Names in Higher-Order Rewriting

Vincent van Oostrom

Theoretical Philosophy
Universiteit Utrecht
The Netherlands

June 8, 2007

Higher-Order Rewriting

HRS meta-theory

Lambda-calculus with explicit substitutions

Lambda-calculus with patterns

CL : TRS = Lambda-calculus : HRS

Combinatory Logic, Lambda-calculus

first-/higher-order term rewriting systems

CL : TRS = Lambda-calculus : HRS

Combinatory Logic, Lambda-calculus

- not closed under rule manipulations

first-/higher-order term rewriting systems

- closed under many rule manipulations

CL : TRS = Lambda-calculus : HRS

Combinatory Logic, Lambda-calculus

- not closed under rule manipulations
- rule schemes

first-/higher-order term rewriting systems

- closed under many rule manipulations
- rules

CL : TRS = Lambda-calculus : HRS

Combinatory Logic, Lambda-calculus

- not closed under rule manipulations
- rule schemes
- logical

first-/higher-order term rewriting systems

- closed under many rule manipulations
- rules
- algebraic

Higher-Order Equational Logic ($=$)

Terms over (simply) typed signature

Inference system:

Higher-Order Equational Logic (=)

Terms over (simply) typed signature

Inference system:

- equivalence rules (reflexivity,symmetry,transitivity)

Higher-Order Equational Logic ($=$)

Terms over (simply) typed signature

Inference system:

- equivalence rules (reflexivity,symmetry,transitivity)
- congruence rules (application,abstraction)

Higher-Order Equational Logic ($=$)

Terms over (simply) typed signature

Inference system:

- equivalence rules (reflexivity,symmetry,transitivity)
- congruence rules (application,abstraction)
- $\alpha\beta\eta$ rule schemes

Higher-Order Equational Logic ($=$)

Terms over (simply) typed signature

Inference system:

- equivalence rules (reflexivity,symmetry,transitivity)
- congruence rules (application,abstraction)
- $\alpha\beta\eta$ rule schemes
- user-defined rules R of terms of same type ($\ell \to r$)

Higher-Order Rewriting ($\longrightarrow$)

- Drop symmetry, allow transitivity only at top level

Higher-Order Rewriting ($\longrightarrow$)

- Drop symmetry, allow transitivity only at top level
- HRS: modulo $\alpha\beta\eta$

Higher-Order Rewriting ($\longrightarrow$)

- Drop symmetry, allow transitivity only at top level
- HRS: modulo $\alpha\beta\eta$
- IDTS: modulo α, but $\beta\eta$ as steps

Higher-Order Rewriting ($\longrightarrow$)

- Drop symmetry, allow transitivity only at top level
- HRS: modulo $\alpha\beta\eta$
- IDTS: modulo α, but $\beta\eta$ as steps

Theorem

$=_R = \leftrightarrow^*_{R(\beta\eta)}$

Higher-Order Rewriting ($\longrightarrow$)

- Drop symmetry, allow transitivity only at top level
- HRS: modulo $\alpha\beta\eta$
- IDTS: modulo α, but $\beta\eta$ as steps

Theorem

$=_R \; = \; \leftrightarrow^*_{R(\beta\eta)}$

Decide equational theory via rewriting

HRS Terms, Rules, Rewriting

Signature:
(Simply) typed symbols

Terms:
λ-terms modulo $\alpha\beta\eta$ over signature
represented by their $\beta\eta$-normal form

Rules:
Pairs of terms of same type, lhs a pattern:

Definition

Pattern: free vars have only distinct bound vars as arguments.

Steps for rule $\ell \to r$:
$s =_{\alpha\beta\eta} C[\lambda\vec{m}.\ell] \to C[\lambda\vec{m}.r] =_{\alpha\beta\eta} t$

TRS as HRS

Signature:

$$\begin{array}{rcl} 0 & : & \iota \\ s & : & \iota \to \iota \\ + & : & \iota \to \iota \to \iota \end{array}$$

Rules for $m, n{:}\iota$

$$\begin{array}{rcl} +\, m\, 0 & \to & m \\ +\, m\,(s\, n) & \to & s\,(+\, m\, n) \end{array}$$

Steps:

$$\begin{array}{rcl} +\, 0\,(s\, 0) & =_{\alpha\beta\eta} & (\lambda mn.+ m\,(s\, n))\, 0\, 0 \\ & \to & (\lambda mn.s\,(+\, m\, n))\, 0\, 0 \\ & =_{\alpha\beta\eta} & s\,(+\, 0\, 0) \\ & =_{\alpha\beta\eta} & s\,((\lambda m.+ m\, 0)\, 0) \\ & \to & s\,((\lambda m.m)\, 0) \\ & =_{\alpha\beta\eta} & s\, 0 \end{array}$$

Lambda-calculus as HRS

Signature:

$$\begin{aligned} \mathsf{app} &: o \to (o \to o) \\ \mathsf{lam} &: (o \to o) \to o \end{aligned}$$

Rules for $M{:}o \to o$, $N{:}o$

$$\begin{aligned} \mathsf{app}\,(\mathsf{lam}\,\lambda x.M\,x)\,N &\to M\,N \\ \mathsf{lam}\,\lambda x.\mathsf{app}\,M\,x &\to M \end{aligned}$$

Steps:

$$\begin{aligned} &\mathsf{app}\,(\mathsf{lam}\,\lambda y.y)\,(\mathsf{lam}\,\lambda z.z) \\ &\quad =_{\alpha\beta\eta} (\lambda MN.\mathsf{app}\,(\mathsf{lam}\,\lambda x.M\,x)\,N)\,(\lambda y.y)\,(\mathsf{lam}\,\lambda z.z) \\ &\quad \to (\lambda MN.M\,N)\,(\lambda y.y)\,(\mathsf{lam}\,\lambda z.z) \\ &\quad =_{\alpha\beta\eta} \mathsf{lam}\,\lambda z.z \end{aligned}$$

$$\mathsf{lam}\,\lambda x.\mathsf{app}\,x\,x \quad \neq_{\alpha\beta\eta} \quad (\lambda M.\mathsf{lam}\,\lambda x.\mathsf{app}\,M\,x)\,t$$

HRS meta-theory

Generalization of TRS and Lambda-calculus
Combined difficulties:

HRS meta-theory

Generalization of TRS and Lambda-calculus
Combined difficulties:

- TRS $\Rightarrow$ arbitrary rules (overlap)

HRS meta-theory

Generalization of TRS and Lambda-calculus
Combined difficulties:

- TRS $\Rightarrow$ arbitrary rules (overlap)
- Lambda-calculus $\Rightarrow$ second-orderness (nesting)

HRS meta-theory

Generalization of TRS and Lambda-calculus
Combined difficulties:

- TRS $\Rightarrow$ arbitrary rules (overlap)
- Lambda-calculus $\Rightarrow$ second-orderness (nesting)

patterns make HRSs first-order-like

HRS meta-theory

Generalization of TRS and Lambda-calculus
Combined difficulties:

- TRS $\Rightarrow$ arbitrary rules (overlap)
- Lambda-calculus $\Rightarrow$ second-orderness (nesting)

patterns make HRSs first-order-like
orthogonality makes HRSs λ-calculus-like

HRS meta-theory: Critical Pair Lemma

Definition

Critical Pair: pair of reducts of most general overlap of lhss.

(Invited talk this afternoon)

HRS meta-theory: Critical Pair Lemma

Definition

Critical Pair: pair of reducts of most general overlap of lhss.

(Invited talk this afternoon)
For Lambda-calculus HRS:

$$\mathsf{app}\, M\, N \leftarrow \mathsf{app}\,(\mathsf{lam}\,\lambda x.\mathsf{app}\, M\, x)\, N \rightarrow \mathsf{app}\, M\, N$$

$$\mathsf{lam}\,\lambda y.M\, y \leftarrow \mathsf{lam}\,\lambda x.\mathsf{app}\,(\mathsf{lam}\,\lambda y.M\, y)\, x \rightarrow \mathsf{lam}\,\lambda x.M\, x$$

HRS meta-theory: Critical Pair Lemma

Definition

Critical Pair: pair of reducts of most general overlap of lhss.

(Invited talk this afternoon)
For Lambda-calculus HRS:

$$\mathsf{app}\, M\, N \leftarrow \mathsf{app}\,(\mathsf{lam}\, \lambda x.\mathsf{app}\, M\, x)\, N \rightarrow \mathsf{app}\, M\, N$$

$$\mathsf{lam}\, \lambda y.M\, y \leftarrow \mathsf{lam}\, \lambda x.\mathsf{app}\,(\mathsf{lam}\, \lambda y.M\, y)\, x \rightarrow \mathsf{lam}\, \lambda x.M\, x$$

Theorem

Locally confluent iff all critical pairs are.

HRS meta-theory: Critical Pair Lemma

Definition

Critical Pair: pair of reducts of most general overlap of lhss.

(Invited talk this afternoon)

For Lambda-calculus HRS:

$$\mathsf{app}\, M\, N \leftarrow \mathsf{app}\,(\mathsf{lam}\, \lambda x.\mathsf{app}\, M\, x)\, N \rightarrow \mathsf{app}\, M\, N$$

$$\mathsf{lam}\, \lambda y.M\, y \leftarrow \mathsf{lam}\, \lambda x.\mathsf{app}\,(\mathsf{lam}\, \lambda y.M\, y)\, x \rightarrow \mathsf{lam}\, \lambda x.M\, x$$

Theorem

Locally confluent iff all critical pairs are.

$\Rightarrow$ for terminating $\rightarrow_R$, $\rightarrow_R$ confluent, $=_R$ decidable.

HRS meta-theory: Bounded termination

Definition

Bounded reduction: creation depth of redexes bounded

HRS meta-theory: Bounded termination

Definition

Bounded reduction: creation depth of redexes bounded

rule $a \longrightarrow a$:

$a \longrightarrow a \longrightarrow a \longrightarrow a$ bounded (by 3)

$a \longrightarrow a \longrightarrow a \longrightarrow \dots$ not bounded

HRS meta-theory: Bounded termination

Definition

Bounded reduction: creation depth of redexes bounded

rule $a \rightarrow a$:

$a \rightarrow a \rightarrow a \rightarrow a$ bounded (by 3)

$a \rightarrow a \rightarrow a \rightarrow \ldots$ not bounded

Theorem

Bounded reductions are terminating.

HRS meta-theory: Bounded termination

Definition

Bounded reduction: creation depth of redexes bounded

rule $a \to a$:

$a \to a \to a \to a$ bounded (by 3)

$a \to a \to a \to \ldots$ not bounded

Theorem

Bounded reductions are terminating.

$\Rightarrow$ finite developments (bound 1)

$\Rightarrow$ reduction up to order of contraction (permutation equivalence)

$\Rightarrow$ neededness, normalisation of needed strategy

HRS meta-theory: Left-linear + fully-extended $\Rightarrow$ Standardisation

Definition

Left-linear: lhss are linear

Fully-extended/applied: free vars have all bound vars as arguments

HRS meta-theory: Left-linear + fully-extended $\Rightarrow$ Standardisation

Definition

Left-linear: lhss are linear

Fully-extended/applied: free vars have all bound vars as arguments

All rules above left-linear

eta-rule not fully-extended.

HRS meta-theory: Left-linear + fully-extended $\Rightarrow$ Standardisation

Definition
Left-linear: lhss are linear
Fully-extended/applied: free vars have all bound vars as arguments

All rules above left-linear
eta-rule not fully-extended.

Definition
Steps Out-of-order: inside-out or right-to-left
Standardisation: swap out-of-order steps

HRS meta-theory: Left-linear + fully-extended $\Rightarrow$ Standardisation

Definition

Left-linear: lhss are linear

Fully-extended/applied: free vars have all bound vars as arguments

All rules above left-linear

eta-rule not fully-extended.

Definition

Steps Out-of-order: inside-out or right-to-left

Standardisation: swap out-of-order steps

Theorem

Left-linear + fully-extended $\Rightarrow$ standardisation ends in standard

HRS meta-theory: Left-linear + fully-extended $\Rightarrow$ Standardisation

Definition

Left-linear: lhss are linear

Fully-extended/applied: free vars have all bound vars as arguments

All rules above left-linear

eta-rule not fully-extended.

Definition

Steps Out-of-order: inside-out or right-to-left

Standardisation: swap out-of-order steps

Theorem

Left-linear + fully-extended $\Rightarrow$ standardisation ends in standard

$\Rightarrow$ Standardised reduction permutation equivalent to original

$\Rightarrow$ normal order sound to implement Lambda-calculus/FP.

HRS meta-theory: Orthogonal $\Rightarrow$ Confluent

Definition

Orthogonal: left-linear and no criticial pairs.

HRS meta-theory: Orthogonal $\Rightarrow$ Confluent

Definition

Orthogonal: left-linear and no criticial pairs.

All rules above.

Non-example: add $\mathsf{eq}(x, x) \to \mathsf{true}$

HRS meta-theory: Orthogonal $\Rightarrow$ Confluent

Definition

Orthogonal: left-linear and no critical pairs.

All rules above.
Non-example: add $\mathrm{eq}(x, x) \longrightarrow \mathrm{true}$

Theorem

Orthogonal $\Rightarrow$ confluent

HRS meta-theory: Orthogonal $\Rightarrow$ Confluent

Definition

Orthogonal: left-linear and no criticial pairs.

All rules above.
Non-example: add $\mathrm{eq}(x, x) \to \mathrm{true}$

Theorem

Orthogonal $\Rightarrow$ *confluent*

$\Rightarrow$ all reductions to normal form permutation equivalent
$\Rightarrow$ unique normal forms (normalising strategy $=_R$ decidable)

HRS meta-theory: RPO termination via semantic labelling

HRS meta-theory: RPO termination via semantic labelling

Definition

RPO termination $\ell >_{RPO} r$:

$>_{RPO}$ obtained by lifting wfo $>$ on signature to terms compatible with computability/reducibility

HRS meta-theory: RPO termination via semantic labelling

Definition

RPO termination $\ell >_{RPO} r$:
$>_{RPO}$ obtained by lifting wfo $>$ on signature to terms compatible with computability/reducibility

Theorem

If $\ell >_{RPO} r$ *then* $\rightarrow$ *terminating*

HRS meta-theory: RPO termination via semantic labelling

Definition

RPO termination $\ell >_{RPO} r$:
$>_{RPO}$ obtained by lifting wfo $>$ on signature to terms
compatible with computability/reducibility

Theorem

If $\ell >_{RPO} r$ then $\rightarrow$ terminating

Definition

Semantics: tutorial this morning, and $[\![\ell]\!] = [\![r]\!]$

HRS meta-theory: RPO termination via semantic labelling

Definition

RPO termination $\ell >_{RPO} r$:
$>_{RPO}$ obtained by lifting wfo $>$ on signature to terms compatible with computability/reducibility

Theorem

If $\ell >_{RPO} r$ then $\rightarrow$ terminating

Definition

Semantics: tutorial this morning, and $[\![\ell]\!] = [\![r]\!]$

Definition

Labelling: label symbols by arguments semantics, labelled rules.

Semantics guarantees labelling invariant under reduction

HRS meta-theory: RPO termination via semantic labelling

Definition

RPO termination $\ell >_{RPO} r$:
$>_{RPO}$ obtained by lifting wfo $>$ on signature to terms compatible with computability/reducibility

Theorem

If $\ell >_{RPO} r$ then $\rightarrow$ terminating

Definition

Semantics: tutorial this morning, and $[\![\ell]\!] = [\![r]\!]$

Definition

Labelling: label symbols by arguments semantics, labelled rules.

Semantics guarantees labelling invariant under reduction

Theorem

If labelled system RPO terminating, then $\rightarrow_R$ terminating

Example: Lambda-labelled explicit subs are RPO-terminating.

Lambda-calculus with explicit subs: usual presentation

$$
\begin{array}{rcl}
(\lambda x.M)N & \rightarrow & M\langle x{:=}N\rangle \\
x\langle x{:=}N\rangle & \rightarrow & N \\
y\langle x{:=}N\rangle & \rightarrow & y \quad \text{where } y \neq x \\
(M_1 M_2)\langle x{:=}N\rangle & \rightarrow & M_1\langle x{:=}N\rangle M_2\langle x{:=}N\rangle \\
(\lambda y.M)\langle x{:=}N\rangle & \rightarrow & \lambda y.M\langle x{:=}N\rangle
\end{array}
$$

Lambda-calculus with explicit subs: naïve HRS

Signature:

$$\begin{array}{rcl} \mathsf{app} & : & o \rightarrow (o \rightarrow o) \\ \mathsf{lam} & : & (o \rightarrow o) \rightarrow o \\ \_\langle \_ := \_ \rangle & : & (o \leftarrow o) \rightarrow o \rightarrow o \end{array}$$

Lambda-calculus with explicit subs: naïve HRS

Signature:

$$
\begin{array}{rcl}
\mathsf{app} & : & o \to (o \to o) \\
\mathsf{lam} & : & (o \to o) \to o \\
\_\langle \_{:=}\_ \rangle & : & (o \leftarrow o) \to o \to o
\end{array}
$$

Rules:

$$
\begin{array}{rcl}
\mathsf{app}(\mathsf{lam} \lambda x. M\, x) N & \to & M\, x\langle x{:=}N\rangle \\
x\langle x{:=}N\rangle & \to & N \\
y\langle x{:=}N\rangle & \to & y \\
(\mathsf{app}\,(M_1\, x)\,(M_2\, x))\langle x{:=}N\rangle & \to & \mathsf{app}\,(M_1\, x)\langle x{:=}N\rangle\,(M_2\, x)\langle x{:=}N\rangle \\
(\mathsf{lam}\,\lambda y. M\, x\, y)\langle x{:=}N\rangle & \to & \mathsf{lam}\,\lambda y.(M\, x\, y)\langle x{:=}N\rangle
\end{array}
$$

Lambda-calculus with explicit subs: naïve HRS

Signature:

$$\begin{array}{rcl} \mathsf{app} & : & o \to (o \to o) \\ \mathsf{lam} & : & (o \to o) \to o \\ \_\langle\_{:=}\_\rangle & : & (o \leftarrow o) \to o \to o \end{array}$$

Rules:

$$\begin{array}{rcl} \mathsf{app}(\mathsf{lam}\lambda x.M\,x)N & \to & M\,x\langle x{:=}N\rangle \\ x\langle x{:=}N\rangle & \to & N \\ y\langle x{:=}N\rangle & \to & y \\ (\mathsf{app}\,(M_1\,x)\,(M_2\,x))\langle x{:=}N\rangle & \to & \mathsf{app}\,(M_1\,x)\langle x{:=}N\rangle\,(M_2\,x)\langle x{:=}N\rangle \\ (\mathsf{lam}\,\lambda y.M\,x\,y)\langle x{:=}N\rangle & \to & \mathsf{lam}\,\lambda y.(M\,x\,y)\langle x{:=}N\rangle \end{array}$$

Problems with third rule:

- not faithful: y term var, substitute any (closed) term for it

Lambda-calculus with explicit subs: naïve HRS

Signature:

$$\begin{array}{rcl} \mathsf{app} & : & o \to (o \to o) \\ \mathsf{lam} & : & (o \to o) \to o \\ \_\langle \_ := \_ \rangle & : & (o \leftarrow o) \to o \to o \end{array}$$

Rules:

$$\begin{array}{rcl} \mathsf{app}(\mathsf{lam}\lambda x.M\,x)N & \to & M\,x\langle x{:=}N\rangle \\ x\langle x{:=}N\rangle & \to & N \\ y\langle x{:=}N\rangle & \to & y \\ (\mathsf{app}\,(M_1\,x)\,(M_2\,x))\langle x{:=}N\rangle & \to & \mathsf{app}\,(M_1\,x)\langle x{:=}N\rangle\,(M_2\,x)\langle x{:=}N\rangle \\ (\mathsf{lam}\,\lambda y.M\,x\,y)\langle x{:=}N\rangle & \to & \mathsf{lam}\,\lambda y.(M\,x\,y)\langle x{:=}N\rangle \end{array}$$

Problems with third rule:

- not faithful: y term var, substitute any (closed) term for it
- not fully-extended: term substituted for y may not contain x

Lambda-calculus with explicit subs: less naïve HRS

Signature:

$$\begin{array}{rcl} \mathsf{app} & : & o \to (o \to o) \\ \mathsf{lam} & : & (\nu \to o) \to o \\ \mathsf{var} & : & \nu \to o \\ \_\langle\_{:=}\_\rangle & : & (o \leftarrow \nu) \to o \to o \end{array}$$

Lambda-calculus with explicit subs: less naïve HRS

Signature:

$$\begin{array}{rcl} \mathsf{app} & : & o \to (o \to o) \\ \mathsf{lam} & : & (\nu \to o) \to o \\ \mathsf{var} & : & \nu \to o \\ \_\langle \_ {:=} \_ \rangle & : & (o \leftarrow \nu) \to o \to o \end{array}$$

Rules:

$$\begin{array}{rcl} \mathsf{app}(\mathsf{lam} \lambda x. M\, x) N & \to & (M\, x)\langle x{:=}N \rangle \\ (\mathsf{var}\, x)\langle x{:=}N \rangle & \to & N \\ (\mathsf{var}\, y)\langle x{:=}N \rangle & \to & y \\ (\mathsf{app}(M_1\, x)(M_2\, x))\langle x{:=}N \rangle & \to & \mathsf{app}(M_1\, x)\langle x{:=}N \rangle (M_2\, x)\langle x{:=}N \rangle \\ (\mathsf{lam}\, \lambda y. M\, x\, y)\langle x{:=}N \rangle & \to & \mathsf{lam}\, \lambda y. (M\, x\, y)\langle x{:=}N \rangle \end{array}$$

Lambda-calculus with explicit subs: less naïve HRS

Signature:

$$\begin{array}{rcl} \mathsf{app} & : & o \to (o \to o) \\ \mathsf{lam} & : & (\nu \to o) \to o \\ \mathsf{var} & : & \nu \to o \\ \_\langle\_{:=}\_\rangle & : & (o \leftarrow \nu) \to o \to o \end{array}$$

Rules:

$$\begin{array}{rcl} \mathsf{app}(\mathsf{lam}\lambda x.M\,x)N & \to & (M\,x)\langle x{:=}N\rangle \\ (\mathsf{var}\,x)\langle x{:=}N\rangle & \to & N \\ (\mathsf{var}\,y)\langle x{:=}N\rangle & \to & y \\ (\mathsf{app}(M_1\,x)(M_2\,x))\langle x{:=}N\rangle & \to & \mathsf{app}(M_1\,x)\langle x{:=}N\rangle(M_2\,x)\langle x{:=}N\rangle \\ (\mathsf{lam}\,\lambda y.M\,x\,y)\langle x{:=}N\rangle & \to & \mathsf{lam}\,\lambda y.(M\,x\,y)\langle x{:=}N\rangle \end{array}$$

Problem with third rule?:

- still not fully-extended

Lambda-calculus with explicit subs: less naïve HRS

Signature:

$$\begin{array}{rcl} \mathsf{app} & : & o \to (o \to o) \\ \mathsf{lam} & : & (\nu \to o) \to o \\ \mathsf{var} & : & \nu \to o \\ \_\langle \_{:=}\_ \rangle & : & (o \leftarrow \nu) \to o \to o \end{array}$$

Rules:

$$\begin{array}{rcl} \mathsf{app}(\mathsf{lam}\lambda x.M\,x)N & \to & (M\,x)\langle x{:=}N\rangle \\ (\mathsf{var}\,x)\langle x{:=}N\rangle & \to & N \\ (\mathsf{var}\,y)\langle x{:=}N\rangle & \to & y \\ (\mathsf{app}(M_1\,x)(M_2\,x))\langle x{:=}N\rangle & \to & \mathsf{app}(M_1\,x)\langle x{:=}N\rangle(M_2\,x)\langle x{:=}N\rangle \\ (\mathsf{lam}\,\lambda y.M\,x\,y)\langle x{:=}N\rangle & \to & \mathsf{lam}\,\lambda y.(M\,x\,y)\langle x{:=}N\rangle \end{array}$$

Problem with third rule?:

- still not fully-extended
- but doesn't matter since never substituted for names

Lambda-calculus with patterns: usual presentation

Terms:

$$M ::= x \mid MM \mid \lambda M.M$$

free variables of abstracted term bound in body

Rule scheme:

$$(\lambda P.M)P^\sigma \to M^\sigma$$

Steps as usual, e.g.

$$(\lambda(\lambda z.zxy).x)\lambda z.zMN \to M$$

Lambda-calculus with patterns: usual presentation

Terms:

$$M ::= x \mid MM \mid \lambda M.M$$

free variables of abstracted term bound in body

Rule scheme:

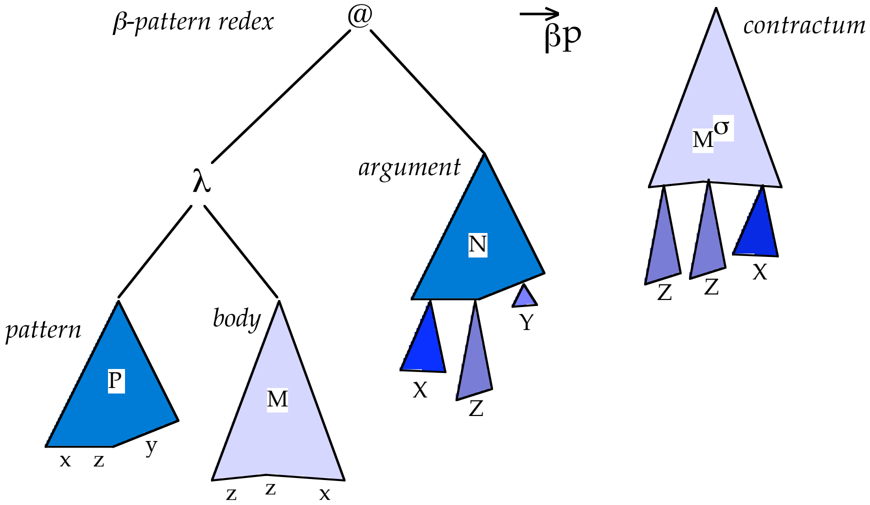
$$(\lambda P.M)P^{\sigma} \to M^{\sigma}$$

Steps as usual, e.g.

$$(\lambda(\lambda z.zxy).x)\lambda z.zMN \to M$$

with syntactic sugar:

$$(\lambda\langle x, y\rangle.x)\langle M, N\rangle \to M$$


β-pattern redex
@
$\xrightarrow{}_{\beta p}$
contractum
M^σ
λ
argument
N
pattern
P
body
M
x z y
z z x
X
Z
Y
Z Z X

Lambda-calculus with patterns: HRS

Rules:

$$\mathsf{app}\,(\mathsf{lam}\,\lambda\vec{x}(P\,\vec{x}).(Z\,\vec{x}))(P\,\vec{Z}) \quad\rightarrow\quad Z\,\vec{Z}$$

for every pattern P

Lambda-calculus with patterns: HRS

Rules:

$$\mathsf{app}\,(\mathsf{lam}\,\lambda\vec{x}(P\,\vec{x}).(Z\,\vec{x}))(P\,\vec{Z}) \quad \rightarrow \quad Z\,\vec{Z}$$

for every pattern P

Theorem

Abstracted terms linear and not narrowable $\Rightarrow$ confluent

Proof.

Orthogonal HRS! $\square$

Pure pattern calculus: part 1

$$d ::= x \ (x \in \varphi) \mid d \ t$$
$$e ::= d \mid [\theta]\, t \to t$$

Definition 7 (Basic Matching). *The* basic matching $\{\!\{u \triangleright_\theta \ p\}\!\}_\gamma$ *of a term* p *(called the* pattern*) against a term* u *(called the* argument*) relative to a set* θ *of* binding variables *and a disjoint set* γ *of* constructing variables *(or* constructors*) is the partial operation defined by applying the following equations in order*

$\{\!\{u \triangleright_\theta \ x\}\!\}_\gamma$	$:= \{u/x\}$	if $x \in \theta$
$\{\!\{x \triangleright_\theta \ x\}\!\}_\gamma$	$:= \{\}$	if $x \in \gamma$
$\{\!\{v \ u \triangleright_\theta \ q \ p\}\!\}_\gamma$	$:= \{\!\{v \triangleright_\theta \ q\}\!\}_\gamma \uplus \{\!\{u \triangleright_\theta \ p\}\!\}_\gamma$	if $q \ p$ is a γ, θ-matchable form and $v \ u$ is a γ-matchable form
$\{\!\{u \triangleright_\theta \ p\}\!\}_\gamma$	$:=$ `none`	if p is a γ, θ-matchable form and u is a γ-matchable form
$\{\!\{u \triangleright_\theta \ p\}\!\}_\gamma$	$:=$ undefined	otherwise.

Pure pattern calculus: part 2

$$([\theta]\, p \to s)\ u >_\gamma \{u/[\theta]\ p\}_\gamma\ s$$

$$\frac{}{([\theta]\, p \to s)\ u \twoheadrightarrow_\gamma \{u/[\theta]\ p\}_\gamma\ s} \qquad \frac{r \twoheadrightarrow_\gamma r'}{r\ u \twoheadrightarrow_\gamma r'\ u} \qquad \frac{u \twoheadrightarrow_\gamma u'}{r\ u \twoheadrightarrow_\gamma r\ u'}$$

$$\frac{p \twoheadrightarrow_{\gamma,\theta} p'}{[\theta]\, p \to s \ \twoheadrightarrow_\gamma [\theta]\, p' \to s} \qquad \frac{s \twoheadrightarrow_\gamma s'}{[\theta]\, p \to s \ \twoheadrightarrow_\gamma [\theta]\, p \to s'}$$

Pure pattern calculus: part 2

$$([\theta]\, p \to s)\ u >_\gamma \{u/[\theta]\ p\}_\gamma\ s$$

$$\frac{}{([\theta]\, p \to s)\ u \twoheadrightarrow_\gamma \{u/[\theta]\ p\}_\gamma\ s} \qquad \frac{r \twoheadrightarrow_\gamma r'}{r\ u \twoheadrightarrow_\gamma r'\ u} \qquad \frac{u \twoheadrightarrow_\gamma u'}{r\ u \twoheadrightarrow_\gamma r\ u'}$$

$$\frac{p \twoheadrightarrow_{\gamma,\theta} p'}{[\theta]\, p \to s \ \ \twoheadrightarrow_\gamma [\theta]\, p' \to s} \qquad \frac{s \twoheadrightarrow_\gamma s'}{[\theta]\, p \to s \ \ \twoheadrightarrow_\gamma [\theta]\, p \to s'}$$

Theorem

Pure pattern calculus is confluent

Proof.

Tait–Martin-Löf ☐

Pure pattern calculus: HRS

Rules:

$$\mathsf{app}\,(\mathsf{lam}\,(\lambda\vec{a}.(P\,\vec{a}))\,(\lambda\vec{x}.(Z\,\vec{x})))(P\,\vec{Z}) \quad \longrightarrow \quad Z\,\vec{Z}$$

for every pattern P

Pure pattern calculus: HRS

Rules:

$$\mathsf{app}\,(\mathsf{lam}\,(\lambda\vec{a}.(P\,\vec{a}))\,(\lambda\vec{x}.(Z\,\vec{x})))(P\,\vec{Z}) \quad \to \quad Z\,\vec{Z}$$

for every pattern P

Theorem

Pure pattern calculus is confluent

Proof.

By orthogonality for HRSs, with non-substitutable names. $\square$