Some considerations on the rewriting calculus

Mathematics is frequently described as "the science of pattern," a characterisation that makes more sense than most, both of pure mathematics, but also of the ability of mathematics to connect to the world teeming with patterns, symmetries, regularities, and uniformities

Jon Barwise
Lawrence Moss

What is rewriting? (1/2)

1- Discriminate

to give the possibility to discriminate directly

What is rewriting? (1/2)

1- Discriminate

to give the possibility to discriminate directly

What is rewriting? (1/2)

1- Discriminate

to give the possibility to discriminate directly

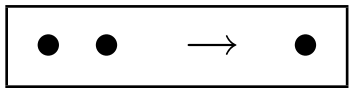
lambda-calculus is not discriminating: one needs to encode matching

What is rewriting? (2/2)

1- Discriminate
2- Transform what has been discriminated

What is rewriting? (2/2)

1- Discriminate
2- Transform what has been discriminated

-  discriminates the repetition of two black bullets

What is rewriting? (2/2)

1- Discriminate
2- Transform what has been discriminated

- $\boxed{\bullet\ \bullet \quad \longrightarrow \quad \bullet}$ discriminates the repetition of two black bullets

  $\bullet\ \circ\ \bullet\ \bullet\ \circ\ \circ\ \bullet\ \bullet$ is rewritten, for example, into $\bullet\ \circ\ \bullet\ \circ\ \circ\ \bullet\ \bullet$

- $\boxed{x \times 0 \quad \longrightarrow \quad 0}$ discriminates objects where x is arbitrary

  3×0 is rewritten into 0

What is rewriting? (2/2)

1- Discriminate
2- Transform what has been discriminated

- $\boxed{\bullet\ \bullet \quad \longrightarrow \quad \bullet}$ discriminates the repetition of two black bullets

  $\bullet\ \circ\ \bullet\ \bullet\ \circ\ \circ\ \bullet\ \bullet$ is rewritten, for example, into $\bullet\ \circ\ \bullet\ \circ\ \circ\ \bullet\ \bullet$

- $\boxed{x \times 0 \quad \longrightarrow \quad 0}$ discriminates objects where x is arbitrary

  3×0 is rewritten into 0

- $\boxed{x \text{ et } x \quad \longrightarrow \quad x}$ discriminates objects where x is repeated

  $x = 3$ et $x = 3$ is rewritten into en $x = 3$

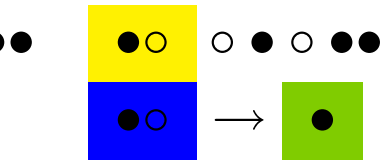
What are the basic operations in rewriting?

What are the basic operations in rewriting?

1– Matching

The data:

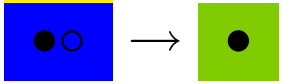
The rewrite rule:



What are the basic operations in rewriting?

1– Matching

The data: 

The rewrite rule: 

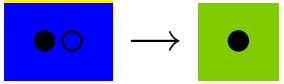
2– Compute what should be substituted

The lefthand side: 

What are the basic operations in rewriting?

1– Matching

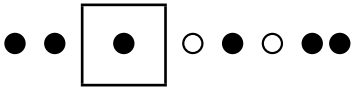
The data: 

The rewrite rule: 

2– Compute what should be substituted

The lefthand side: 

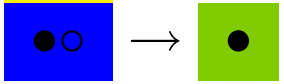
3– Replacement

The new generated data: 

What are the basic operations in rewriting?

1– Matching

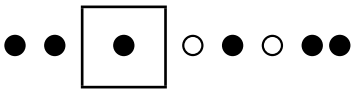
The data: 

The rewrite rule: 

2– Compute what should be substituted

The lefthand side: 

3– Replacement

The new generated data: 

(4)– Strategy

Chose the appropriate rule

Find an appropriate occurrence

Chose the appropriate result

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

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

Lambda-calculus is great, but

- lacks of discrimination capabilities
- difficult to control

A "simple" λ-term...

$$(\lambda Y.\Big((\lambda y.(yx_{\perp}(\lambda X.X)))Y\Big))\Big((\lambda x_1.\lambda z_1\lambda z_2.(z_2x_1))(\lambda u_1\lambda u_2.u_1)\Big)$$

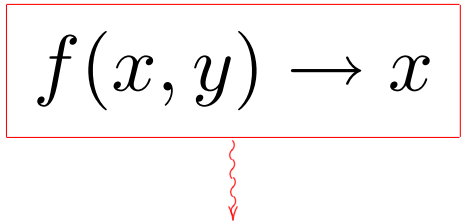
.... and its meaning

$$\overbrace{\llbracket\, \lambda Y.(\overbrace{f(X) \twoheadrightarrow X}^{\llbracket\, f(X) \twoheadrightarrow X\,\rrbracket})\bullet Y\,\rrbracket}^{(\lambda Y.(\overbrace{(\lambda y.(y x_\perp (\lambda X.X)))}^{\llbracket f\rrbracket} Y))} \overbrace{\llbracket\, f(a)\,\rrbracket}^{\left(\overbrace{(\lambda x_1.\lambda z_1 \lambda z_2.(z_2 x_1))}^{\llbracket f\rrbracket}\overbrace{(\lambda u_1 \lambda u_2.u_1)}^{\llbracket a\rrbracket}\right)}$$

Simple encoding of rewriting in the λ-calculus

$$
\begin{array}{ll}
 & \overbrace{\overbrace{[\![\, \lambda Y.(f(X) \twoheadrightarrow X)\bullet Y \,]\!]}^{[\![\, f(X) \twoheadrightarrow X \,]\!]}}\overbrace{[\![\, f(a) \,]\!]}^{[\![\, f \,]\!]\ [\![\, a \,]\!]} \\
 & (\lambda Y.\Big(\overbrace{(\underline{\lambda y}.(y x_\perp(\lambda X.X)))}^{[\![\, f(X) \twoheadrightarrow X \,]\!]}\underline{Y}\Big))\Big(\overbrace{(\lambda x_1.\lambda z_1\lambda z_2.(z_2 x_1))}^{[\![\, f \,]\!]}\overbrace{(\lambda u_1\lambda u_2.u_1)}^{[\![\, a \,]\!]}\Big) \\
\mapsto_\beta & \big(\lambda Y.(Y x_\perp(\lambda X.X))\big)\Big((\underline{\lambda x_1}.\lambda z_1\lambda z_2.(z_2 x_1))\underline{(\lambda u_1\lambda u_2.u_1)}\Big) \\
\mapsto_\beta & \big(\underline{\lambda Y}.(Y x_\perp(\lambda X.X))\big)\underline{\big(\lambda z_1\lambda z_2.(z_2(\lambda u_1\lambda u_2.u_1))\big)} \\
\mapsto_\beta & \big(\underline{\lambda z_1}\lambda z_2.(z_2(\lambda u_1\lambda u_2.u_1))\big)\underline{x_\perp}(\lambda X.X) \\
\mapsto_\beta & \big(\underline{\lambda z_2}.(z_2(\lambda u_1\lambda u_2.u_1))\big)\underline{(\lambda X.X)} \\
\mapsto_\beta & (\underline{\lambda X}.X)\underline{(\lambda u_1\lambda u_2.u_1)} \\
\mapsto_\beta & (\lambda u_1\lambda u_2.u_1) \\
= & [\![\, a \,]\!]
\end{array}
$$

Term rewriting

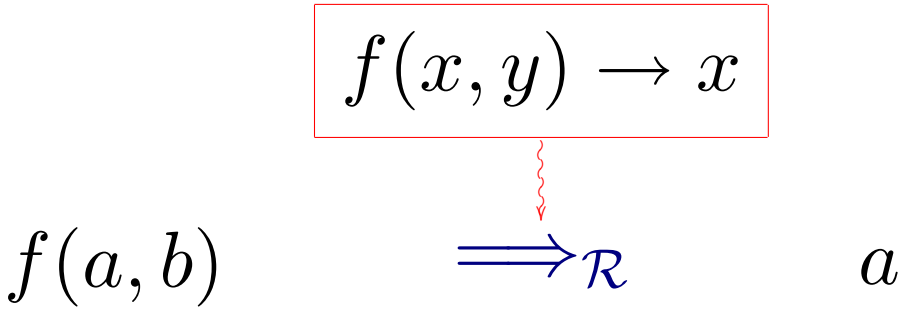


Term rewriting

$$f(x, y) \longrightarrow x$$

$$f(a, b)$$

Term rewriting



Rewriting calculus - abstraction

$$f(X, Y) \quad \twoheadrightarrow \quad X$$

$\hookrightarrow$ Abstraction Operator

Rewriting calculus - application

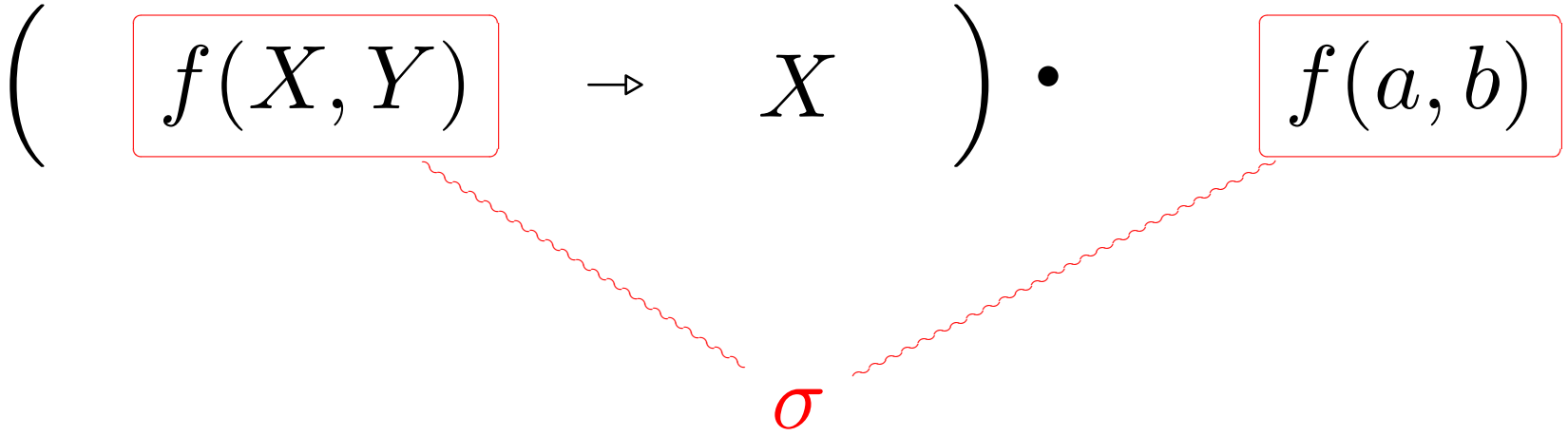
$$f(X,Y) \quad \twoheadrightarrow \quad X \qquad f(a,b)$$

Rewriting calculus - application

$$\left(f(X,Y) \quad \twoheadrightarrow \quad X \right) \bullet f(a,b)$$

➥ Application Operator

Rewriting calculus - compute the substitution



Rewriting calculus - replacement

$$\left(\quad f(X,Y) \quad \twoheadrightarrow \quad X \quad \right) \bullet \quad f(a,b)$$

$$\sigma$$

$$= \{X \mapsto a, Y \mapsto b\}$$

Rewriting calculus - replacement

$$\Big(\quad f(X,Y) \quad \twoheadrightarrow \quad X \quad \Big) \bullet \quad f(a,b)$$

σ

$$= \{X \mapsto a, Y \mapsto b\}$$

Rewriting calculus - result

$$\left(\quad f(X,Y) \quad \rightarrow \quad X \quad \right) \bullet \quad f(a,b)$$

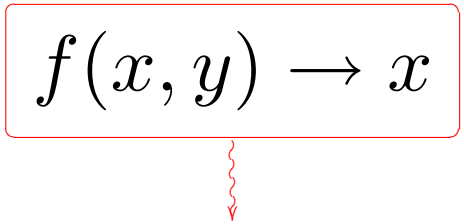
$$\longmapsto \quad \sigma(X)$$

Rewriting calculus - result

$$\left(\quad f(X,Y) \quad \twoheadrightarrow \quad X \quad \right) \bullet \quad f(a,b)$$

$\longmapsto \quad \sigma(X)$ *i.e.* a

For the rewriting relation

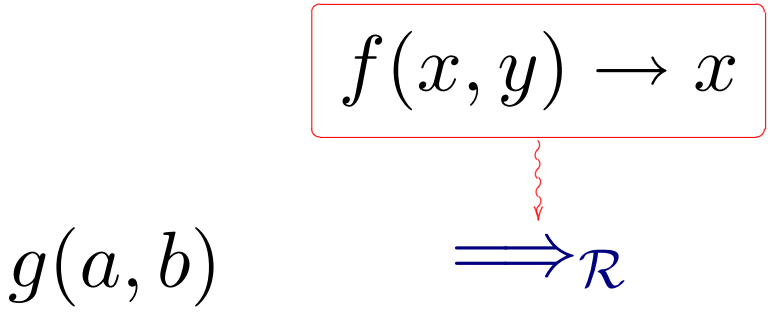


For the rewriting relation

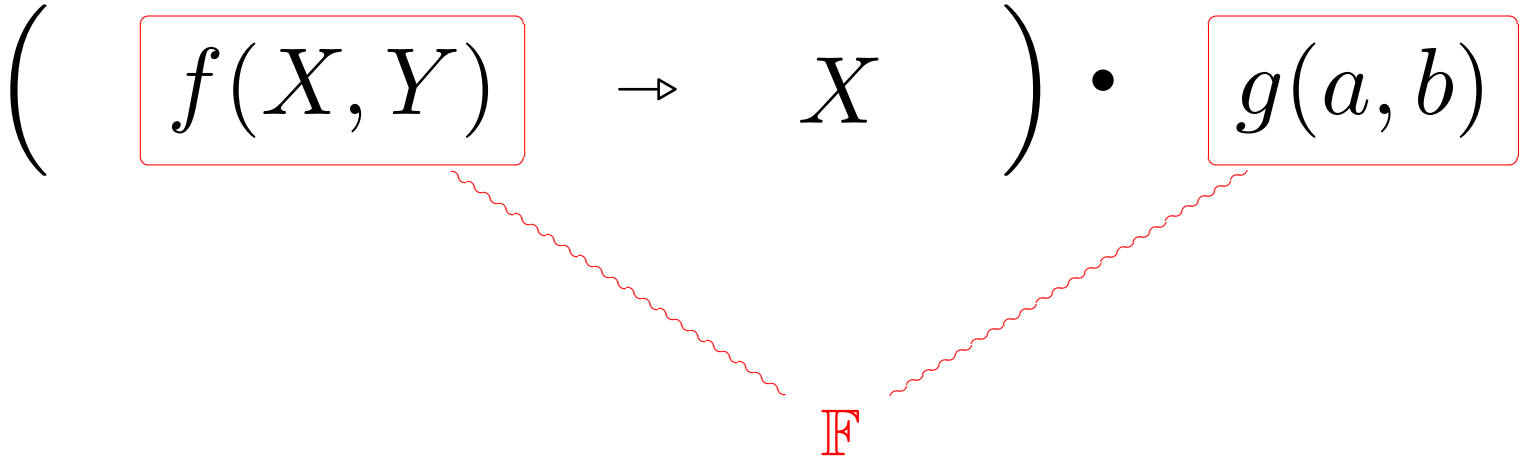
$$f(x, y) \longrightarrow x$$

$$g(a, b)$$

For the rewriting relation



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

The Simplest Reduction Semantics

$$(P \twoheadrightarrow A)\bullet B \;\rightarrow_{\rho} A\theta_{(P \lessdot B)} \qquad \text{if } \exists\theta.\ P\theta =_{\mathbb{T}} B$$

Non unitary matching

$$\left(f_{AC}(X, Y) \twoheadrightarrow X \right) \bullet f_{AC}(a, b)$$

Non unitary matching

$$\left(f_{AC}(X, Y) \twoheadrightarrow X \right) \bullet f_{AC}(a, b)$$

$$\overset{?}{\longmapsto} \quad a$$

Non unitary matching

$$\left(f_{AC}(X, Y) \twoheadrightarrow X \right) \bullet f_{AC}(a, b)$$

$\overset{?}{\longmapsto}\ a$

$\overset{?}{\longmapsto}\ b$

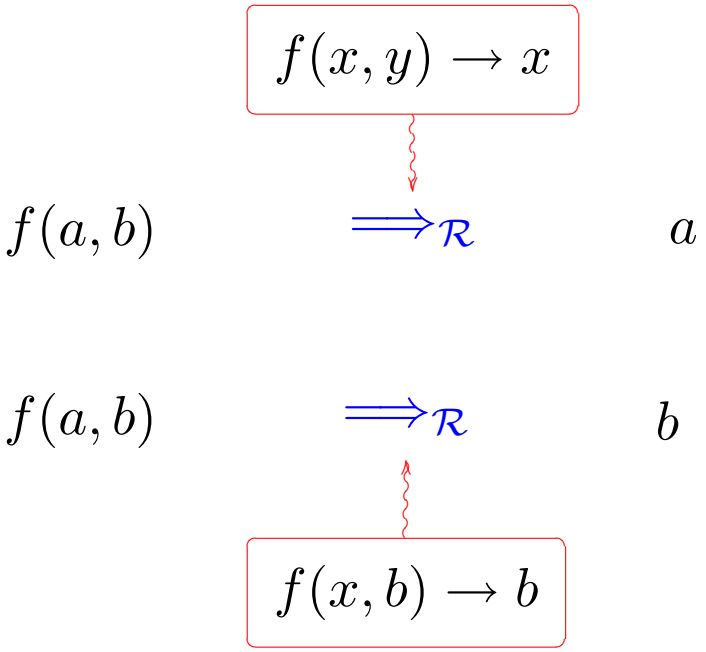
Basic ρ-calculus

$$(P \twoheadrightarrow A)\bullet B \;\rightarrow_\rho\; A\theta_1;\ldots;A\theta_n,\ldots$$

with $\{\theta_1,\ldots,\theta_n,\ldots\} = \mathcal{S}ol(P \ll_{\mathbb{T}} B)$

Going from rewrite rule to rewriting system:
For the rewriting reduction

$$\begin{cases} f(x,y) \to x \\ f(x,b) \to b \end{cases}$$



Non Determinism

Structure ρ-calculus

$$(P \rightarrowtriangle A)\bullet B \;\rightarrow_{\rho}\; A\theta_1; \ldots; A\theta_n, \ldots$$

with $\{\theta_1, \ldots, \theta_n, \ldots\} = \mathcal{S}ol(P \lessdot_{\mathbb{T}} B)$

$$(A; B)\bullet C \;\rightarrow_{\partial}\; A\bullet C; B\bullet C$$

Going from rewrite rule to rewriting system: In the ρ-calculus

$f(X,Y) \twoheadrightarrow X \qquad f(X,b) \twoheadrightarrow b$

Going from rewrite rule to rewriting system: In the ρ-calculus

$f(X,Y) \twoheadrightarrow X; \qquad f(X,b) \twoheadrightarrow b$

Going from rewrite rule to rewriting system: In the ρ-calculus

$f(X,Y) \twoheadrightarrow X;$ $\quad f(X,b) \twoheadrightarrow b$ $\quad f(a,b)$

Going from rewrite rule to rewriting system: In the ρ-calculus

$$\left(f(X,Y) \twoheadrightarrow X; \qquad f(X,b) \twoheadrightarrow b\right) \qquad \bullet f(a,b)$$

Going from rewrite rule to rewriting system: In the ρ-calculus

$$\Big(f(X,Y) \twoheadrightarrow X; \qquad f(X,b) \twoheadrightarrow b\Big) \qquad \bullet f(a,b)$$

$$\rightarrow \quad (f(X,Y) \twoheadrightarrow X)\bullet f(a,b); (f(X,b) \twoheadrightarrow b)\bullet f(a,b)$$

Going from rewrite rule to rewriting system: In the ρ-calculus

$$\Big(f(X,Y) \twoheadrightarrow X; \qquad f(X,b) \twoheadrightarrow b\Big) \qquad \bullet f(a,b)$$

$\rightarrow$ $(f(X,Y) \twoheadrightarrow X)\bullet f(a,b); (f(X,b) \twoheadrightarrow b)\bullet f(a,b)$

$\rightarrow$ $a; b$

Failures

$f(X, Y) \rightarrowtriangle X \qquad f(X, c) \rightarrowtriangle c$

Failures

$f(X, Y) \twoheadrightarrow X; \qquad f(X, c) \twoheadrightarrow c$

Failures

$f(X, Y) \twoheadrightarrow X;$ $\quad f(X, c) \twoheadrightarrow c$ $\quad f(a, b)$

Failures

$$\Big(f(X,Y) \twoheadrightarrow X; \qquad f(X,c) \twoheadrightarrow c\Big) \qquad \bullet f(a,b)$$

Failures

$$\Big(f(X,Y) \twoheadrightarrow X; \qquad f(X,c) \twoheadrightarrow c\Big) \qquad \bullet f(a,b)$$

$$\rightarrow \quad (f(X,Y) \twoheadrightarrow X)\bullet f(a,b); (f(X,c) \twoheadrightarrow c)\bullet f(a,b)$$

Failures

$$\Big(f(X,Y) \twoheadrightarrow X; \qquad f(X,c) \twoheadrightarrow c\Big) \qquad \bullet f(a,b)$$

$\rightsquigarrow$ $(f(X,Y) \twoheadrightarrow X)\bullet f(a,b); (f(X,c) \twoheadrightarrow c)\bullet f(a,b)$

$\rightsquigarrow$ $a; (f(X,c) \twoheadrightarrow c)\bullet f(a,b)$

Detecting matching failures: the symbol stk

1. The relation $P \not\sqsubseteq A$ detects (some) definitive matching failures:

   $\forall \sigma$, $\forall B$ s.t. $\sigma(A) \mapsto B$, $\nexists \tau$ s.t. $\tau(P) = B$

2. The relation $\rightarrow_{\mathsf{stk}}$ treats matching failures uniformly:

$$
\begin{array}{rcll}
(P \rightarrowtriangle B)\bullet A & \rightarrow_{\mathsf{stk}} & \mathsf{stk} & \text{if } P \not\sqsubseteq A \\
\mathsf{stk}; A & \rightarrow_{\mathsf{stk}} & A & \\
A; \mathsf{stk} & \rightarrow_{\mathsf{stk}} & A & \\
\mathsf{stk}\bullet A & \rightarrow_{\mathsf{stk}} & \mathsf{stk} &
\end{array}
$$

Failures

$f(X, Y) \twoheadrightarrow X \qquad f(X, c) \twoheadrightarrow c$

Failures

$f(X, Y) \twoheadrightarrow X; \qquad f(X, c) \twoheadrightarrow c$

Failures

$f(X, Y) \twoheadrightarrow X;$ $\qquad f(X, c) \twoheadrightarrow c$ $\qquad f(a, b)$

Failures

$$\Big(f(X,Y) \twoheadrightarrow X; \qquad f(X,c) \twoheadrightarrow c\Big) \qquad \bullet f(a,b)$$

Failures

$$\Big(f(X,Y) \twoheadrightarrow X; \qquad f(X,c) \twoheadrightarrow c\Big) \qquad \bullet f(a,b)$$

$$\rightarrow \quad (f(X,Y) \twoheadrightarrow X)\bullet f(a,b); (f(X,c) \twoheadrightarrow c)\bullet f(a,b)$$

Failures

$$\Big(f(X,Y) \twoheadrightarrow X; \qquad f(X,c) \twoheadrightarrow c\Big) \qquad \bullet f(a,b)$$

➥ $(f(X,Y) \twoheadrightarrow X)\bullet f(a,b); (f(X,c) \twoheadrightarrow c)\bullet f(a,b)$

➥ a; stk

➥ a

(Well-typed) Encoding of Rewriting in the ρ-calculus

- rewrite rules and their application,
  - ρ-abstractions and applications (Simple Encoding)

(Well-typed) Encoding of Rewriting in the ρ-calculus

- rewrite rules and their application,
  - ρ-abstractions and applications (Simple Encoding)

- an iteration operator that applies $repeatedly$ a set of rewrite rules,
  - $\omega \bullet (f \bullet \omega)$

(Well-typed) Encoding of Rewriting in the ρ-calculus

- rewrite rules and their application,
  - ρ-abstractions and applications (Simple Encoding)

- an iteration operator that applies $repeatedly$ a set of rewrite rules,
  - $\omega \bullet (f \bullet \omega)$

- a construction grouping together a set of rewrite rules,
  - structures and objects

(Well-typed) Encoding of Rewriting in the ρ-calculus

- rewrite rules and their application,
  - ρ-abstractions and applications (Simple Encoding)

- an iteration operator that applies $repeatedly$ a set of rewrite rules,
  - $\omega \bullet (f \bullet \omega)$

- a construction grouping together a set of rewrite rules,
  - structures and objects

- an operator testing if a set of rewrite rules is $applicable$ to a term.
  - the symbol stk

Encoding rewriting - Addition over Peano integers

$$plus \triangleq \left[\begin{array}{l} S \rightarrowtriangle add(0, y) \rightarrowtriangle y; \\ S \rightarrowtriangle add(suc(x), y) \rightarrowtriangle suc\big((S \bullet S) \bullet add(x, y)\big) \end{array} \right]$$

Encoding rewriting - Addition over Peano integers

$$plus \triangleq \left[\begin{array}{l} S \rightarrowtriangle add(0, y) \rightarrowtriangle y; \\ S \rightarrowtriangle add(suc(x), y) \rightarrowtriangle suc\big((S \bullet S) \bullet add(x, y)\big) \end{array} \right]$$

$$(plus \bullet plus) \bullet add(N, M) \quad \overset{}{\underset{\beta\delta\cup\mathsf{stk}}{\longmapsto\!\!\!\!\!\rightarrow}} \quad M + N$$

Encoding rewriting - Addition over Peano integers

$$plus \triangleq \left[\begin{array}{l} S \rightarrowtriangle add(0, y) \rightarrowtriangle y; \\ S \rightarrowtriangle add(suc(x), y) \rightarrowtriangle suc\big((S \bullet S) \bullet add(x, y)\big) \end{array} \right]$$

$$(plus \bullet plus) \bullet add(N, M) \quad \mapsto\!\!\!\!\!\!\twoheadrightarrow_{\mathcal{S} \cup \mathsf{stk}} \quad M + N$$

Fill in the blanks with your favorite rewrite system...

$$func \triangleq \left[\begin{array}{lll} S \rightarrowtriangle & & ; \\ S \rightarrowtriangle & & (S \bullet S) \bullet \end{array} \right]$$

Encoding rewriting - Addition over Peano integers

$$plus \triangleq \left[\begin{array}{l} S \rightarrowtriangle add(0, y) \rightarrowtriangle y; \\ S \rightarrowtriangle add(suc(x), y) \rightarrowtriangle suc\big((S \bullet S) \bullet add(x, y)\big) \end{array} \right]$$

$$(plus \bullet plus) \bullet add(N, M) \quad \mapsto\!\!\!\!\!\!\twoheadrightarrow_{\beta\mathcal{S}\cup \mathsf{stk}} \quad M + N$$

Fill in the blanks with your favorite rewrite system...

$$func \triangleq \left[\begin{array}{l} S \rightarrowtriangle len([\,]) \rightarrowtriangle 0 \;\; ; \\ S \rightarrowtriangle len(Cons(x, l)) \rightarrowtriangle suc\big((S \bullet S) \bullet len(l)\big) \end{array} \right]$$

Encoding rewriting - Addition over Peano integers

$$plus \triangleq \left[\begin{array}{l} S \rightarrowtriangle add(0, y) \rightarrowtriangle y; \\ S \rightarrowtriangle add(suc(x), y) \rightarrowtriangle suc\big((S \bullet S) \bullet add(x, y)\big) \end{array} \right]$$

$$(plus \bullet plus) \bullet add(N, M) \quad \mapsto\!\!\!\!\!\!\twoheadrightarrow_{\beta\mathcal{S}\cup \mathsf{stk}} \quad M + N$$

Fill in the blanks with your favorite rewrite system... provided it is convergent and ground reducible if you want completeness.

$$func \triangleq \left[\begin{array}{l} S \rightarrowtriangle len([\,]) \rightarrowtriangle 0 \;\; ; \\ S \rightarrowtriangle len(Cons(x, l)) \rightarrowtriangle suc\big((S \bullet S) \bullet len(l)\big) \end{array} \right]$$

Encoding rewriting - Addition over Peano integers

$$plus \triangleq \left[\begin{array}{l} S \rightarrowtriangle add(0, y) \rightarrowtriangle y; \\ S \rightarrowtriangle add(suc(x), y) \rightarrowtriangle suc\big((S \bullet S) \bullet add(x, y)\big) \end{array} \right]$$

$$(plus \bullet plus) \bullet add(N, M) \quad \mapsto\!\!\!\!\!\!\twoheadrightarrow_{\beta\mathcal{S}\cup \mathsf{stk}} \quad M + N$$

Fill in the blanks with your favorite rewrite system... provided it is convergent and ground reducible if you want completeness.

$$func \triangleq \left[\begin{array}{l} S \rightarrowtriangle len([\,]) \rightarrowtriangle 0 \;\; ; \\ S \rightarrowtriangle len(Cons(x, l)) \rightarrowtriangle suc\big((S \bullet S) \bullet len(l)\big) \end{array} \right]$$

A first step towards an explicit calculus

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

$$[P \ll B]A \;\to_\sigma\; A\theta_1; \ldots; A\theta_n, \ldots$$
$$\text{with } \{\theta_1, \ldots, \theta_n, \ldots\} = \mathcal{S}ol(P \lessdot_{\mathbb{T}} B)$$

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

Explicit matching - $\rho_{\mathtt{m}}(1)$

Terms	M, N, P	$::=$	X	(Variables)
		$\mid$	c	(Constants)
		$\mid$	$P \twoheadrightarrow M$	(Abstraction)
		$\mid$	$M \bullet N$	(Functional application)
		$\mid$	$M; N$	(Structure)
		$\mid$	$[\mathcal{C}]\, N$	(Constraint application)
Constraints	$\mathcal{C}, \mathcal{D}$	$::=$	$\mathtt{id}$	(Identity)
		$\mid$	$P \ll M$	(Match-equation)
		$\mid$	$\mathcal{C} \wedge \mathcal{D}$	(Conjunction of constraints)

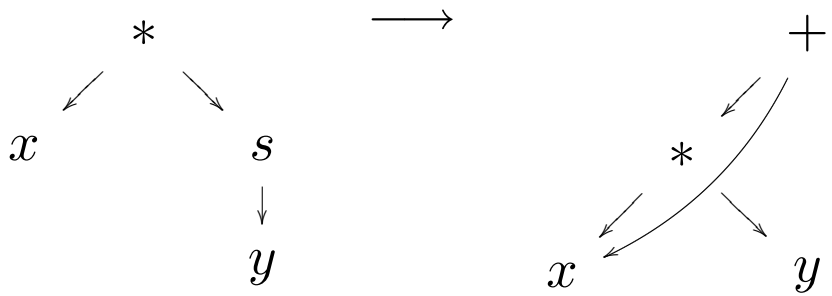
where $\wedge$ is associative and $\mathtt{id}$ is a neutral element

Explicit matching - $\rho_{\mathtt{m}}(2)$

(ρ)	$(P \twoheadrightarrow M)\bullet N$	$\rightarrow$	$[P \ll N]M$
(δ)	$(M_1; M_2)\bullet N$	$\rightarrow$	$M_1\bullet N; M_2\bullet N$
(Dec)	$M_1; M_2 \ll N_1; N_2$	$\rightarrow$	$M_1 \ll N_1 \wedge M_2 \ll N_2$
(Dec)	$f(M_1, \ldots, M_n) \ll f(N_1, \ldots, N_n)$	$\rightarrow$	$\bigwedge_{i=1}^{n} (M_i \ll N_i)$
$(Idem)$	$\mathcal{C} \wedge (X \ll M) \wedge \mathcal{D} \wedge (X \ll M) \wedge \mathcal{E}$	$\rightarrow$	$\mathcal{C} \wedge (X \ll M) \wedge \mathcal{D} \wedge \mathcal{E}$
$(Subst)$	$[\mathcal{C} \wedge (X \ll M) \wedge \mathcal{D}]\, N$	$\rightarrow$	$[\mathcal{C} \wedge \mathcal{D}]\, (\{M/X\}N)$ if $X \notin \mathcal{D}om(\mathcal{C} \wedge \mathcal{D})$
(Id)	$[\mathtt{id}]\, M$	$\rightarrow$	M

Graphs in the ρ-calculus

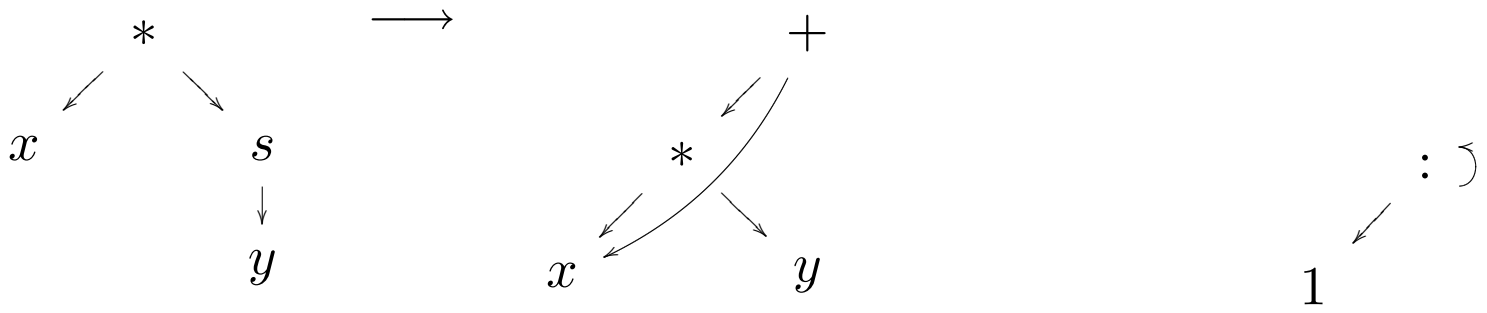
- improve efficiency
  - save space (sharing subterms)
  - save time (reduce only once)



Graphs in the ρ-calculus

- improve efficiency
  - save space (sharing subterms)
  - save time (reduce only once)

- improve expressiveness
  - infinite data structures

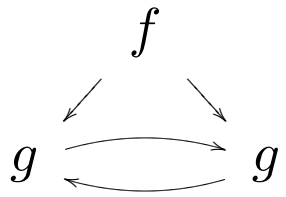


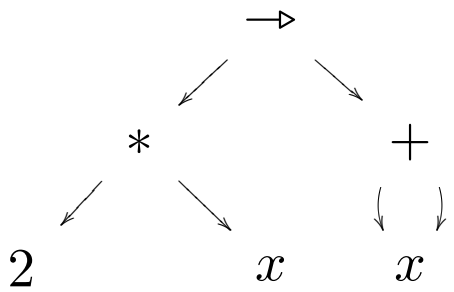
Building graphs - ρ_{g}

Terms	M, N, P	$::=$	X	(Variables)
		$\mid$	c	(Constants)
		$\mid$	$P \twoheadrightarrow M$	(Abstraction)
		$\mid$	$M \bullet N$	(Functional application)
		$\mid$	$M; N$	(Structure)
		$\mid$	$[\mathcal{C}]\, N$	(Constraint application)
Constraints	$\mathcal{C}, \mathcal{D}$	$::=$	$\mathtt{id}$	(Identity)
		$\mid$	$P \ll M$	(Match-equation)
		$\mid$	$X = M$	(Equation)
		$\mid$	$\mathcal{C} \wedge \mathcal{D}$	(Conjunction of constraints)

where $\wedge$ is associative and $\mathtt{id}$ is a neutral element

Some ρ_g-terms





$[x = g(y) \wedge y = g(x)]f(x, y)$

$(2 * x) \rightarrowtriangle (\ [y = x](y + y))$

Current topics

- Expressiveness ($e.g.$ encoding TRSs)
- Type systems
- Logical aspects
- Models
- Matching and unification
- Implementation