The Functional Machine Calculus

confluence via higher-order critical pairs

(Willem Heijltjes & Vincent van Oostrom)[1]

[1]Supported by EPSRC Project EP/R029121/1 Typed lambda-calculi with sharing and unsharing.

The Functional Machine Calculus (FMC)

Two independent modifications to the λ-calculus:

$$M, N \ ::= \quad x \quad | \quad M\,N \quad | \quad \lambda x.M$$

$$M, N \ ::= \ \star \ | \ x.\,M \ | \ [N]a.\,M \ | \ a\langle x\rangle.\,M$$

Sequencing	**Locations**
Split the variable into a unit $\star$ and a variable–with–continuation $x.\,M$	Parameterize abstraction and application in a set of locations $\mathcal{A}$
Encodes **strategies**	Encode **effects**

A simple stack machine/operational semantics (Landin, Krivine)

Stacks: $S \; ::= \; \varepsilon \mid S \cdot M$

$$\frac{}{(\varepsilon\,,\; M)} \quad \frac{(S\,,\; M\,N)}{(S \cdot N\,,\; M)} \quad \frac{(S \cdot N\,,\; \lambda x.\, M)}{(S\,,\; \{N/x\}M)} \quad \frac{(S\,,\; x)}{} \quad \frac{(\varepsilon\,,\; \lambda x.\, M)}{}$$

- **Application**: push
- **Abstraction**: pop and bind to local variable

The poly-λ-calculus

Parameterize application and abstraction in a set of locations $\mathcal{A}$

$$M,N \ ::= \ x \ \mid \ M\,N \ \mid \ \lambda x.M$$

$$M,N \ ::= \ x \ \mid \ [N]a.\,M \ \mid \ a\langle x\rangle.\,M \qquad (a \in \mathcal{A})$$

Embed λ-calculus by a reserved location $\lambda \in \mathcal{A}$ (may omit)

$$\lambda x.\,M \ = \ \lambda\langle x\rangle.\,M \ = \ \langle x\rangle.\,M \qquad M\,N \ = \ [N]\lambda.\,M \ = \ [N].\,M$$

Poly-stack machine/operational semantics

A memory S is a family of stacks: one for every location $a \in \mathcal{A}$.

$$S \;=\; \{S_a \mid a \in \mathcal{A}\} \;=\; S_{a_1}; S_{a_2}; \ldots; S_{a_n}$$

States are pairs (S, M), and transitions are:

$$\frac{(\, S \;;\; S_a \qquad , \; [N]a.\,M \,)}{(\, S \;;\; S_a \cdot N \;, \qquad M \,)} \qquad\qquad \frac{(\, S \;;\; S_a \cdot N \;, \; a\langle x\rangle.\,M \,)}{(\, S \;;\; S_a \qquad , \; \{N/x\}M \,)}$$

Encoding state

A memory cell is modelled by a location $c \in \mathcal{A}$

$$\begin{array}{lrcl} \text{update:} & c := N\,;\, M & = & c\langle\_\rangle.\,[N]c.\,M \\ \text{lookup:} & !c & = & c\langle x\rangle.\,[x]c.\,x \end{array}$$

$$c := N\,;M \;:\quad \frac{\dfrac{(\,S\,;\;\varepsilon_c\cdot P\;,\;c\langle\_\rangle.\,[N]c.\,M\,)}{(\,S\,;\;\varepsilon_c\;,\;[N]c.\,M\,)}}{(\,S\,;\;\varepsilon_c\cdot N\,,\;M\,)}$$

$$!c \;:\quad \frac{\dfrac{(\,S\,;\;\varepsilon_c\cdot N\,,\;c\langle x\rangle.\,[x]c.\,x\,)}{(\,S\,;\;\varepsilon_c\;,\;[N]c.\,N\,)}}{(\,S\,;\;\varepsilon_c\cdot N\,,\;N\,)}$$

β-Reduction

"Skips" stack actions on other locations:

$$[M]a.\,A_1 \ldots A_n.\,a\langle x\rangle.\,N \quad \longrightarrow \quad A_1 \ldots A_n.\,\{M/x\}N$$

where each A_i is an abstraction or application not on location a

Equivalently: "normal" β-reduction

$$[M]a.\,a\langle x\rangle.\,N \longrightarrow \{M/x\}N$$

modulo permutations

$$\begin{aligned}
[M]a.\,[N]b.\,P &\sim [N]b.\,[M]a.\,P \\
a\langle x\rangle.\,[N]b.\,P &\sim [N]b.\,a\langle x\rangle.\,P \quad \text{where } x \notin \mathrm{fv}(N) \\
a\langle x\rangle.\,b\langle y\rangle.\,P &\sim b\langle y\rangle.\,a\langle x\rangle.\,P
\end{aligned}$$

The Functional Machine Calculus

Split the variable into a unit $\star$ and a variable–with–continuation $x.\,M$

$$M, N \;::=\; x \;\mid\; MN \;\mid\; \lambda x.M$$

$$M, N \;::=\; \star \;\mid\; x.\,M \;\mid\; [N]a.\,M \;\mid\; a\langle x\rangle.\,M$$

Some example terms (the trailing $.\,\star$ will be omitted):

$\langle x\rangle.\,[x].\,[x]$ $\quad$ $\langle x\rangle$ $\quad$ $\langle x\rangle.\,[a\langle y\rangle.\,x.\,[y]a]$ $\quad$ $[\mathrm{rnd}\langle x\rangle.\,[x]\mathrm{out}].\,\langle f\rangle.\,f.\,f.\,f$

Nonsense as functions or λ-terms; fine as stack machine instructions!

Composition $N\,;M$ (or $N.\,M$) has unit $\star$ and is capture-avoiding.

$$
\begin{array}{rcll}
\star\,;M &=& M & \\
x.\,N\,;M &=& x.\,(N\,;M) & \\
[P]a.\,N\,;M &=& [P]a.\,(N\,;M) & \\
a\langle x\rangle.\,N\,;M &=& a\langle y\rangle.\,(\{y/x\}N\,;M) & (y \text{ fresh})
\end{array}
$$

Substitution uses composition for the variable case.

$$
\begin{array}{rcll}
\{M/x\}\star &=& \star & \\
\{M/x\}x.\,N &=& M\,;\{M/x\}N & \\
\{M/x\}y.\,N &=& y.\,\{M/x\}N & (x \neq y) \\
\{M/x\}[P]a.\,N &=& [\{M/x\}P]a.\,\{M/x\}N & \\
\{M/x\}a\langle x\rangle.\,N &=& a\langle x\rangle.\,N & \\
\{M/x\}a\langle y\rangle.\,N &=& a\langle z\rangle.\,\{M/x\}\{z/y\}N & (x \neq y, z \text{ fresh})
\end{array}
$$

β-Reduction skips abstractions and applications but not variables,

$$[M]a.\,A_1\,\ldots A_n.\,a\langle x\rangle.\,N \;\longrightarrow\; A_1\,\ldots A_n.\,\{M/x\}N$$

where each A_i is of the form $[P]b$ or $b\langle y\rangle$ with $a \neq b$.

Theorem

β-Reduction in the FMC is confluent.

A rewriter's perspective

dialogue

- P: can you say something about the FMC?

A rewriter's perspective

dialogue

- P: can you say something about the FMC?
- O: what are the objects A and what are the rules P?

A rewriter's perspective

dialogue

- P: can you say something about the FMC?
- O: what are the objects A and what are the rules P?
- P: FMC terms and a beta-rule

A rewriter’s perspective

dialogue

- P: can you say something about the FMC?
- O: what are the objects A and what are the rules P?
- P: FMC terms and a beta-rule
- O: what is the signature and what do you mean by a beta-rule?

A rewriter's perspective

dialogue

- P: can you say something about the FMC?
- O: what are the objects A and what are the rules P?
- P: FMC terms and a beta-rule
- O: what is the signature and what do you mean by a beta-rule?
- P: just as on the previous slides . . . ?

A rewriter's perspective

dialogue

- P: can you say something about the FMC?
- O: what are the objects *A* and what are the rules *P*?
- P: FMC terms and a beta-rule
- O: what is the signature and what do you mean by a beta-rule?
- P: just as on the previous slides . . . ?

analysis and synthesis

1. FMC terms by explicit grammar, with external notion of binding

A rewriter's perspective

dialogue

- P: can you say something about the FMC?
- O: what are the objects A and what are the rules P?
- P: FMC terms and a beta-rule
- O: what is the signature and what do you mean by a beta-rule?
- P: just as on the previous slides . . . ?

analysis and synthesis

1. FMC terms by explicit grammar, with external notion of binding

A rewriter's perspective

dialogue

- P: can you say something about the FMC?
- O: what are the objects A and what are the rules P?
- P: FMC terms and a beta-rule
- O: what is the signature and what do you mean by a beta-rule?
- P: just as on the previous slides . . . ?

analysis and synthesis

1. FMC terms by explicit grammar, with external notion of binding
   $\Rightarrow$ terms as λ-terms over simply typed signature, with λ-binding

A rewriter's perspective

dialogue

- P: can you say something about the FMC?
- O: what are the objects A and what are the rules P?
- P: FMC terms and a beta-rule
- O: what is the signature and what do you mean by a beta-rule?
- P: just as on the previous slides . . . ?

analysis and synthesis

1. FMC terms by explicit grammar, with external notion of binding
   $\Rightarrow$ terms as λ-terms over simply typed signature, with λ-binding
2. FMC open rule schema employing meta-level variables and substitution

A rewriter's perspective

dialogue

- P: can you say something about the FMC?
- O: what are the objects A and what are the rules P?
- P: FMC terms and a beta-rule
- O: what is the signature and what do you mean by a beta-rule?
- P: just as on the previous slides . . . ?

analysis and synthesis

1. FMC terms by explicit grammar, with external notion of binding
   $\Rightarrow$ terms as λ-terms over simply typed signature, with λ-binding
2. FMC open rule schema employing meta-level variables and substitution
   $\Rightarrow$ closed rule schema employing object-level variables and substitution

Higher-order term rewrite systems (Nipkow)

arbitrary signatures

combinatory logic (CL) : term rewrite system (TRS)

$\vdots$ $\vdots$

lambda-calculus (lambda) : higher-order term rewrite system (PRS)

closed under renaming, adding recursion / algebraic rules, etc.

freeness: signature $\Longrightarrow$ terms

simply typed λ-terms modulo $\alpha\beta\eta$ over simply typed signature

implicit grammar (term : simply typed λ-term in long $\beta\eta$-normal form)
internal notion of binding (λ-abstraction)

Higher-order term rewrite systems

Example (addition TRS as a PRS)

- signature $0 : o$ (nullary), $S : o \to o$ (unary), $A : o \to o \to o$ (binary)

Higher-order term rewrite systems

Example (addition TRS as a PRS)

- signature $0 : o$ (nullary), $S : o \to o$ (unary), $A : o \to o \to o$ (binary)
- rules $\rho : o \to o$ and $\theta : o \to o \to o$, for variables $x, y : o$:

$$\begin{array}{lcl} \rho: \quad \lambda x.A\,x\,0 & \to & \lambda x.x \\ \theta: \lambda xy.A\,x\,(S\,y) & \to & \lambda xy.S\,(A\,x\,y) \end{array}$$

cf. Frege's shift from $\forall x.(t = s)$ to $(\lambda x.t) = (\lambda x.s)$

Higher-order term rewrite systems

Example (addition TRS as a PRS)

- signature $0 : o$ (nullary), $S : o \to o$ (unary), $A : o \to o \to o$ (binary)
- rules $\rho : o \to o$ and $\theta : o \to o \to o$, for variables $x, y : o$

$$
\begin{array}{lrcl}
\rho: & \lambda x.A(x,0) & \to & \lambda x.x \\
\theta: & \lambda xy.A(x,S(y)) & \to & \lambda xy.S(A(x,y))
\end{array}
$$

with syntactic sugar added

Higher-order term rewrite systems

Example (untyped lambda-beta-eta calculus as a PRS $\mathcal{L}am$)

- signature $\mathsf{abs} : (o \to o) \to o$, $\mathsf{app} : o \to o \to o$

Higher-order term rewrite systems

Example (untyped lambda-beta-eta calculus as a PRS $\mathcal{L}am$)

- signature abs : $(o \rightarrow o) \rightarrow o$, app : $o \rightarrow o \rightarrow o$
- rules eta : $o \rightarrow o$, beta : $(o \rightarrow o) \rightarrow o \rightarrow o$, variables $M : o \rightarrow o$ and $N, K : o$

$$\begin{array}{rrcl} \text{eta:} & \lambda K.\mathsf{abs}\,\lambda x.\mathsf{app}\,K\,x & \rightarrow & \lambda K.K \\ \text{beta:} & \lambda MN.\mathsf{app}\,(\mathsf{abs}\,\lambda x.M\,x)\,N & \rightarrow & \lambda MN.M\,N \end{array}$$

without syntactic sugar; x is parameter to M; K no parameters

Higher-order term rewrite systems

Example (untyped lambda-beta-eta calculus as a PRS $\mathcal{L}am$)

- signature abs $: (o \to o) \to o$, app $: o \to o \to o$
- rules eta $: o \to o$, beta $: (o \to o) \to o \to o$, variables $M : o \to o$ and $N, K : o$

$$
\begin{array}{rrcl}
\text{eta:} & \lambda K.\text{abs}(\lambda x.\text{app}(K, x)) & \to & \lambda K.K \\
\text{beta:} & \lambda MN.\text{app}(\text{abs}(\lambda x.M(x)), N) & \to & \lambda MN.M(N)
\end{array}
$$

with syntactic sugar

Higher-order term rewrite systems

Example (untyped lambda-beta-eta calculus as a PRS $\mathcal{L}am$)

- signature abs : $(o \to o) \to o$, app : $o \to o \to o$
- rules eta : $o \to o$, beta : $(o \to o) \to o \to o$, variables $M : o \to o$ and $N, K : o$

$$\begin{array}{rrcl} \text{eta:} & \lambda K.\text{abs}(\lambda x.\text{app}(K, x)) & \to & \lambda K.K \\ \text{beta:} & \lambda MN.\text{app}(\text{abs}(\lambda x.M(x)), N) & \to & \lambda MN.M(N) \end{array}$$

with syntactic sugar

embedding

PRS $\mathcal{L}am$ 2^{nd}-order since abs : $(o \to o) \to o$.

Higher-order term rewrite systems

Example (untyped lambda-beta-eta calculus as a PRS $\mathcal{L}am$)

- signature abs : $(o \to o) \to o$, app : $o \to o \to o$
- rules eta : $o \to o$, beta : $(o \to o) \to o \to o$, variables $M : o \to o$ and $N, K : o$

$$\begin{array}{rrcl} \text{eta:} & \lambda K.\text{abs}(\lambda x.\text{app}(K, x)) & \to & \lambda K.K \\ \text{beta:} & \lambda MN.\text{app}(\text{abs}(\lambda x.M(x)), N) & \to & \lambda MN.M(N) \end{array}$$

with syntactic sugar

embedding

PRS $\mathcal{L}am$ 2^{nd}-order since abs : $(o \to o) \to o$. untyped lambda-calculus embedded in fragment: all variables of type o.

Higher-order term rewrite systems

Example (untyped lambda-beta-eta calculus as a PRS $\mathcal{L}am$)

- signature abs : $(o \to o) \to o$, app : $o \to o \to o$
- rules eta : $o \to o$, beta : $(o \to o) \to o \to o$, variables $M : o \to o$ and $N, K : o$

$$\begin{array}{rcl} \text{eta:} \quad \lambda K.\mathrm{abs}(\lambda x.\mathrm{app}(K, x)) & \to & \lambda K.K \\ \text{beta:} \lambda MN.\mathrm{app}(\mathrm{abs}(\lambda x.M(x)), N) & \to & \lambda MN.M(N) \end{array}$$

with syntactic sugar

embedding

PRS $\mathcal{L}am$ 2^{nd}-order since abs : $(o \to o) \to o$. untyped lambda-calculus embedded in fragment: all variables of type o. $\mathcal{L}am$ orthogonal $\Longrightarrow$ fragment confluent.

FMC substitution

FMC substitution based on auxiliary notion of composition

Composition $N\,;M$ (or $N.\,M$) has unit $\star$ and is capture-avoiding.

$$
\begin{aligned}
\star\;;M &= M \\
x.\,N\,;M &= x.\,(N\,;M) \\
[P]a.\,N\,;M &= [P]a.\,(N\,;M) \\
a\langle x\rangle.\,N\,;M &= a\langle y\rangle.\,(\{y/x\}N\,;M) \quad (y \text{ fresh})
\end{aligned}
$$

Substitution uses composition for the variable case.

$$
\begin{aligned}
\{M/x\}\star &= \star \\
\{M/x\}x.\,N &= M\,;\{M/x\}N \\
\{M/x\}y.\,N &= y.\,\{M/x\}N \quad (x \neq y) \\
\{M/x\}[P]a.\,N &= [\{M/x\}P]a.\,\{M/x\}N \\
\{M/x\}a\langle x\rangle.\,N &= a\langle x\rangle.\,N \\
\{M/x\}a\langle y\rangle.\,N &= a\langle z\rangle.\,\{M/x\}\{z/y\}N \quad (x \neq y, z \text{ fresh})
\end{aligned}
$$

FMC substitution

FMC substitution based on auxiliary notion of composition

Composition $N\,;M$ (or $N.\,M$) has unit $\star$ and is capture-avoiding.

$$\begin{aligned}
\star\;;M &= M \\
x.\,N\,;M &= x.\,(N\,;M) \\
[P]a.\,N\,;M &= [P]a.\,(N\,;M) \\
a\langle x\rangle.\,N\,;M &= a\langle y\rangle.\,(\{y/x\}N\,;M) \quad (y \text{ fresh})
\end{aligned}$$

Substitution uses composition for the variable case.

$$\begin{aligned}
\{M/x\}\star &= \star \\
\{M/x\}x.\,N &= M\,;\{M/x\}N \\
\{M/x\}y.\,N &= y.\,\{M/x\}N \quad (x \neq y) \\
\{M/x\}[P]a.\,N &= [\{M/x\}P]a.\,\{M/x\}N \\
\{M/x\}a\langle x\rangle.\,N &= a\langle x\rangle.\,N \\
\{M/x\}a\langle y\rangle.\,N &= a\langle z\rangle.\,\{M/x\}\{z/y\}N \quad (x \neq y, z \text{ fresh})
\end{aligned}$$

composition $A_1 \ldots A_n.\star\,;M$ is substitution of M for $\star$

FMC substitution

FMC substitution based on auxiliary notion of composition

Composition $N\,;M$ (or $N.\,M$) has unit $\star$ and is capture-avoiding.

$$
\begin{aligned}
\star\,;M &= M \\
x.\,N\,;M &= x.\,(N\,;M) \\
[P]a.\,N\,;M &= [P]a.\,(N\,;M) \\
a\langle x\rangle.\,N\,;M &= a\langle y\rangle.\,(\{y/x\}N\,;M) \quad (y \text{ fresh})
\end{aligned}
$$

Substitution uses composition for the variable case.

$$
\begin{aligned}
\{M/x\}\star &= \star \\
\{M/x\}x.\,N &= M\,;\{M/x\}N \\
\{M/x\}y.\,N &= y.\,\{M/x\}N \quad (x \neq y) \\
\{M/x\}[P]a.\,N &= [\{M/x\}P]a.\,\{M/x\}N \\
\{M/x\}a\langle x\rangle.\,N &= a\langle x\rangle.\,N \\
\{M/x\}a\langle y\rangle.\,N &= a\langle z\rangle.\,\{M/x\}\{z/y\}N \quad (x \neq y, z \text{ fresh})
\end{aligned}
$$

represent $A_1 \ldots A_n.\star$ as $\lambda\,\chi.A_1 \ldots A_n.\chi$ for bound variable χ

FMC substitution

FMC substitution based on auxiliary notion of composition

Composition $N\,;M$ (or $N.\,M$) has unit $\star$ and is capture-avoiding.

$$\begin{aligned}
\star\,;M &= M \\
x.\,N\,;M &= x.\,(N\,;M) \\
[P]a.\,N\,;M &= [P]a.\,(N\,;M) \\
a\langle x\rangle.\,N\,;M &= a\langle y\rangle.\,(\{y/x\}N\,;M) \quad (y \text{ fresh})
\end{aligned}$$

Substitution uses composition for the variable case.

$$\begin{aligned}
\{M/x\}\star &= \star \\
\{M/x\}x.\,N &= M\,;\{M/x\}N \\
\{M/x\}y.\,N &= y.\,\{M/x\}N \quad (x \neq y) \\
\{M/x\}[P]a.\,N &= [\{M/x\}P]a.\,\{M/x\}N \\
\{M/x\}a\langle x\rangle.\,N &= a\langle x\rangle.\,N \\
\{M/x\}a\langle y\rangle.\,N &= a\langle z\rangle.\,\{M/x\}\{z/y\}N \quad (x \neq y, z \text{ fresh})
\end{aligned}$$

represent $A_1 \ldots A_n.\star$ as $\lambda\chi.A_1 \ldots A_n.\chi$ for variable χ; $x.\,N$ as application $x\,N$

FMC substitution

FMC substitution based on auxiliary notion of composition

Composition $N\,;M$ (or $N.\,M$) has unit $\star$ and is capture-avoiding.

$$
\begin{aligned}
\star\,;M &= M \\
x.\,N\,;M &= x.\,(N\,;M) \\
[P]a.\,N\,;M &= [P]a.\,(N\,;M) \\
a\langle x\rangle.\,N\,;M &= a\langle y\rangle.\,(\{y/x\}N\,;M) \quad (y \text{ fresh})
\end{aligned}
$$

Substitution uses composition for the variable case.

$$
\begin{aligned}
\{M/x\}\star &= \star \\
\{M/x\}x.\,N &= M\,;\{M/x\}N \\
\{M/x\}y.\,N &= y.\,\{M/x\}N \quad (x \neq y) \\
\{M/x\}[P]a.\,N &= [\{M/x\}P]a.\,\{M/x\}N \\
\{M/x\}a\langle x\rangle.\,N &= a\langle x\rangle.\,N \\
\{M/x\}a\langle y\rangle.\,N &= a\langle z\rangle.\,\{M/x\}\{z/y\}N \quad (x \neq y, z \text{ fresh})
\end{aligned}
$$

represent $A_1 \ldots A_n.\star$ as $\lambda\chi.A_1 \ldots A_n.\chi$ for variable χ; increases order!

Higher-order term rewrite systems

Example (the FMC as a PRS $\mathcal{FMC}$)

- signature $\mathsf{abs}_a : ((o \to o) \to o) \to o$, $\mathsf{app}_a : o \to (o \to o) \to o$ for every a

Higher-order term rewrite systems

Example (the FMC as a PRS $\mathcal{FMC}$)

- signature $\mathsf{abs}_a : ((o \to o) \to o) \to o$, $\mathsf{app}_a : o \to (o \to o) \to o$ for every a
- rule schema $\mathsf{beta}_H : \ldots$ for variables N, $\vec{x}$, and x of type $o \to o$ given by:

$$\mathsf{beta}_{a,H}{:}\lambda M\vec{P}N.\mathsf{app}_a(H[\mathsf{abs}_a(\lambda x.M(\vec{x},x))],N) \quad \to \quad \lambda M\vec{P}N.H[M(\vec{x},N)]\lambda MN.M\,N$$

contexts H, given for locations $b \neq a$ by:

$$H \quad ::= \quad \Box \mid \mathsf{app}_b(H,P(\vec{x})) \mid \mathsf{abs}_b(\lambda x.H)$$

$P \in \vec{P}$ fresh variable with as parameters variables bound above

Higher-order term rewrite systems

embedding

$\mathcal{FMC}$ 3rd-order since $\mathsf{abs}_a : ((o \to o) \to o) \to o$

Higher-order term rewrite systems

embedding

FMC embedded by map $\langle\,\rangle$ in fragment $\lambda\chi.S$ of $\mathcal{FMC}$ with

$$S ::= \chi \mid x\,S \mid \mathsf{app}_a(S, \lambda\chi.S) \mid \mathsf{abs}_a(\lambda x.S)$$

- $\star$ maps to χ
- $x.\,M$ maps to $x\,\langle M\rangle$
- $[N]a.\,M$ maps to $\mathsf{app}_a(\langle M\rangle, \lambda\chi.\langle N\rangle)$
- $a\langle x\rangle.\,M$ maps to $\mathsf{abs}_a(\lambda x.\langle M\rangle)$

Higher-order term rewrite systems

embedding

$\mathcal{FMC}$ not orthogonal; (schematic) self-overlaps:

$$\mathsf{app}_a\text{–}\mathsf{app}_b\text{–}\mathsf{abs}_b\text{–}\mathsf{abs}_a$$

$$\mathsf{app}_b\text{–}\mathsf{app}_a\text{–}\mathsf{abs}_b\text{–}\mathsf{abs}_a$$

$\Longrightarrow$ why confluent?

Higher-order term rewrite systems

embedding

$\mathcal{FMC}$ not orthogonal; (schematic) self-overlaps:

$$\mathsf{app}_a\text{–}\mathsf{app}_b\text{–}\mathsf{abs}_b\text{–}\mathsf{abs}_a$$

$$\mathsf{app}_b\text{–}\mathsf{app}_a\text{–}\mathsf{abs}_b\text{–}\mathsf{abs}_a$$

$\Longrightarrow$ why confluent? because overlaps are harmless

Higher-order term rewrite systems

embedding

$\mathcal{FMC}$ not orthogonal; (schematic) self-overlaps:

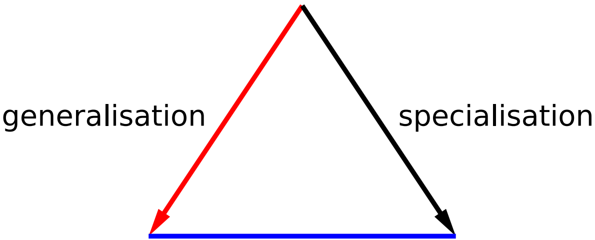
$$\mathsf{app}_a\text{–}\mathsf{app}_b\text{–}\mathsf{abs}_b\text{–}\mathsf{abs}_a$$

$$\mathsf{app}_b\text{–}\mathsf{app}_a\text{–}\mathsf{abs}_b\text{–}\mathsf{abs}_a$$

$\Longrightarrow$ why confluent? because overlaps are harmless
to express this we need formal notions of overlap, critical peak, step, ...

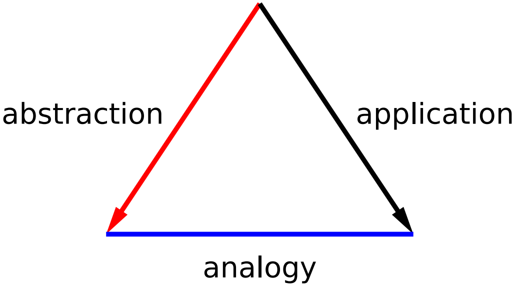
Steps as terms over signature + rules

Pólya's triangle



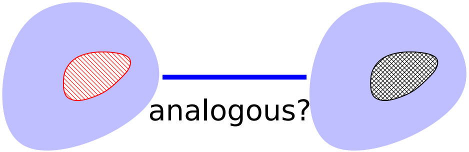
Steps as terms over signature + rules

Pólya's triangle

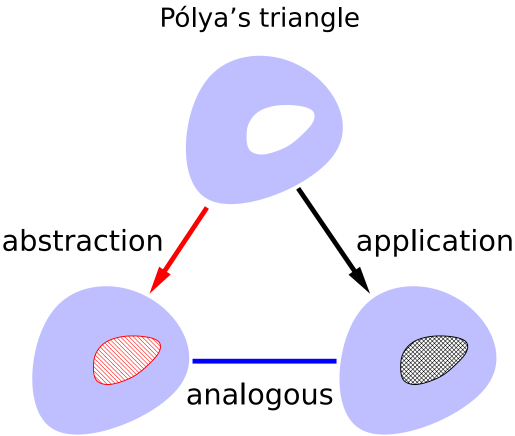


Steps as terms over signature + rules

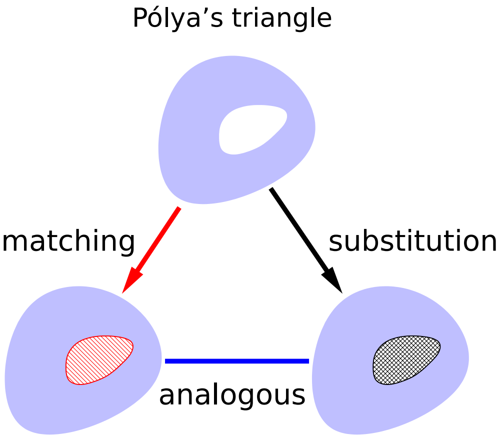
Pólya's triangle



Steps as terms over signature + rules

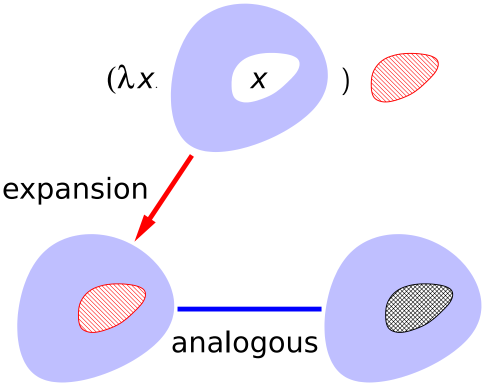


Steps as terms over signature + rules



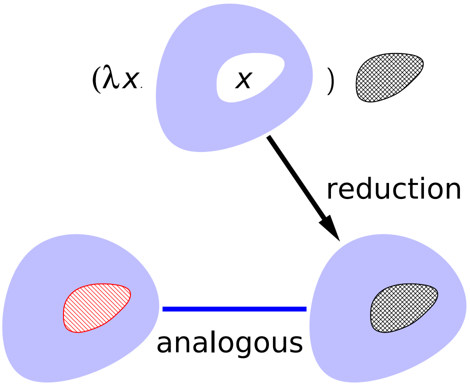
Steps as terms over signature + rules

matching–replacement–substitution rule $\rho : \ell \rightarrow r$ (& van Raamsdonk 1994)



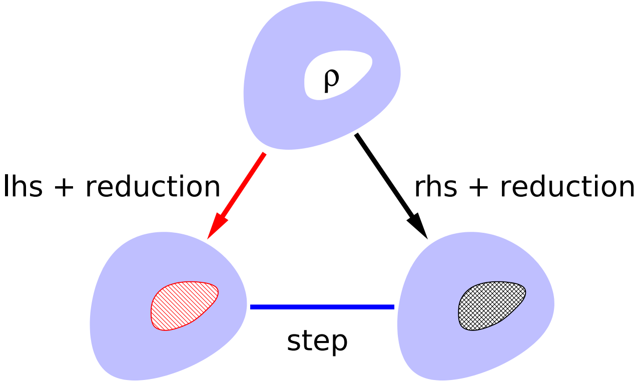
Steps as terms over signature + rules

matching–replacement–substitution rule ρ : $\color{red}{\ell} \to r$



Steps as terms over signature + rules

structured rewrite step for rule $\rho : \ell \to r$ (Terese ⩔ 2003)



Steps as terms over signature + rules

Example (Steps in HRS for addition)

$$\rho\colon \quad \lambda x.A(x,0) \to \lambda x.x$$
$$\theta\colon \lambda xy.A(x,S(y)) \to \lambda xy.S(A(x,y))$$

- $S(\rho(0))$ step from $S((\lambda x.A(x,0))\,0){\downarrow} = S(A(0,0))$ to $S((\lambda x.x)\,0){\downarrow} = S(0)$

Steps as terms over signature + rules

Example (Steps in HRS for addition)

$$\rho\colon \quad \lambda x.A(x,0) \;\to\; \lambda x.x$$
$$\theta\colon \lambda xy.A(x,S(y)) \;\to\; \lambda xy.S(A(x,y))$$

- $S(\rho(0))$ step from $S((\lambda x.A(x,0))\,0){\downarrow} = S(A(0,0))$ to $S((\lambda x.x)\,0){\downarrow} = S(0)$
- $\rho(\theta(0,0))$ multistep from $(\lambda x.A(x,0))\,((\lambda xy.A(x,S(y)))\,0\,0){\downarrow} = A(A(0,S(0)),0)$ to $(\lambda x.x)((\lambda xy.S(A(x,y)))\,0\,0){\downarrow} = S(A(0,0))$

Steps as terms over signature + rules

Example (Steps in HRS for addition)

$$\rho: \quad \lambda x.A(x,0) \to \lambda x.x$$
$$\theta: \lambda xy.A(x,S(y)) \to \lambda xy.S(A(x,y))$$

- $S(\rho(0))$ step from $S((\lambda x.A(x,0))\,0)\downarrow = S(A(0,0))$ to $S((\lambda x.x)\,0)\downarrow = S(0)$
- $\rho(\theta(0,0))$ multistep from $(\lambda x.A(x,0))\,((\lambda xy.A(x,S(y)))\,0\,0)\downarrow = A(A(0,S(0)),0)$ to $(\lambda x.x)((\lambda xy.S(A(x,y)))\,0\,0)\downarrow = S(A(0,0))$

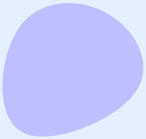
freeness: signature + rules $\Longrightarrow$ multisteps

multistep $\multimap$: simply typed λ-term modulo $\alpha\beta\eta$ over typed signature & rules
source (target) by mapping each rule $\rho : \ell \to r$ in multistep to lhs ℓ (rhs r)
step $\to$ is $\multimap$ restricted to multisteps having one rule (symbol)

Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Idea: allow to carve out well-behaved part, pat $\Longleftrightarrow$ pattern

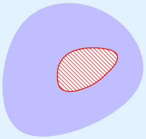
given a term



Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Idea: allow to carve out well-behaved part, pat $\iff$ pattern

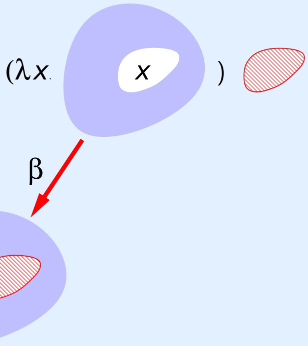
select convex set of edges and nodes, a pat P (geometric)



Geometric ⇔ inductive pattern, to define overlap

Idea: allow to carve out well-behaved part, pat $\Longleftrightarrow$ pattern

β-expand to occurrence of pattern π (inductive)



Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Definition (pat; geometric)

non-empty set P of positions in tree of λ-term.

- (convex) if $p, q \in P$ then positions on path between p and q in P;

Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Definition (pat; geometric)

non-empty set P of positions in tree of λ-term.

- (convex) if $p, q \in P$ then positions on path between p and q in P;
- (rigid) if $t(p)$ is variable and $p \in P$, then bound by λ-abstraction at P-position

Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Definition (pat; geometric)

non-empty set P of positions in tree of λ-term.

- (convex) if $p, q \in P$ then positions on path between p and q in P;
- (rigid) if $t(p)$ is variable and $p \in P$, then bound by λ-abstraction at P-position
- (base-fringe) $t_{|p}$ of base type for p root of P or a child not in P of P-position

Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Definition (pat; geometric)

non-empty set P of positions in tree of λ-term.

- (convex) if $p, q \in P$ then positions on path between p and q in P;
- (rigid) if $t(p)$ is variable and $p \in P$, then bound by λ-abstraction at P-position
- (base-fringe) $t_{|p}$ of base type for p root of P or a child not in P of P-position
- (normal) if $t(p)$ is an application and $p \in P$, then left child not λ-position

multipat vector $\vec{P}$ of pairwise disjoint pats

Example

$\{\varepsilon, 1, 11, 12, 121, 122\}$ is pat in $\Delta := \mathrm{app}(\mathrm{abs}(\lambda y.\mathrm{app}(y, y)), \mathrm{abs}(\lambda z.\mathrm{app}(z, z)))$

Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Definition (multipattern occurrence; inductive)

positional pattern π is closed term of shape $\lambda \vec{F}.f(\vec{t})$

Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Definition (multipattern occurrence; inductive)

positional pattern π is closed term of shape $\lambda\vec{F}.f(\vec{t})$ that is

- (head-defined) f function symbol and $f(\vec{t})$ of base type

Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Definition (multipattern occurrence; inductive)

positional pattern π is closed term of shape $\lambda\vec{F}.f(\vec{t})$ that is

- (head-defined) f function symbol and $f(\vec{t})$ of base type
- (linear) π linear in $\vec{F}$, each F_i occurs once in $f(\vec{t})$, left-to-right

Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Definition (multipattern occurrence; inductive)

positional pattern π is closed term of shape $\lambda\vec{F}.f(\vec{t})$ that is

- (head-defined) f function symbol and $f(\vec{t})$ of base type
- (linear) π linear in $\vec{F}$, each F_i occurs once in $f(\vec{t})$, left-to-right
- (fully-extended) each $F \in \vec{F}$ occurs in π as $F(\vec{x})$ with $\vec{x}$ the outside-in list of (η-expansions of) variables bound above F in $f(\vec{t})$

Geometric $\Leftrightarrow$ inductive pattern, to define overlap

Definition (multipattern occurrence; inductive)

positional pattern π is closed term of shape $\lambda\vec{F}.f(\vec{t})$ that is

- (head-defined) f function symbol and $f(\vec{t})$ of base type
- (linear) π linear in $\vec{F}$, each F_i occurs once in $f(\vec{t})$, left-to-right
- (fully-extended) each $F \in \vec{F}$ occurs in π as $F(\vec{x})$ with $\vec{x}$ the outside-in list of (η-expansions of) variables bound above F in $f(\vec{t})$

$(\lambda\vec{G}.s)\,\vec{\pi}$ multipattern occurrence of $\vec{\pi}$ in $\overline{(\lambda\vec{G}.s)\,\vec{\pi}}$ if s linear in $\vec{G}$, up to permutation of $\vec{\pi}$, $\vec{F}$ (overlining : reduce β-redex and recursively created ones)

Example

lhs $\lambda FS.\mathsf{app}(\mathsf{abs}(\lambda x.F(x)),S)$ of rule beta of $\mathcal{L}am$ is a positional pattern occurring in Δ because $\Delta = \overline{(\lambda G.G(\lambda y.\mathsf{app}(y,y),\mathsf{abs}(\lambda z.\mathsf{app}(z,z))))\,\mathrm{lhs}}$

Geometric vs. inductive patterns

Theorem (geometric vs. inductive)

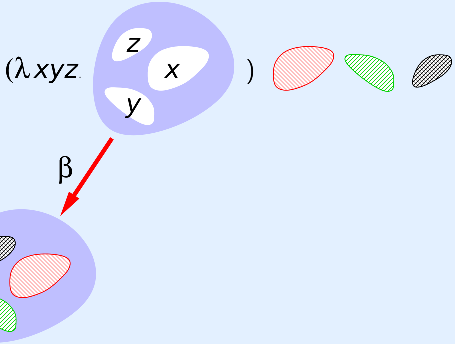
given a λ-term isomorphism between

- *multipats and multipattern-occurrences*

Geometric vs. inductive patterns

isomorphism in a picture

internal positions of patterns bijectively trace to pats



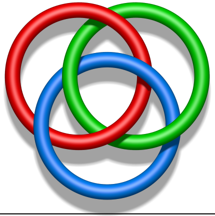
Geometric vs. inductive patterns

Theorem (geometric vs. inductive)

given a λ-term isomorphism between

- *multipats and multipattern-occurrences*

no 3 pairwise disjoint pats not expandable to triplepattern-occurrence (1994)



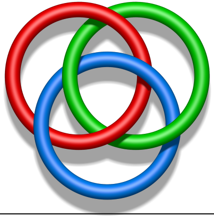
Geometric vs. inductive patterns

Theorem (geometric vs. inductive)

given a λ-term isomorphism between

- *multipats and multipattern-occurrences*

no Borromean rings



Geometric vs. inductive patterns

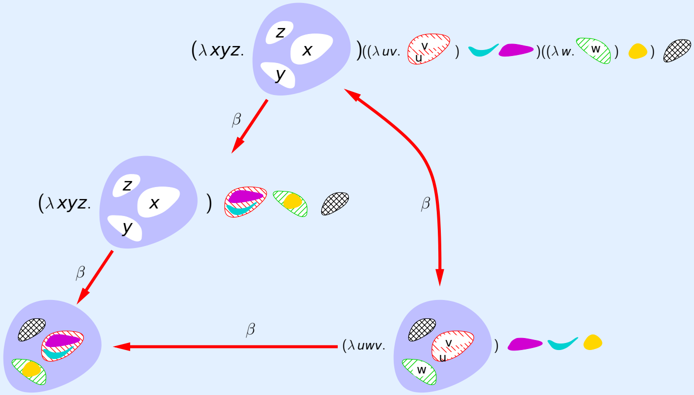
Theorem (geometric vs. inductive)

given a λ-term isomorphism between

- *multipats and multipattern-occurrences*
- *refinement of multipats and multipattern-occurrences*

Geometric vs. inductive patterns

refinement isomorphism in a picture



Geometric vs. inductive patterns

Theorem (geometric vs. inductive)

given a λ-term isomorphism between

- *multipats and multipattern-occurrences*
- *refinement of multipats and multipattern-occurrences*
- *refinement is finite distributive lattice (closed under union, intersection)*

Geometric vs. inductive patterns

Theorem (geometric vs. inductive)

given a λ-term isomorphism between

- *multipats and multipattern-occurrences*
- *refinement of multipats and multipattern-occurrences*
- *refinement is finite distributive lattice*

upshots

- redex-patterns orthogonal because there is a multipattern containing them (& van Raamsdonk 1994)

Geometric vs. inductive patterns

Theorem (geometric vs. inductive)

given a λ-term isomorphism between

- *multipats and multipattern-occurrences*
- *refinement of multipats and multipattern-occurrences*
- *refinement is finite distributive lattice*

upshots

- redex-patterns orthogonal because there is a multipattern containing them
- redex-patterns overlapping because their pats are (have non-empty intersection) (Hirokawa et al. 2019)

Geometric vs. inductive patterns

Theorem (geometric vs. inductive)

given a λ-term isomorphism between

- *multipats and multipattern-occurrences*
- *refinement of multipats and multipattern-occurrences*
- *refinement is finite distributive lattice*

upshots

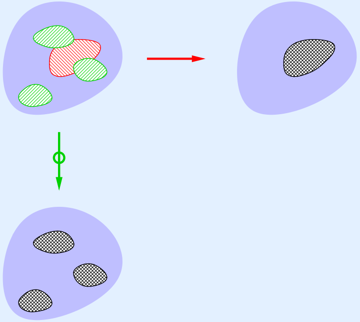
- redex-patterns orthogonal because there is a multipattern containing them
- redex-patterns overlapping because their pats are
- peak is critical if union of pats is the whole source (definition!)

A multi–one critical peak criterion; for TRSs (Okui 1998)

Theorem

$\to$ *is confluent if* $\forall$ *critical multi–one peaks* $b \mathrel{\leftarrow\!\circ\!-} a \to c$, $\exists\, b \twoheadrightarrow d \mathrel{\leftarrow\!\circ\!-} c$

Geometric proof (proof by potatoes).



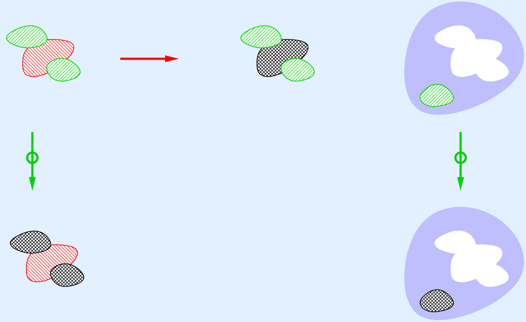
$\square$

A multi–one critical peak criterion

Theorem

$\to$ is confluent if $\forall$ critical multi–one peaks $b \mathrel{\leftarrow\!\!\circ\!\!-} a \to c$, $\exists\, b \twoheadrightarrow d \mathrel{\leftarrow\!\!\circ\!\!-} c$

Geometric proof.



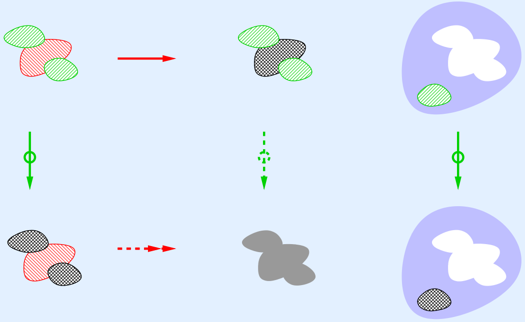
$\square$

A multi–one critical peak criterion

Theorem

$\to$ is confluent if $\forall$ critical multi–one peaks $b \leftarrow\!\!\circ\!\!- a \to c$, $\exists\, b \twoheadrightarrow d \leftarrow\!\!\circ\!\!- c$

Geometric proof.



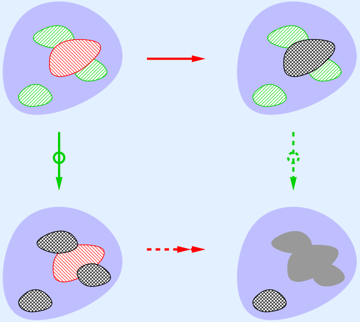
$\square$

A multi–one critical peak criterion

Theorem

$\to$ *is confluent if* $\forall$ *critical multi–one peaks* $b \leftarrow\!\!\circ\!\!- a \to c$, $\exists\, b \twoheadrightarrow d \leftarrow\!\!\circ\!\!- c$

Geometric proof.



$\square$

A multi–one critical peak criterion

Theorem

$\rightarrow$ is confluent if $\forall$ critical multi–one peaks $b \mathrel{\leftarrow\!\!\circ\!\!-} a \rightarrow c$, $\exists$ $b \twoheadrightarrow d \mathrel{\leftarrow\!\!\circ\!\!-} c$

Inductive proof.

- any overlapping multi–one peak $t \mathrel{\leftarrow\!\!\circ\!\!-} s \rightarrow r$
- decomposes as $(\lambda x.D)\,\hat{t} \mathrel{\leftarrow\!\!\circ\!\!-} (\lambda x.C)\,\hat{s} \rightarrow (\lambda x.C)\,\hat{r}$
  for multi–one critical peak $\hat{t} \mathrel{\leftarrow\!\!\circ\!\!-} \hat{s} \rightarrow \hat{r}$ and multistep $D \mathrel{\leftarrow\!\!\circ\!\!-} C$
- for multi–one critical peak $\hat{t} \mathrel{\leftarrow\!\!\circ\!\!-} \hat{s} \rightarrow \hat{r}$ exists many–multi valley $\hat{t} \twoheadrightarrow \hat{u} \mathrel{\leftarrow\!\!\circ\!\!-} \hat{r}$
- recomposing with multistep $D \mathrel{\leftarrow\!\!\circ\!\!-} C$ yields many–multi valley
  $(\lambda x.D)\,\hat{t} \rightarrow (\lambda x.D)\,\hat{u} \mathrel{\leftarrow\!\!\circ\!\!-} (\lambda x.C)\,\hat{r}$ $\square$

Confluence of $\mathcal{FMC}$

Theorem

$\mathcal{FMC}$ is confluent

Proof.

by checking that all (∞ly many) multi–one critical peaks are many–multi joinable (in fact one–multi) $\square$

Confluence of $\mathcal{FMC}$

Theorem

$\mathcal{FMC}$ is confluent

Proof.

by checking that all multi–one critical peaks are many–multi joinable

$$\begin{array}{ccc}
-\mathsf{app}_b-\mathsf{app}_a-\mathsf{abs}_b-\mathsf{app}_c-\mathsf{abs}_a-\mathsf{abs}_c- & \to & -\mathsf{app}_b-\mathsf{abs}_b-\mathsf{app}_c-\mathsf{abs}_c- \\
\downarrow & & \downarrow \\
-\mathsf{app}_a-\mathsf{abs}_a- & \to & -
\end{array}$$

$\square$

Further properties of FMC via HRS theory for $\mathcal{FMC}$?

Some rewrite questions and (provisional) answers

1. is ${}_{\beta}^{+}\!\leftarrow$ well-founded (termination model)?
   yes, for typed FMC by Gandy-proof (Barrett, H, McCusker, MFPS 2022)

Further properties of FMC via HRS theory for $\mathcal{FMC}$?

Some rewrite questions and (provisional) answers

1. is $\overset{+}{\underset{\beta}{\leftarrow}}$ well-founded (termination model)?
   yes, for typed FMC by Gandy-proof (Barrett, H, McCusker, MFPS 2022)
2. is equational theory $=_{\text{beta}}$ consistent (non-trivial model)?
   yes, because Church–Rosser and distinct normal forms (Church–Rosser)

Further properties of FMC via HRS theory for $\mathcal{FMC}$?

Some rewrite questions and (provisional) answers

1. is $\overset{+}{\underset{\beta}{\leftarrow}}$ well-founded (termination model)?
   yes, for typed FMC by Gandy-proof (Barrett, H, McCusker, MFPS 2022)
2. is equational theory $=_{\text{beta}}$ consistent (non-trivial model)?
   yes, because Church–Rosser and distinct normal forms (Church–Rosser)
3. do we have good strategies?
   yes, spine reduction is hyper-normalising by random descent

Further properties of FMC via HRS theory for $\mathcal{FMC}$?

Some rewrite questions and (provisional) answers

1. is $\overset{+}{\underset{\beta}{\leftarrow}}$ well-founded (termination model)?
   yes, for typed FMC by Gandy-proof (Barrett, H, McCusker, MFPS 2022)
2. is equational theory $=_{\text{beta}}$ consistent (non-trivial model)?
   yes, because Church–Rosser and distinct normal forms (Church–Rosser)
3. do we have good strategies?
   yes, spine reduction is hyper-normalising by random descent
4. is the combination with eta well-behaved?
   yes, commutes with beta by critical peak criterion

Further properties of FMC via HRS theory for $\mathcal{FMC}$?

Some rewrite questions and (provisional) answers

1. is ${}_{\beta}^{+}\!\!\leftarrow$ well-founded (termination model)?
   yes, for typed FMC by Gandy-proof (Barrett, H, McCusker, MFPS 2022)
2. is equational theory $=_{\text{beta}}$ consistent (non-trivial model)?
   yes, because Church–Rosser and distinct normal forms (Church–Rosser)
3. do we have good strategies?
   yes, spine reduction is hyper-normalising by random descent
4. is the combination with eta well-behaved?
   yes, commutes with beta by critical peak criterion
5. reductions modulo permutation equivalence a computation category?
   yes, because multisteps $\multimap\!\!\!\!\rightarrow_{\text{beta}}$ constitute residual system

Conclusions

- FMC meta-theory via PRS meta-theory feasible via embedding in $\mathcal{FMC}$

Conclusions

- FMC meta-theory via PRS meta-theory feasible via embedding in $\mathcal{FMC}$
- inductive $\Leftrightarrow$ geometric isomorphism gives formal proof–by–potato–picture

Conclusions

- FMC meta-theory via PRS meta-theory feasible via embedding in $\mathcal{FMC}$
- inductive $\Leftrightarrow$ geometric isomorphism gives formal proof–by–potato–picture
- FMC semantics via $\mathcal{FMC}$? surely coding of stacks too coarse; linear types?

Conclusions

- FMC meta-theory via PRS meta-theory feasible via embedding in $\mathcal{FMC}$
- inductive $\Leftrightarrow$ geometric isomorphism gives formal proof–by–potato–picture
- FMC semantics via $\mathcal{FMC}$? surely coding of stacks too coarse; linear types?
- rule schema instead of rule? rule pattern is regular language

Conclusions

- FMC meta-theory via PRS meta-theory feasible via embedding in $\mathcal{FMC}$
- inductive $\Leftrightarrow$ geometric isomorphism gives formal proof–by–potato–picture
- FMC semantics via $\mathcal{FMC}$? surely coding of stacks too coarse; linear types?
- rule schema instead of rule? rule pattern is regular language
- work modulo permutation to make beta, eta local? (no modulo theory . . .)

Opinions to feed your head

1. reviewer 1: Negative aspects of the paper: The technical work seems to be in progress. Most proofs have been omitted, and even the proofs in the appendix have been labeled as “proof sketches”. I haven’t been able to convince myself that the results hold. The main weakness of the paper, from my point of view, is that of communication

Opinions to feed your head

1. reviewer 1: Negative aspects of the paper: The technical work seems to be in progress. Most proofs have been omitted, and even the proofs in the appendix have been labeled as “proof sketches”. I haven’t been able to convince myself that the results hold. The main weakness of the paper, from my point of view, is that of communication
workshop paper reporting on work in progress; which results? reviewer read precisely (thank you!). reported only minor issues. confluence of FMC proven elsewhere by other means (MFPS 2022). surely result is not in doubt.

Opinions to feed your head

1. workshop paper reporting on work in progress
2. reviewer 2: ...the FMC is an important milestone for functional programming ...delve into the idea of FMC and its relation to logic

Opinions to feed your head

1. workshop paper reporting on work in progress
2. reviewer 2: ...the FMC is an important milestone for functional programming ...delve into the idea of FMC and its relation to logic
   sorry; focussed on confluence (IWC); more on FMC on Willem's page

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem's page
3. reviewer 1: . . . speaks about "occurrences" of a term in another term; but the authors use this word with a non-standard meaning

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem's page
3. reviewer 1: . . . speaks about "occurrences" of a term in another term; but the authors use this word with a non-standard meaning
   no standard notion of occurrence of h-o term in literature (programming language theory, proof theory, . . . informal / imprecise / incorrect); used ours (& van Raamsdonk 1994) factoring through HOAS / Church; renders traditional redex-orthogonality-talk obsolete

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem’s page
3. no standard notion of occurrence of h-o term in literature; used ours
4. use higher-order because closed under abstraction from h-o-terms / patterns enabling notion of occurrence (by having variables for h-o-terms / patterns)

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem's page
3. no standard notion of occurrence of h-o term in literature; used ours
4. use higher-order because closed under abstraction from h-o-terms / patterns enabling notion of occurrence (by having variables for h-o-terms / patterns)

   first-order TRSs and second-order frameworks (Klop, Hamana) not closed under abstraction; paraphrasing using contexts, substitutions or ad hoc extending term language possible but awkward (double work)

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem's page
3. no standard notion of occurrence of h-o term in literature; used ours
4. use higher-order because closed under abstraction enabling occurrence
5. use inductive $\Leftrightarrow$ geometric view (Hirokawa et al. CADE 2019) enables implementing critical peak criteria (implemented ; 1 week in Haskell). as basis for formalising critical peak criteria (multi–one for PRSs, one–one (parallel-closed for TRSs (Huet 1980); development-closed for PRSs (1995), and parallel–one (Toyama 1981,Gramlich 1996 for TRSs))

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem's page
3. no standard notion of occurrence of h-o term in literature; used ours
4. use higher-order because closed under abstraction enabling occurrence
5. use inductive $\Leftrightarrow$ geometric view as basis for formalising critical peak criteria
6. conference papers as performance-metric for academic positions (job adverts) is harmful for the environment

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem's page
3. no standard notion of occurrence of h-o term in literature; used ours
4. use higher-order because closed under abstraction enabling occurrence
5. use inductive $\Leftrightarrow$ geometric view as basis for formalising critical peak criteria
6. conference papers as performance-metric for academic positions (job adverts) is harmful for the environment
   if important, then suggest to also pay (even after contracted has ended)

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem's page
3. no standard notion of occurrence of h-o term in literature; used ours
4. use higher-order because closed under abstraction enabling occurrence
5. use inductive $\Leftrightarrow$ geometric view as basis for formalising critical peak criteria
6. conference papers as performance-metric for academic positions is harmful
7. performance-metric proposal: #original results formalised by others

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem's page
3. no standard notion of occurrence of h-o term in literature; used ours
4. use higher-order because closed under abstraction enabling occurrence
5. use inductive $\Leftrightarrow$ geometric view as basis for formalising critical peak criteria
6. conference papers as performance-metric for academic positions is harmful
7. performance-metric proposal: #original results formalised by others

   thanks to CL group in Innsbruck for formalising decreasing diagrams converted ($\hookleftarrow$ de Bruijn,Pous), confluence by Z ($\hookleftarrow$ Dehornoy) for λ and CL, modularity of confluence ($\hookleftarrow$ Toyama) via layer framework (Felgenhauer et al.), part of random descent ($\hookleftarrow$ Newman,Toyama), confluence by development-closedness ($\hookleftarrow$ Huet,Toyama) for TRSs, proof terms ($\hookleftarrow$ Meseguer) for left-linear TRSs and to Yamada for part of sub-Birkhoff ($\hookleftarrow$ Plotkin), . . . ?

Opinions to feed your head

1. workshop paper reporting on work in progress
2. sorry; focussed on confluence; more on FMC on Willem's page
3. no standard notion of occurrence of h-o term in literature; used ours
4. use higher-order because closed under abstraction enabling occurrence
5. use inductive $\Leftrightarrow$ geometric view as basis for formalising critical peak criteria
6. conference papers as performance-metric for academic positions is harmful
7. performance-metric proposal: #original results formalised by others

thank you

(return to NL tomorrow night; contact me after at `oostrom@javakade.nl`)

Semantics of / via higher-order term rewriting?

semantics of addition HRS?

$$
\begin{aligned}
\lambda x.A\,x\,0 &\rightarrow_\rho \lambda x.x \\
\lambda xy.A\,x\,(S\,y) &\rightarrow_\theta \lambda xy.S\,(A\,x\,y)
\end{aligned}
$$

Semantics of / via higher-order term rewriting?

semantics of addition HRS?

$$\begin{aligned} \lambda x.A\,x\,0 &\to_\rho \lambda x.x \\ \lambda xy.A\,x\,(S\,y) &\to_\theta \lambda xy.S\,(A\,x\,y) \end{aligned}$$

factorise through semantics of substitution; simply typed $\lambda\beta\eta$

- interpret base type o as $\mathbb{N}$ ($[\![o]\!] := \mathbb{N}$), $\tau \to \sigma$ as set $[\![\tau]\!] \Rightarrow [\![\sigma]\!]$ of functions from $[\![\tau]\!]$ to $[\![\sigma]\!]$, function application / abstraction according to their name

Semantics of / via higher-order term rewriting?

semantics of addition HRS?

$$\lambda x.A\,x\,0 \;\to_\rho\; \lambda x.x$$
$$\lambda xy.A\,x\,(S\,y) \;\to_\theta\; \lambda xy.S\,(A\,x\,y)$$

factorise through semantics of substitution; simply typed $\lambda\beta\eta$

- interpret each symbol $f : \tau$ as an element of its type $[\![\tau]\!]$, say 0 as $37 \in \mathbb{N}$, S as id $\in \mathbb{N} \Rightarrow \mathbb{N}$, A as first projection $\pi_1 \in \mathbb{N} \Rightarrow \mathbb{N} \Rightarrow \mathbb{N}$

Semantics of / via higher-order term rewriting?

semantics of addition HRS?

$$\begin{aligned} \lambda x.A\,x\,0 &\rightarrow_\rho \lambda x.x \\ \lambda xy.A\,x\,(S\,y) &\rightarrow_\theta \lambda xy.S\,(A\,x\,y) \end{aligned}$$

factorise through semantics of substitution; simply typed $\lambda\beta\eta$

- interpret each symbol $f : \tau$ as an element of its type $[\![\tau]\!]$, say 0 as $37 \in \mathbb{N}$, S as id $\in \mathbb{N} \Rightarrow \mathbb{N}$, A as first projection $\pi_1 \in \mathbb{N} \Rightarrow \mathbb{N} \Rightarrow \mathbb{N}$
- interpret rules ρ and θ as equalities

$$\begin{aligned} (n \mapsto n) &= (n \mapsto n) \\ (n, m \mapsto n) &= (n, m \mapsto n) \end{aligned}$$

Semantics of / via higher-order term rewriting?

semantics of addition HRS?

$$\begin{aligned} \lambda x.A\,x\,0 &\rightarrow_\rho \lambda x.x \\ \lambda xy.A\,x\,(S\,y) &\rightarrow_\theta \lambda xy.S\,(A\,x\,y) \end{aligned}$$

factorise through semantics of substitution; simply typed $\lambda\beta\eta$

- interpret each symbol $f : \tau$ as an element of its type $[\![\tau]\!]$, say 0 as $37 \in \mathbb{N}$, S as $\mathrm{id} \in \mathbb{N} \Rightarrow \mathbb{N}$, A as first projection $\pi_1 \in \mathbb{N} \Rightarrow \mathbb{N} \Rightarrow \mathbb{N}$
- interpret rules ρ and θ as equalities

$$\begin{aligned} (n \mapsto n) &= (n \mapsto n) \\ (n, m \mapsto n) &= (n, m \mapsto n) \end{aligned}$$

of course interpreting as zero, successor, and addition also works

Semantics of / via higher-order term rewriting?

semantics of addition HRS?

$$\begin{aligned} \lambda x.A\,x\,0 &\to_\rho \lambda x.x \\ \lambda xy.A\,x\,(S\,y) &\to_\theta \lambda xy.S\,(A\,x\,y) \end{aligned}$$

factorise through semantics of substitution; simply typed $\lambda\beta\eta$

- interpret each symbol $f : \tau$ as an element of its type $[\![\tau]\!]$, say 0 as $37 \in \mathbb{N}$, S as id $\in \mathbb{N} \Rightarrow \mathbb{N}$, A as first projection $\pi_1 \in \mathbb{N} \Rightarrow \mathbb{N} \Rightarrow \mathbb{N}$
- interpret rules ρ and θ as equalities

$$\begin{aligned} (n \mapsto n) &= (n \mapsto n) \\ (n, m \mapsto n) &= (n, m \mapsto n) \end{aligned}$$

two, successor, and multiplication gives inequalities $>$ on $\mathbb{N}_{\geq 2}$ (termination)

Semantics of / via higher-order term rewriting?

semantics of untyped lambda-beta-eta HRS?

$$\lambda\,(K).\mathrm{abs}\,\lambda x.\mathrm{app}\,K\,x \quad \to_{\mathrm{eta}} \quad \lambda\,(K).K$$

$$\lambda\,(M\,N).\mathrm{app}\,(\mathrm{abs}\,\lambda x.M\,x)\,N \quad \to_{\mathrm{beta}} \quad \lambda\,(M\,N).M\,N$$

Semantics of / via higher-order term rewriting?

semantics of untyped lambda-beta-eta HRS?

$$\begin{array}{rcl} \lambda\,(K).\mathsf{abs}\,\lambda x.\mathsf{app}\,K\,x & \to_{\mathsf{eta}} & \lambda\,(K).K \\ \lambda\,(M\,N).\mathsf{app}\,(\mathsf{abs}\,\lambda x.M\,x)\,N & \to_{\mathsf{beta}} & \lambda\,(M\,N).M\,N \end{array}$$

factorise through semantics of substitution; simply typed $\lambda\beta\eta$; CCC

- interpret beta and eta-rules in CCC (cf. Koymans):

$$\begin{array}{rcl} @ \circ \langle [\![\mathsf{abs}]\!] \circ \langle\,\rangle, @ \circ \langle [\![\mathsf{app}]\!] \circ \langle\,\rangle, \mathsf{id}\rangle\rangle & = & \mathsf{id} \\ @ \circ \langle [\![\mathsf{app}]\!] \circ \langle\,\rangle, @ \circ \langle [\![\mathsf{abs}]\!] \circ \langle\,\rangle, \mathsf{id}\rangle\rangle & = & \mathsf{id} \end{array}$$

Semantics of / via higher-order term rewriting?

semantics of untyped lambda-beta-eta HRS?

$$\begin{array}{rcl} \lambda\,(K).\mathsf{abs}\,\lambda x.\mathsf{app}\,K\,x & \to_{\mathsf{eta}} & \lambda\,(K).K \\ \lambda\,(M\,N).\mathsf{app}\,(\mathsf{abs}\,\lambda x.M\,x)\,N & \to_{\mathsf{beta}} & \lambda\,(M\,N).M\,N \end{array}$$

factorise through semantics of substitution; simply typed $\lambda\beta\eta$; CCC

- interpret beta and eta-rules in CCC (cf. Koymans):

$$\begin{array}{rcl} @ \circ \langle [\![\mathsf{abs}]\!] \circ \langle\,\rangle, @ \circ \langle [\![\mathsf{app}]\!] \circ \langle\,\rangle, \mathsf{id}\rangle\rangle & = & \mathsf{id} \\ @ \circ \langle [\![\mathsf{app}]\!] \circ \langle\,\rangle, @ \circ \langle [\![\mathsf{abs}]\!] \circ \langle\,\rangle, \mathsf{id}\rangle\rangle & = & \mathsf{id} \end{array}$$

- for set / functions: $[\![\mathsf{abs}]\!] \circ [\![\mathsf{app}]\!] = \mathsf{id}$ on D and $[\![\mathsf{app}]\!] \circ [\![\mathsf{abs}]\!] = \mathsf{id}$ on $D \Rightarrow D$