall \theta_1, \theta_2,\ \forall C,\ A\theta_2 \mapsto\!\!\!\!\!\rightarrow_{\rho\alpha\delta} C \ \Rightarrow\ C \not\equiv P\theta_1}{\Vdash [P \ll A]B = \mathsf{stk}}$$

$$\frac{}{\Vdash \mathsf{stk}\bullet T = \mathsf{stk}}$$

Examples

$\Vdash [3 \ll 4]5 = \mathsf{stk}$

$\not\Vdash [3 \ll 3]5 = \mathsf{stk}$

$\not\Vdash [3 \ll \mathcal{X}]5 = \mathsf{stk}$

BACK

"Untyped" Matching theories V

Theory of the Lambda Calculus of Objects $\mathbb{T}_{\lambda\mathcal{O}bj}$ is obtained by considering the symbol ";" as associative and stk as its neutral element, $i.e.$:

$$\mathbb{T}_{\lambda\mathcal{O}bj} = \mathbb{T}_{A(;)} \cup \mathbb{T}_{\mathsf{stk}}$$

Theory of the Object Calculus $\mathbb{T}_{\varsigma\mathcal{O}bj}$ is obtained by considering the symbol ";" as associative and commutative and stk as its neutral element, $i.e.$:

$$\mathbb{T}_{\varsigma\mathcal{O}bj} = \mathbb{T}_{A(;)} \cup \mathbb{T}_{C(;)} \cup \mathbb{T}_{\mathsf{stk}} = \mathbb{T}_{\lambda\mathcal{O}bj} \cup \mathbb{T}_{C(;)}$$

THEORIES RECORDS

The Matching Algorithm for $\mathbb{T}_\emptyset$

The matching substitution solving a matching equation can be computed by the following *matching reduction system*:

$$(Appl) \qquad (T_1 \bullet T_2) \ll (T_3 \bullet T_4) \rightsquigarrow T_1 \ll T_3 \wedge T_2 \ll T_4$$

$$(Struct) \quad (T_1 ; T_2) \ll (T_3 ; T_4) \rightsquigarrow T_1 \ll T_3 \wedge T_2 \ll T_4$$

Example

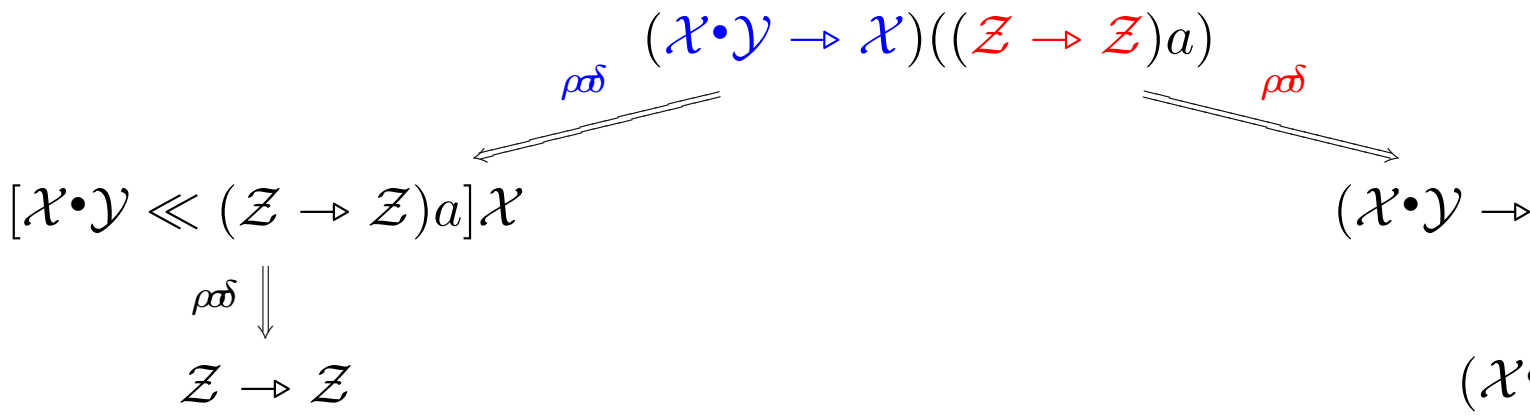
$f(\mathcal{X}, \mathcal{Y}) \ll f(a, b) \rightsquigarrow \mathcal{X} \ll a \wedge \mathcal{Y} \ll b$ successful

$f(\mathcal{X}, \mathcal{X}) \ll f(a, b) \rightsquigarrow \mathcal{X} \ll a \wedge \mathcal{Y} \ll b$ unsuccessful

BACK

On the (non-)confluence

Variables in applicative position



Rigid Pattern Condition (RPC) [van Oostrom 90]

$\mathcal{P} \triangleq \{T \in NF(\rho\sigma\delta) \mid T$ is "linear" with no "active" variables$\}$

BACK

On the (non-)confluence