Rho-Calculus with explicit constraint handling

Workshop Rho-Calculus

Explicit constraint handling

Why?

- to implement the ρ-calculus

Explicit constraint handling

Why?

- to implement the ρ-calculus
- to represent proof-terms of rewriting derivation [Nguyen]

Explicit constraint handling

Why?

- to implement the ρ-calculus
- to represent proof-terms of rewriting derivation [Nguyen]
- to have a precise control over matching and constraints. Better deal of errors.

Explicit constraint handling

Why?

- to implement the ρ-calculus
- to represent proof-terms of rewriting derivation [Nguyen]
- to have a precise control over matching and constraints. Better deal of errors.

How?

- by making explicit matching computations
- by making explicit application of substitutions

Run Time errors in O'CAML

```
#let car l = match l with
 x::m -> x;;
```

Run Time errors in O'CAML

```
#let car l = match l with
 x::m -> x;;

#car [];;
Exception: Match_failure (", 12, 42).
```

Run Time errors in O'CAML

```
#let car l = match l with
 x::m -> x;;

#car [];;
Exception: Match_failure (", 12, 42).

#let car l = match l with
  [] -> failwith ‘‘error in car’’
 |x::m -> x;;
```

Run Time errors in O'CAML

```
#let car l = match l with
 x::m -> x;;

#car [];;
Exception: Match_failure (", 12, 42).

#let car l = match l with
  [] -> failwith ‘‘error in car’’
 |x::m -> x;;
```

↪ Need to deal with error "by hand"

Errors in the ρ-calculus

$$car \triangleq \mathtt{Cons}(X, Y) \twoheadrightarrow X$$

$$(\mathtt{Cons}(X, Y) \twoheadrightarrow X)\ \mathtt{Empty}$$
$$\quad \mapsto_{\rho}\ [\mathtt{Cons}(X, Y) \ll \mathtt{Empty}]X$$

Errors in the ρ-calculus

$$f(g(h(X), i(a), k(l(Y))), c) \twoheadrightarrow \pi(X, Y) \quad f(g(h(a), i(b), k(l(b))), c)$$
$$\mapsto_{\rho} \ [f(g(h(X), i(a), k(l(Y))), c) \ll f(g(h(a), i(b), k(l(b))), c)] \ \pi(X, Y)$$

Errors in the ρ-calculus

$$f(g(h(X), i(a), k(l(Y))), c) \twoheadrightarrow \pi(X, Y) \quad f(g(h(a), i(b), k(l(b))), c)$$
$$\mapsto_{\rho} \ [f(g(h(X), i(a), k(l(Y))), c) \ll f(g(h(a), i(b), k(l(b))), c)] \ \pi(X, Y)$$

$$\mapsto\!\!\!\!\!\rightarrow_{Decompose_{\mathcal{F}}} \quad [X \ll a \wedge \underline{a \ll b} \wedge Y \ll b \wedge c \ll c] \quad \pi(X, Y)$$

Errors in the ρ-calculus

$$f(g(h(X), i(a), k(l(Y))), c) \twoheadrightarrow \pi(X, Y) \quad f(g(h(a), i(b), k(l(b))), c)$$
$$\mapsto_{\rho} \; [f(g(h(X), i(a), k(l(Y))), c) \ll f(g(h(a), i(b), k(l(b))), c)] \; \pi(X, Y)$$

$$\begin{array}{ll} \mapsto\!\!\!\!\!\rightarrow_{Decompose_{\mathcal{F}}} & [X \ll a \wedge \underline{a \ll b} \wedge Y \ll b \wedge c \ll c] \quad \pi(X, Y) \\ \mapsto\!\!\!\!\!\rightarrow & [a \ll b] \quad \pi(a, b) \end{array}$$

Errors in the ρ-calculus

$$f(g(h(X), i(a), k(l(Y))), c) \twoheadrightarrow \pi(X, Y) \quad f(g(h(a), i(b), k(l(b))), c)$$
$$\mapsto_{\rho} \ [f(g(h(X), i(a), k(l(Y))), c) \ll f(g(h(a), i(b), k(l(b))), c)] \ \pi(X, Y)$$

$$\begin{array}{ll} \mapsto\!\!\!\!\twoheadrightarrow_{Decompose_{\mathcal{F}}} & [X \ll a \wedge \underline{a \ll b} \wedge Y \ll b \wedge c \ll c] \quad \pi(X, Y) \\ \mapsto\!\!\!\!\twoheadrightarrow & [a \ll b] \quad \pi(a, b) \end{array}$$

➥ Explicit handling of constraints should allow a very nice and precise deal of errors.

Explicit matching decomposition

- Decompose functional symbols.
- Decompose the structure “;”.
- Do not decompose: Abstraction and Application.

Constraint application

What about?

$$[X \ll a \wedge a \ll b]X$$

Constraint application

What about?

$$[X \ll a \wedge a \ll b]X$$

➥ Do not apply constraint without solution

Constraint application

What about?

$$[X \ll a \wedge a \ll b]X$$

➥ Do not apply constraint without solution

From constraints to substitutions

$$[\, X \ll A \wedge \underbrace{\qquad\qquad}_{\mathfrak{C}} \,]B$$

Constraint application

What about?

$$[X \ll a \wedge a \ll b]X$$

➥ Do not apply constraint without solution

From constraints to substitutions

$$[\ X \ll A \wedge \underbrace{\qquad}_{\mathfrak{C}}\ ]B \quad \longrightarrow \quad [\ \underbrace{\qquad}_{\mathfrak{C}}\ ]\{\ X \ll A\ \}B$$

$$\texttt{if}\ \ X \notin \mathcal{D}om(\mathfrak{C})$$

Syntax of the ρ_{x}-calculus

Terms	A, B	$::=$	$\mathcal{X}$	(Variables)
		$\mid$	$\mathcal{K}$	(Constants)
		$\mid$	$A \twoheadrightarrow B$	(Abstraction)
		$\mid$	$A\ B$	(Fonctional application)
		$\mid$	$[\mathcal{C}]\ B$	(Constraint application)
		$\mid$	$A; B$	(Structure)
		$\mid$	$\{X \ll A\}B$	(Substitution application on terms)
Constraints	$\mathcal{C}, \mathcal{D}$	$::=$	$A \ll B$	(Match-equation)
		$\mid$	$\mathcal{C} \wedge \mathcal{D}$	(Conjonctions of contraints)
		$\mid$	$\{X \ll A\}\mathcal{C}$	(Substitution application on const.)

Semantics of the ρ_{x}-calculus

From rewrite rules to constraints

(ρ)	$(A \twoheadrightarrow B)\ C$	$\longrightarrow$	$[A \ll C]\ B$
(δ)	$(A; B)\ C$	$\longrightarrow$	$A\ C; B\ C$

Semantics of the ρ_{x}-calculus

From rewrite rules to constraints

(ρ)	$(A \twoheadrightarrow B)\ C$	$\rightarrow$	$[A \ll C]\ B$
(δ)	$(A; B)\ C$	$\rightarrow$	$A\ C; B\ C$

From constraints to substitutions

Decomposition

$(Decompose_{;})$	$A_1; A_2 \ll B_1; B_2$	$\rightarrow$	$A_1 \ll B_1 \wedge\ A_2 \ll B_2$
$(Decompose_{\mathcal{F}})$	$f(A_1, \ldots, A_n) \ll f(B_1, \ldots, B_n)$	$\rightarrow$	$A_1 \ll B_1 \wedge \ldots \wedge A_n \ll B_n$
$(NGood)$	$f(A_1, \ldots, A_n) \ll g(B_1, \ldots, B_n)$	$\rightarrow$	$f(A_1, \ldots, A_n)\ \ll^{\mathbf{ng}}\ g(B_1, \ldots, B_n)$

Semantics of the ρ_{x}-calculus

From rewrite rules to constraints

(ρ)	$(A \twoheadrightarrow B)\ C$	$\rightarrow$	$[A \ll C]\ B$
(δ)	$(A; B)\ C$	$\rightarrow$	$A\ C; B\ C$

From constraints to substitutions

Decomposition

$(Decompose_{;})$	$A_1; A_2 \ \ll\ B_1; B_2$	$\rightarrow$	$A_1 \ll B_1 \wedge\ A_2 \ll B_2$
$(Decompose_{\mathcal{F}})$	$f(A_1 \ldots A_n) \ll f(B_1 \ldots B_n)$	$\rightarrow$	$A_1 \ll B_1 \wedge \ldots \wedge A_n \ll B_n$
$(NGood)$	$f(A_1 \ldots A_n) \ll g(B_1 \ldots B_n)$	$\rightarrow$	$f(A_1 \ldots A_n)\ \ll^{\mathbf{ng}}\ g(B_1 \ldots B_n)$

From constraints to substitutions

$(ToSubst_{\wedge})$	$[X \ll A \wedge \mathcal{C}]B$	$\rightarrow$	$[\mathcal{C}](\{X \ll A\}B)$ `if` $X \notin \mathcal{D}om(\mathcal{C})$

From rewrite rules to constraints

(ρ)	$(A \twoheadrightarrow B)\ C$	$\rightarrow$	$[A \ll C]\ B$
(δ)	$(A; B)\ C$	$\rightarrow$	$A\ C; B\ C$

From constraints to substitutions

Decomposition

$(Decompose_{;})$	$A_1; A_2 \ \ll\ B_1; B_2$	$\rightarrow$	$A_1 \ll B_1 \wedge\ A_2 \ll B_2$
$(Decompose_{\mathcal{F}})$	$f(A_1, \ldots, A_n) \ll f(B_1, \ldots, B_n)$	$\rightarrow$	$A_1 \ll B_1 \wedge \ldots \wedge A_n \ll B_n$
$(NGood)$	$f(A_1, \ldots, A_n) \ll g(B_1, \ldots, B_n)$	$\rightarrow$	$f(A_1, \ldots, A_n)\ \ll^{\mathbf{ng}}\ g(B_1, \ldots, B_n)$

From constraints to substitutions

$(ToSubst_{\wedge})$	$[X \ll A \wedge \mathcal{C}]B$	$\rightarrow$	$[\mathcal{C}](\{X \ll A\}B)$ $\texttt{if}\ \ X \notin \mathcal{D}om(\mathcal{C})$

Substitution applications

$(Replace)$	$\{X \ll A\}X$	$\rightarrow$	A
$(Eliminate_{\mathcal{X}})$	$\{X \ll A\}Y$	$\rightarrow$	Y $\texttt{if}\ \ X \neq Y$
$(Eliminate_{\mathcal{F}})$	$\{X \ll A\}f$	$\rightarrow$	f
$(Share_{;})$	$\{X \ll A\}(B \fatsemi C)$	$\rightarrow$	$\{X \ll A\}B \fatsemi \{X \ll A\}C$
$(Share_{[]})$	$\{X \ll A\}([B]C)$	$\rightarrow$	$[\{X \ll A\}B]\{X \ll A\}C$
$(Share_{\twoheadrightarrow})$	$\{X \ll A\}(B \twoheadrightarrow C)$	$\rightarrow$	$B \twoheadrightarrow \{X \ll A\}C$
$(Share_{\ll})$	$\{X \ll A\}(B \ll C)$	$\rightarrow$	$B \ll \{X \ll A\}C$
$(Share_{\wedge})$	$\{X \ll A\}(\mathcal{C} \wedge \mathcal{D})$	$\rightarrow$	$\{X \ll A\}\mathcal{C}\ \wedge\ \{X \ll A\}\mathcal{D}$

Example

$$(\mathrm{not}\,(\mathrm{and}\,(X,Y)) \twoheadrightarrow \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y))\,) \qquad \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))$$

Example

$$(\mathrm{not}\,(\mathrm{and}\,(X,Y)) \twoheadrightarrow \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y))\,) \qquad \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))$$

$$\mapsto_{\rho} \qquad \Big[\mathrm{not}\,(\mathrm{and}\,(X,Y)) \ll \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))\Big]\mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y))$$

Example

$$(\mathrm{not}\,(\mathrm{and}\,(X,Y)) \twoheadrightarrow \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y))\,) \qquad \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))$$

$$\begin{array}{ll} \mapsto_{p} & \left[\mathrm{not}\,(\mathrm{and}\,(X,Y)) \ll \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))\right] \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \\ \mapsto_{Decompose} & \left[\mathrm{and}\,(X,Y) \ll (\mathrm{and}\,(\mathrm{tt},\mathrm{ff})\right] \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \end{array}$$

Example

$$(\mathrm{not}\,(\mathrm{and}\,(X,Y)) \twoheadrightarrow \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y))\,) \qquad \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))$$

$$\begin{aligned}
&\mapsto_{p} && \left[\mathrm{not}\,(\mathrm{and}\,(X,Y)) \ll \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))\right] \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \\
&\mapsto_{Decompose} && \left[\mathrm{and}\,(X,Y) \ll (\mathrm{and}\,(\mathrm{tt},\mathrm{ff})\right] \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \\
&\mapsto_{Decompose} && \left[X \ll \mathrm{tt} \wedge Y \ll \mathrm{ff}\right] \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y))
\end{aligned}$$

Example

$$(\mathrm{not}\,(\mathrm{and}\,(X,Y)) \twoheadrightarrow \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y))\,) \qquad \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))$$

$$\begin{aligned}
&\mapsto_{p} && \left[\mathrm{not}\,(\mathrm{and}\,(X,Y)) \ll \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))\right] \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \\
&\mapsto_{Decompose} && \left[\mathrm{and}\,(X,Y) \ll (\mathrm{and}\,(\mathrm{tt},\mathrm{ff})\right] \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \\
&\mapsto_{Decompose} && \left[X \ll \mathrm{tt} \wedge Y \ll \mathrm{ff}\right] \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \\
&\mapsto_{Subst} && \left[X \ll \mathrm{tt}\right] \{Y \ll \mathrm{ff}\} \mathrm{or}\,(\mathrm{not}\,(X)\ \ \mathrm{not}\,(Y))
\end{aligned}$$

Example

$$(\mathrm{not}\,(\mathrm{and}\,(X,Y)) \twoheadrightarrow \mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y))\,) \qquad \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))$$

$$\begin{array}{ll}
\mapsto_{p} & \left[\mathrm{not}\,(\mathrm{and}\,(X,Y)) \ll \mathrm{not}\,(\mathrm{and}\,(\mathrm{tt},\mathrm{ff}))\right]\mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \\
\mapsto_{Decompose} & \left[\mathrm{and}\,(X,Y) \ll (\mathrm{and}\,(\mathrm{tt},\mathrm{ff})\right]\mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \\
\mapsto_{Decompose} & \left[X \ll \mathrm{tt} \wedge Y \ll \mathrm{ff}\right]\mathrm{or}\,(\mathrm{not}\,(X)\,,\mathrm{not}\,(Y)) \\
\mapsto_{Subst} & \left[X \ll \mathrm{tt}\right]\{Y \ll \mathrm{ff}\}\mathrm{or}\,(\mathrm{not}\,(X)\ \ \mathrm{not}\,(Y)) \\
\mapsto^{*}_{Propagation} & (\mathrm{or}\,(\mathrm{not}\,(\mathrm{tt})\,,\mathrm{not}\,(\mathrm{ff})))
\end{array}$$

➪ Need to compose, to able to combine substitutions, to limit term traversal.

The $\rho_{\mathtt{xC}}$-calculus

$$(GI) \qquad X \ll A \wedge Y \ll B \quad \rightarrow \quad X \ll A \wedge_{\mathbf{g}} Y \ll B \quad \texttt{if } X \neq Y$$

The $\rho_{\mathtt{xC}}$-calculus

$$
\begin{array}{lll}
(GI) & X \ll A \wedge Y \ll B & \rightarrow \quad X \ll A \wedge_{\boldsymbol{g}} Y \ll B \\
 & & \quad \mathtt{if}\ \ X \neq Y \\
(Good) & \mathcal{C}^{\mathbf{g}} \wedge \mathcal{D}^{\mathbf{g}} & \rightarrow \quad \mathcal{C}^{\mathbf{g}} \wedge_{\boldsymbol{g}} \mathcal{D}^{\mathbf{g}} \\
 & & \quad \mathtt{if}\ \ \mathcal{D}om(\mathcal{C}^{\mathbf{g}}) \cap \mathcal{D}om(\mathcal{D}^{\mathbf{g}}) = \emptyset \\
(ToSubst) & [\mathcal{C}^{\mathbf{g}} \wedge \mathcal{D}]A & \rightarrow \quad [\mathcal{D}]\{\mathcal{C}^{\mathbf{g}}\}A
\end{array}
$$

The $\rho_{\mathtt{xC}}$-calculus

$$
\begin{array}{lll}
(GI) & X \ll A \wedge Y \ll B & \rightarrow \quad X \ll A \wedge_{\mathbf{g}} Y \ll B \\
 & & \quad \mathtt{if}\ \ X \neq Y \\
 & & \\
(Good) & \mathcal{C}^{\mathtt{g}} \wedge \mathcal{D}^{\mathtt{g}} & \rightarrow \quad \mathcal{C}^{\mathtt{g}} \wedge_{\mathbf{g}} \mathcal{D}^{\mathtt{g}} \\
 & & \quad \mathtt{if}\ \ \mathcal{D}om(\mathcal{C}^{\mathtt{g}}) \cap \mathcal{D}om(\mathcal{D}^{\mathtt{g}}) = \emptyset \\
(ToSubst) & [\mathcal{C}^{\mathtt{g}} \wedge \mathcal{D}]A & \rightarrow \quad [\mathcal{D}]\{\mathcal{C}^{\mathtt{g}}\}A \\
 & & \quad \mathtt{if}\ \ \mathcal{D}om(\mathcal{C}^{\mathtt{g}}) \cap \mathcal{D}om(\mathcal{D}) = \emptyset \\
(ToSubst) & [\mathcal{C}^{\mathtt{g}}]A & \rightarrow \quad \{\mathcal{C}^{\mathtt{g}}\}A
\end{array}
$$

The $\rho_{\mathtt{xC}}$-calculus

To deal with compositions: the $\rho_{\mathtt{xC}}$-calculus

$$(Compose) \quad \{\vartheta\}(\{\varphi\}A) \quad \rightarrow \quad \{\{\vartheta\}\varphi\}(\{\vartheta\}A)$$

The $\rho_{\mathtt{xC}}$-calculus

To deal with compositions: the $\rho_{\mathtt{xC}}$-calculus

$$(Compose) \quad \{\vartheta\}(\{\varphi\}A) \rightarrow \{\{\vartheta\}\varphi\}(\{\vartheta\}A)$$

$$(Compose) \quad \{\vartheta\}(\{\varphi\}A) \rightarrow \{\vartheta \wedge_{\mathbf{g}} \{\vartheta\}\varphi\}\ A$$

Properties of the ρ_{x}-calculus

- Well behaved properties for substitution application.
- Termination of $\longrightarrow_{\mathcal{C}}$.
- Confluence of the calculus.
- Conservativity
- Simulating the λ_x-calculus

Conclusion & Future work

Conclusion

- Pattern-matching calculi with explicit constraint handling. Modular: we can deal with arbitrary matching theories and arbitraries theories for the structure.

Conclusion & Future work

Conclusion

- Pattern-matching calculi with explicit constraint handling. Modular: we can deal with arbitrary matching theories and arbitraries theories for the structure.

- First Extention works on explicit substitutions (SML, λ-Prolog. . .)

Conclusion & Future work

Conclusion

- Pattern-matching calculi with explicit constraint handling. Modular: we can deal with arbitrary matching theories and arbitraries theories for the structure.
- First Extention works on explicit substitutions (SML, λ-Prolog. . .)
- Explicit constraint handling provides a nice framework to deal with errors.

Conclusion & Future work

Conclusion

- Pattern-matching calculi with explicit constraint handling. Modular: we can deal with arbitrary matching theories and arbitraries theories for the structure.
- First Extention works on explicit substitutions (SML, λ-Prolog. . .)
- Explicit constraint handling provides a nice framework to deal with errors.

Conclusion & Future work

Future work

- Named exceptions (thanks to labels for "bad" constraints)

Todays works

- Higher order unification/matching

Conclusion & Future work

Future work

- Named exceptions (thanks to labels for "bad" constraints)
- To deal with α-conversion.

Todays works

- Higher order unification/matching

Conclusion & Future work

Future work

- Named exceptions (thanks to labels for "bad" constraints)
- To deal with α-conversion.
- Implementations for the ρ-calculus.

Todays works

- Higher order unification/matching

From rewrite rules to constraints

(ρ)	$(A \twoheadrightarrow B)\ C$	$\rightarrow$	$(A \ll C)\ B$
(δ)	$(A, B)\ C$	$\rightarrow$	$A\ C, B\ C$

From constraints to substitutions

Decomposition

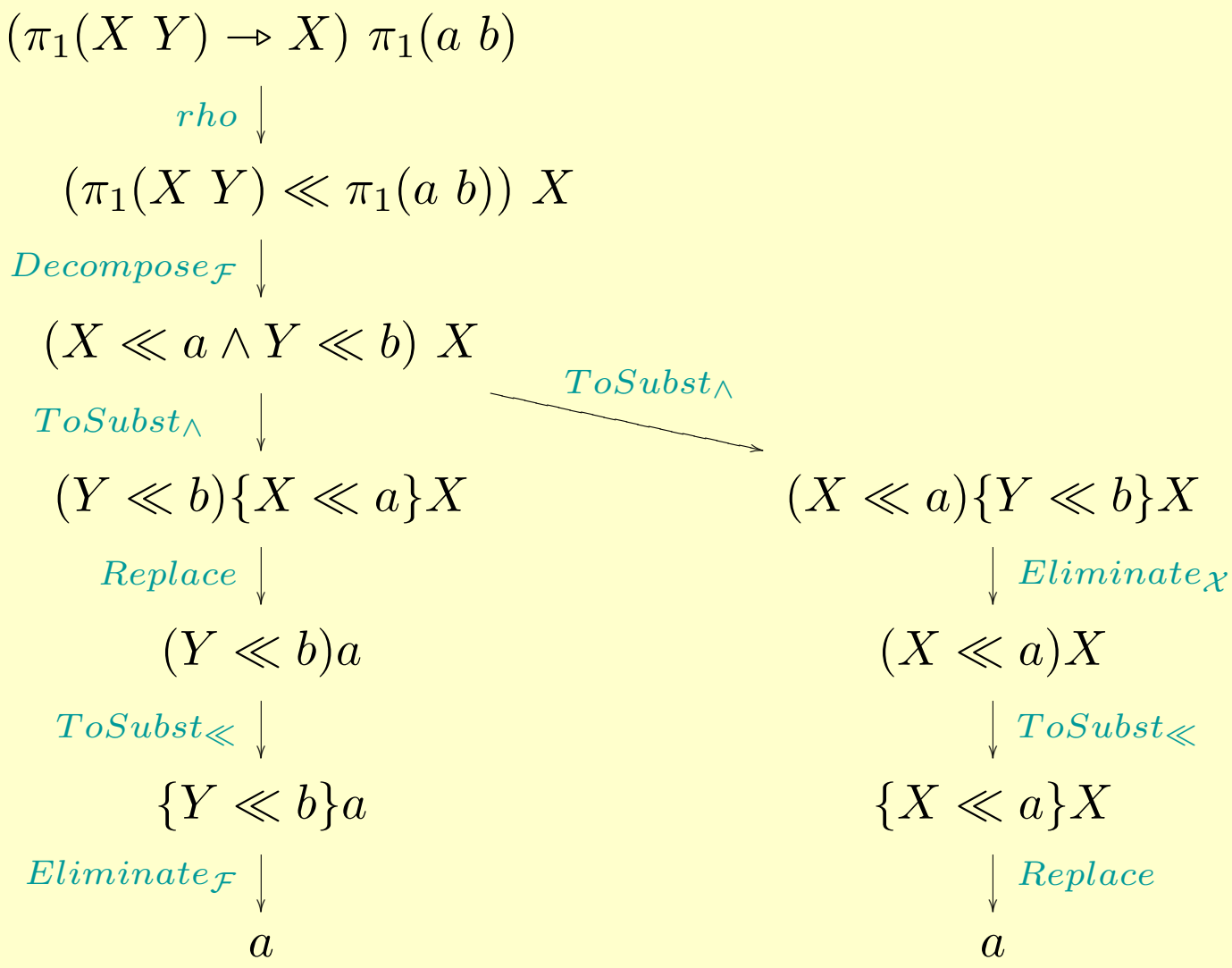
$(Decompose,)$	$A_1, A_2 \ll B_1, B_2$	$\rightarrow$	$A_1 \ll B_1 \wedge\ A_2 \ll B_2$
$(Decompose_{\mathcal{F}})$	$f(A_1 \ldots A_n) \ll f(B_1 \ldots B_n)$	$\rightarrow$	$A_1 \ll B_1 \wedge \ldots \wedge A_n \ll B_n$

From constraints to substitutions

$(ToSubst_{\wedge})$	$(X \ll A \wedge \mathcal{C})B$	$\rightarrow$	$(\mathcal{C})(\{X \ll A\}B)$
			$\texttt{if}\ X \notin \mathcal{D}om(\mathcal{C})$
$(ToSubst_{\ll})$	$(X \ll A)B$	$\rightarrow$	$\{X \ll A\}B$
$(ToSubst_{Id})$	$(a \ll a)B$	$\rightarrow$	B

Substitution applications

$(Replace)$	$\{X \ll A\}X$	$\rightarrow$	A
$(Eliminate_{\mathcal{X}})$	$\{X \ll A\}Y$	$\rightarrow$	Y
			$\texttt{if}\ X \neq Y$
$(Eliminate_{\mathcal{F}})$	$\{X \ll A\}f$	$\rightarrow$	f
$(Share_{;})$	$\{X \ll A\}(B \,;\, C)$	$\rightarrow$	$\{X \ll A\}B \,;\, \{X \ll A\}C$
$(Share_{\twoheadrightarrow})$	$\{X \ll A\}(B \twoheadrightarrow C)$	$\rightarrow$	$B \twoheadrightarrow \{X \ll A\}C$
$(Share_{\ll})$	$\{X \ll A\}(B \ll C)$	$\rightarrow$	$B \ll \{X \ll A\}C$
$(Share_{\wedge})$	$\{X \ll A\}(\mathcal{C} \wedge \mathcal{D})$	$\rightarrow$	$\{X \ll A\}\mathcal{C}\ \wedge\ \{X \ll A\}\mathcal{D}$


$(\pi_1(X\ Y) \rightarrowtriangle X)\ \pi_1(a\ b)$
rho
$(\pi_1(X\ Y) \ll \pi_1(a\ b))\ X$
$Decompose_{\mathcal{F}}$
$(X \ll a \wedge Y \ll b)\ X$
$ToSubst_{\wedge}$
$ToSubst_{\wedge}$
$(Y \ll b)\{X \ll a\}X$
$(X \ll a)\{Y \ll b\}X$
$Replace$
$Eliminate_{\mathcal{X}}$
$(Y \ll b)a$
$(X \ll a)X$
$ToSubst_{\ll}$
$ToSubst_{\ll}$
$\{Y \ll b\}a$
$\{X \ll a\}X$
$Eliminate_{\mathcal{F}}$
$Replace$
a
a

Example: application of a non-linear rewrite rule

$$(\mathrm{xor}(X\ X) \twoheadrightarrow \mathrm{ff})\ \mathrm{xor}(\mathrm{tt}\ \mathrm{tt})$$

$$
\begin{array}{ll}
\mapsto_{rho} & \big(\mathrm{xor}(X\ X) \ll \mathrm{xor}(\mathrm{tt}\ \mathrm{tt})\big)\ \mathrm{ff} \\
\mapsto_{Decompose_{\mathcal{F}}} & \big(X \ll \mathrm{tt} \wedge X \ll \mathrm{tt}\big)\ \mathrm{ff} = \big(X \ll \mathrm{tt}\big)\ \mathrm{ff} \quad \text{since } \wedge \text{ is idempotent} \\
\mapsto_{ToSubst_{\ll}} & \{X \ll \mathrm{tt}\}\mathrm{ff} \\
\mapsto_{Eliminate_{\mathcal{F}}} & \mathrm{ff}
\end{array}
$$

Properties of the ρ_{x}-calculus

- Well behaved properties for substitution application.
- Termination of $\longrightarrow_{\mathcal{C}}$.
- Confluence of the calculus.
- Conservativity
- Simulating the λ_x-calculus

Conclusion & Future work

Conclusion

- Pattern-matching calculi with explicit constraint handling. Modular: we can deal with arbitrary matching theories.

- Generalisation of the λ_x-calculus and the λ_σ-calculus

- Explicit constraint handling provide a nice framework to deal with errors.

Conclusion & Future work

Conclusion

- Pattern-matching calculi with explicit constraint handling. Modular: we can deal with arbitrary matching theories.

- Generalisation of the λ_x-calculus and the λ_σ-calculus

- Explicit constraint handling provide a nice framework to deal with errors.

Future and current work

- Composition of substitutions.

Conclusion & Future work

Conclusion

- Pattern-matching calculi with explicit constraint handling. Modular: we can deal with arbitrary matching theories.
- Generalisation of the λ_x-calculus and the λ_σ-calculus
- Explicit constraint handling provide a nice framework to deal with errors.

Future and current work

- Composition of substitutions.
- Named exceptions (thanks to labels for "bad" constraints)

Conclusion & Future work

Conclusion

- Pattern-matching calculi with explicit constraint handling. Modular: we can deal with arbitrary matching theories.
- Generalisation of the λ_x-calculus and the λ_σ-calculus
- Explicit constraint handling provide a nice framework to deal with errors.

Future and current work

- Composition of substitutions.
- Named exceptions (thanks to labels for "bad" constraints)

- Extension to deal with α-conversion.

- Extension to deal with α-conversion.
- Interpreter/Compiler for the ρ-calculus.