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
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 (\(\equiv\))

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 \((\ell \rightarrow r)\)
Higher-Order Rewriting ($\rightarrow$)

- Drop symmetry, allow transitivity only at top level
Higher-Order Rewriting (→)

- Drop symmetry, allow transitivity only at top level
- HRS: modulo $\alpha\beta\eta$
Higher-Order Rewriting ($\rightarrow$)

- Drop symmetry, allow transitivity only at top level
- HRS: modulo $\alpha\beta\eta$
- IDTS: modulo $\alpha$, but $\beta\eta$ as steps
Higher-Order Rewriting (→)

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

**Theorem**

$$=R = \leftrightarrow^*_R(\beta\eta)$$
Higher-Order Rewriting (→)

- Drop symmetry, allow transitivity only at top level
- HRS: modulo $\alpha \beta \eta$
- IDTS: modulo $\alpha$, 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:
$\lambda$-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 $\ell \rightarrow r$:
$s =_{\alpha\beta\eta} C[\lambda\vec{m}.\ell] \rightarrow C[\lambda\vec{m}.r] =_{\alpha\beta\eta} t$
TRS as HRS

Signature:

0 : \( \iota \)

\( s : \iota \rightarrow \iota \)

\( + : \iota \rightarrow \iota \rightarrow \iota \)

Rules for \( m, n: \iota \)

\( + m 0 \rightarrow m \)

\( + m (s n) \rightarrow s (+ m n) \)

Steps:

\( + 0 (s 0) =_{\alpha \beta \eta} (\lambda mn. (+ m (s n)) 0 0 \)

\rightarrow (\lambda mn. s (+ m n)) 0 0 \)

\( =_{\alpha \beta \eta} s (+ 0 0) \)

\( =_{\alpha \beta \eta} s ((\lambda m. (+ m 0) 0) \)

\rightarrow s ((\lambda m.m) 0) \)

\( =_{\alpha \beta \eta} s 0 \)
Lambda-calculus as HRS

Signature:

\[
\begin{align*}
\text{app} & : o \rightarrow (o \rightarrow o) \\
\text{lam} & : (o \rightarrow o) \rightarrow o
\end{align*}
\]

Rules for \(M:o \rightarrow o\), \(N:o\):

\[
\begin{align*}
\text{app}\ (\text{lam}\ \lambda x.\ M\ x)\ N & \rightarrow M\ N \\
\text{lam}\ \lambda x.\ \text{app}\ M\ x & \rightarrow M
\end{align*}
\]

Steps:

\[
\begin{align*}
\text{app}\ (\text{lam}\ \lambda y.y)\ (\text{lam}\ \lambda z.z) & \\
=_{\alpha\beta\eta} & (\lambda MN.\ \text{app}\ (\text{lam}\ \lambda x.\ M\ x)\ N)\ (\lambda y.y)\ (\text{lam}\ \lambda z.z) \\
& \rightarrow (\lambda MN.\ M\ N)\ (\lambda y.y)\ (\text{lam}\ \lambda z.z) \\
=_{\alpha\beta\eta} & \text{lam}\ \lambda z.z
\end{align*}
\]

\[
\begin{align*}
\text{lam}\ \lambda x.\ \text{app}\ x\ x & \neq_{\alpha\beta\eta} (\lambda M.\ \text{lam}\ \lambda x.\ \text{app}\ M\ x)\ t
\end{align*}
\]
HRS meta-theory

Generalization of TRS and Lambda-calculus

Combined difficulties:
HRS meta-theory

Generalization of TRS and Lambda-calculus
Combined difficulties:
- TRS ⇒ arbitrary rules (overlap)
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 $\lambda$-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:

\[
\text{app } M \, N \leftarrow \text{app } (\text{lam } \lambda x. \text{app } M \, x) \, N \rightarrow \text{app } M \, N
\]

\[
\text{lam } \lambda y. M \, y \leftarrow \text{lam } \lambda x. \text{app } (\text{lam } \lambda y. M \, y) \, x \rightarrow \text{lam } \lambda x. M \, x
\]
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:

\[
\text{app } M \ N \leftarrow \text{app } (\text{lam } \lambda x.\text{app } M \ x) \ N \rightarrow \text{app } M \ N
\]

\[
\text{lam } \lambda y.\ M \ y \leftarrow \text{lam } \lambda x.\text{app } (\text{lam } \lambda y.\ M \ y) \ x \rightarrow \text{lam } \lambda x.\ M \ x
\]

Theorem

Locally confluent iff all critical pairs are.
Definition

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

(Invited talk this afternoon)

For Lambda-calculus HRS:

\[
\text{app } M N \leftarrow \text{app } (\text{lam } \lambda x. \text{app } M x) N \rightarrow \text{app } M N
\]

\[
\text{lam } \lambda y. M y \leftarrow \text{lam } \lambda x. \text{app } (\text{lam } \lambda y. M y) x \rightarrow \text{lam } \lambda x. M x
\]

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 \( a \rightarrow a \):
\[ a \rightarrow a \rightarrow a \rightarrow a \text{ bounded (by 3)} \]
\[ a \rightarrow a \rightarrow a \rightarrow \ldots \text{ not bounded} \]
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 $a \rightarrow a$:

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

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

**Theorem**

*Bounded reductions are terminating.*

$\Rightarrow$ 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 ⇒ 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 ⇒ 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 ⇒ 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 ⇒ standardisation ends in standard
⇒ Standardised reduction permutation equivalent to original
⇒ normal order sound to implement Lambda-calculus/FP.
HRS meta-theory: Orthogonal $\Rightarrow$ Confluent

Definition

Orthogonal: left-linear and no critical pairs.
HRS meta-theory: Orthogonal $\Rightarrow$ Confluent

Definition
Orthogonal: left-linear and no critical pairs.
All rules above.
Non-example: add $\text{eq}(x, x) \rightarrow \text{true}$
Definition

**Orthogonal**: left-linear and no critical pairs.

All rules above.

Non-example: add \( \text{eq}(x, x) \rightarrow \text{true} \)

Theorem

**Orthogonal** \( \Rightarrow \) **confluent**
HRS meta-theory: Orthogonal $\Rightarrow$ Confluent

**Definition**

Orthogonal: left-linear and no critical pairs.

All rules above.

Non-example: add $\text{eq}(x, x) \rightarrow \text{true}$

**Theorem**

Orthogonal $\Rightarrow$ confluent

$\Rightarrow$ all reductions to normal form permutation equivalent

$\Rightarrow$ unique normal forms (normalising strategy $\equiv_R$ decidable)
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 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 \( \rightarrow \) terminating

**Example:** Lambda-labelled explicit subs are RPO-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
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 $\ell >_{RPO} r$:
$\succ_{RPO}$ obtained by lifting wfo $\succ$ 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 \( \text{wfo} > \) 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

\[(\lambda x. M)N \rightarrow M\langle x:=N \rangle\]

\[x\langle x:=N \rangle \rightarrow N\]

\[y\langle x:=N \rangle \rightarrow y \quad \text{where } y \neq x\]

\[(M_1 M_2)\langle x:=N \rangle \rightarrow M_1\langle x:=N \rangle M_2\langle x:=N \rangle\]

\[(\lambda y. M)\langle x:=N \rangle \rightarrow \lambda y. M\langle x:=N \rangle\]
Lambda-calculus with explicit subs: naïve HRS

Signature:

\[
\begin{align*}
\text{app} & : \ o \rightarrow (o \rightarrow o) \\
\text{lam} & : (o \rightarrow o) \rightarrow o \\
\langle \_ := \_ \rangle & : (o \leftarrow o) \rightarrow o \rightarrow o
\end{align*}
\]
Lambda-calculus with explicit subs: naïve HRS

Signature:

\[
\begin{align*}
  \text{app} & : \ o \to (o \to o) \\
  \text{lam} & : (o \to o) \to o \\
  (-\langle-:=\rangle) & : (o \leftarrow o) \to o \to o
\end{align*}
\]

Rules:

\[
\begin{align*}
  \text{app}(\text{lam}\lambda x. M x)N & \to M x\langle x:=N \rangle \\
  x\langle x:=N \rangle & \to N \\
  y\langle x:=N \rangle & \to y \\
  (\text{app}(M_1 x)(M_2 x))\langle x:=N \rangle & \to \text{app}(M_1 x)\langle x:=N \rangle (M_2 x)\langle x:=N \rangle \\
  (\text{lam}\lambda y. M x y)\langle x:=N \rangle & \to \text{lam}\lambda y. (M x y)\langle x:=N \rangle
\end{align*}
\]
Lambda-calculus with explicit subs: naïve HRS

Signature:

\[
\begin{align*}
\text{app} &: \ o \to (o \to o) \\
\text{lam} &: \ (o \to o) \to o \\
\langle \_:=\_ \rangle &: \ (o \leftrightarrow o) \to o \to o
\end{align*}
\]

Rules:

\[
\begin{align*}
\text{app} (\text{lam} \lambda x. M x) N & \to M x \langle x:=N \rangle \\
\lambda x. M \langle x:=N \rangle & \to N \\
\lambda y. M \langle x:=N \rangle & \to y \\
(\text{app} (M_1 x) (M_2 x)) \langle x:=N \rangle & \to \text{app} (M_1 x) \langle x:=N \rangle (M_2 x) \langle x:=N \rangle \\
\text{lam} \lambda y. M x y \langle x:=N \rangle & \to \text{lam} \lambda y. (M x y) \langle x:=N \rangle
\end{align*}
\]

Problems with third rule:

- not faithful: \( y \) term var, substitute any (closed) term for it
Lambda-calculus with explicit subs: naïve HRS

Signature:

\[
\begin{align*}
\text{app} & : \ o \rightarrow (o \rightarrow o) \\
\text{lam} & : \ (o \rightarrow o) \rightarrow o \\
\langle \_:=\_ \rangle & : \ (o \leftarrow o) \rightarrow o \rightarrow o
\end{align*}
\]

Rules:

\[
\begin{align*}
\text{app}(\text{lam}\lambda x. M x) N & \rightarrow M x\langle x:=N \rangle \\
x\langle x:=N \rangle & \rightarrow N \\
y\langle x:=N \rangle & \rightarrow y \\
(\text{app}(M_1 x)(M_2 x))\langle x:=N \rangle & \rightarrow \text{app}(M_1 x)\langle x:=N \rangle (M_2 x)\langle x:=N \rangle \\
(\text{lam}\lambda y. M x y)\langle x:=N \rangle & \rightarrow \text{lam}\lambda y. (M x y)\langle x:=N \rangle
\end{align*}
\]

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:

\[
\begin{align*}
\text{app} & : \ o \rightarrow (o \rightarrow o) \\
\text{lam} & : \ (\nu \rightarrow o) \rightarrow o \\
\text{var} & : \ \nu \rightarrow o \\
\langle\_\_::=\_\_\rangle & : \ (o \leftarrow \nu) \rightarrow o \rightarrow o
\end{align*}
\]
Lambda-calculus with explicit subs: less naïve HRS

Signature:

\[
\begin{align*}
app & : o \rightarrow (o \rightarrow o) \\
lam & : (\nu \rightarrow o) \rightarrow o \\
var & : \nu \rightarrow o \\
\langle\_:=\_\rangle & : (o \leftarrow \nu) \rightarrow o \rightarrow o
\end{align*}
\]

Rules:

\[
\begin{align*}
app(lam\lambda x.M x)N & \rightarrow (M x)\langle x:=N\rangle \\
(var x)\langle x:=N\rangle & \rightarrow N \\
(var y)\langle x:=N\rangle & \rightarrow y \\
(app(M_1 x)(M_2 x))\langle x:=N\rangle & \rightarrow app(M_1 x)\langle x:=N\rangle(M_2 x)\langle x:=N\rangle \\
(lam \lambda y.M x y)\langle x:=N\rangle & \rightarrow lam \lambda y.(M x y)\langle x:=N\rangle
\end{align*}
\]
Lambda-calculus with explicit subs: less naïve HRS

Signature:

\[ \begin{align*}
app &: o \to (o \to o) \\
\lambda &: (\nu \to o) \to o \\
\nu &: o \\
\langle \nu ::= \rangle &: (o \leftarrow \nu) \to o \to o
\end{align*} \]

Rules:

\[ \begin{align*}
app(\lambda x.Mx)N & \to (Mx)\langle x::=N \rangle \\
\langle x::=N \rangle & \to N \\
\langle x::=N \rangle & \to y \\
(app(M_1x)(M_2x))\langle x::=N \rangle & \to app(M_1x)\langle x::=N \rangle(M_2x)\langle x::=N \rangle \\
\lambda y.(Mxy)\langle x::=N \rangle & \to \lambda y.(Mxy)\langle x::=N \rangle
\end{align*} \]

Problem with third rule?:

- still not fully-extended
Lambda-calculus with explicit subs: less naïve HRS

Signature:

\[
\begin{align*}
\text{app} & : o \rightarrow (o \rightarrow o) \\
\text{lam} & : (\nu \rightarrow o) \rightarrow o \\
\text{var} & : \nu \rightarrow o \\
\langle \_ := \_ \rangle & : (o \leftarrow \nu) \rightarrow o \rightarrow o
\end{align*}
\]

Rules:

\[
\begin{align*}
\text{app}(\text{lam}\lambda x.M x)N & \rightarrow (M x)\langle x := N \rangle \\
(\text{var} x)\langle x := N \rangle & \rightarrow N \\
(\text{var} y)\langle x := N \rangle & \rightarrow y \\
(\text{app}(M_1 x)(M_2 x))\langle x := N \rangle & \rightarrow \text{app}(M_1 x)\langle x := N \rangle(M_2 x)\langle x := N \rangle \\
(\text{lam} \lambda y.M x y)\langle x := N \rangle & \rightarrow \text{lam} \lambda y.(M x y)\langle x := N \rangle
\end{align*}
\]

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:

\[ (\lambda P.M)P^\sigma \rightarrow M^\sigma \]

Steps as usual, e.g.

\[ (\lambda(\lambda z.zxy).x)\lambda z.zMN \rightarrow M \]
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:

\[(\lambda P.M)P^\sigma \rightarrow M^\sigma\]

Steps as usual, e.g.

\[(\lambda(\lambda z.zxy).x)\lambda z.zMN \rightarrow M\]

with syntactic sugar:

\[(\lambda\langle x, y\rangle.x)\langle M, N\rangle \rightarrow M\]
\[ \Rightarrow \beta_p \]

\[ \lambda \]

\[ \text{pattern} \]

\[ \text{body} \]

\[ \Rightarrow \beta_p \]

\[ \text{contractum} \]
Rules:

\[
\text{app (lam } \lambda \vec{x}(P \vec{x}).(Z \vec{x})) (P \vec{Z}) \rightarrow Z \vec{Z}
\]

for every pattern \( P \)
Lambda-calculus with patterns: HRS

Rules:

\[
\text{app (lam } \lambda \vec{x}(P \vec{x}).(Z \vec{x}))(P \vec{Z}) \rightarrow Z \vec{Z}
\]

for every pattern \( P \)

**Theorem**

*Abstracted terms linear and not narrowable \( \Rightarrow \) confluent*

**Proof.**

Orthogonal HRS!
Pure pattern calculus: part 1

\[
\begin{align*}
  d &::= x \ (x \in \varphi) \mid d \ t \\
  e &::= d \mid [\theta] t \rightarrow t 
\end{align*}
\]

**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 \( \theta \) of binding variables and a disjoint set \( \gamma \) of constructing variables (or constructors) is the partial operation defined by applying the following equations in order

\[
\begin{align*}
\{ u \triangleright \theta \ x \}_\gamma &::= \{ u/x \} \quad \text{if } x \in \theta \\
\{ x \triangleright \theta \ x \}_\gamma &::= \{ \} \quad \text{if } x \in \gamma \\
\{ v \ u \triangleright \theta \ q \ p \}_\gamma &::= \{ v \triangleright \theta \ q \}_\gamma \cup \{ u \triangleright \theta \ p \}_\gamma \quad \text{if } q \ p \text{ is a } \gamma, \theta\text{-matchable form and } v \ u \text{ is a } \gamma\text{-matchable form} \\
\{ u \triangleright \theta \ p \}_\gamma &::= \text{none} \quad \text{if } p \text{ is a } \gamma, \theta\text{-matchable form and } u \text{ is a } \gamma\text{-matchable form} \\
\{ u \triangleright \theta \ p \}_\gamma &::= \text{undefined} \quad \text{otherwise.}
\end{align*}
\]
Pure pattern calculus: part 2

\((\theta p \to s) \ u \overset{\gamma}{\to} \{u/\theta\} p\) \(\gamma\) \(s\)

\[
\frac{r \to_{\gamma} r'}{\[\theta\] p \to s \to_{\gamma} \[\theta\] p' \to s}
\]

\[
\frac{[\theta] p \to s \to_{\gamma} \[\theta\] p \to s'}{[\theta] p \to s \to_{\gamma} \[\theta\] p \to s'}
\]
Pure pattern calculus: part 2

$([\theta] p \rightarrow s) u \triangleright_\gamma \{u/[\theta] p\}_{\gamma} s$

\[
\begin{array}{c}
([\theta] p \rightarrow s) u \rightarrow_\gamma \{u/[\theta] p\}_{\gamma} s \\
\hline
r \rightarrow_\gamma r' \\
\hline
u \rightarrow_\gamma u'
\end{array}
\]

\[
\begin{array}{c}
p \rightarrow_\gamma,\theta p' \\
\hline
[\theta] p \rightarrow s \rightarrow_\gamma [\theta] p' \rightarrow s
\end{array}
\]

\[
\begin{array}{c}
r u \rightarrow_\gamma r' u \\
\hline
r u \rightarrow_\gamma r u'
\end{array}
\]

\[
\begin{array}{c}
s \rightarrow_\gamma s' \\
\hline
[\theta] p \rightarrow s \rightarrow_\gamma [\theta] p \rightarrow s'
\end{array}
\]

Theorem

*Pure pattern calculus is confluent*

Proof.

Tait–Martin-Löf
Pure pattern calculus: HRS

Rules:

\[ \text{app (lam (λ\vec{a}.(P \vec{a})) (λ\vec{x}.(Z \vec{x}))(P \vec{Z}))} \rightarrow Z \vec{Z} \]

for every pattern \( P \)
Pure pattern calculus: HRS

Rules:

\[
\text{app (lam (\lambda \vec{a}. (P \vec{a}))) (\lambda \vec{x}. (Z \vec{x}))) (P \vec{Z}) \rightarrow Z \vec{Z}
\]

for every pattern \( P \)

**Theorem**

*Pure pattern calculus is confluent*

**Proof.**

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