higher-order rewriting

Femke van Raamsdonk

July 4, 2007

combinatory logic

terms:

$$M, N ::= x \mid \mathsf{S} \mid \mathsf{K} \mid \mathsf{I} \mid (M\,N)$$

rules:

$$\begin{array}{lll} \mathsf{S}\,X\,Y\,Z & \to & (X\,Z)\,(Y\,Z) \\ \mathsf{K}\,X\,Y & \to & X \\ \mathsf{I}\,X & \to & X \end{array}$$

combinatory logic

introduced by Moses Schönfinkel 1924

combinatory logic

terms:

$$M, N ::= x \mid \mathsf{S} \mid \mathsf{K} \mid \mathsf{I} \mid (M\,N)$$

rules:

$$\begin{array}{lll} \mathsf{S}\,X\,Y\,Z & \to & (X\,Z)\,(Y\,Z) \\ \mathsf{K}\,X\,Y & \to & X \\ \mathsf{I}\,X & \to & X \end{array}$$

I can be defined using K and S:

$$(\mathsf{S}\,\mathsf{K}\,\mathsf{K})\,x \to (\mathsf{K}x)\,(\mathsf{K}x) \to x$$

combinatory logic

terms:

$$M, N ::= x \mid \mathsf{S} \mid \mathsf{K} \mid \mathsf{I} \mid (M\,N)$$

rules:

$$\begin{array}{lcl} \mathsf{S}\,X\,Y\,Z & \to & (X\,Z)\,(Y\,Z) \\ \mathsf{K}\,X\,Y & \to & X \\ \mathsf{I}\,X & \to & X \end{array}$$

I can be defined using K and S:

$$(\mathsf{S}\,\mathsf{K}\,\mathsf{K})\,x \to (\mathsf{K}x)\,(\mathsf{K}x) \to x$$

reduction may be infinite:

$$(\mathsf{S}\,\mathsf{I}\,\mathsf{I})\,(\mathsf{S}\,\mathsf{I}\,\mathsf{I}) \to \mathsf{I}\,(\mathsf{S}\,\mathsf{I}\,\mathsf{I})\,(\mathsf{I}\,(\mathsf{S}\,\mathsf{I}\,\mathsf{I})) \to (\mathsf{S}\,\mathsf{I}\,\mathsf{I})\,(\mathsf{I}\,(\mathsf{S}\,\mathsf{I}\,\mathsf{I})) \to (\mathsf{S}\,\mathsf{I}\,\mathsf{I})\,(\mathsf{S}\,\mathsf{I}\,\mathsf{I})$$

lambda calculus

terms:

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

rule:

$$(\lambda x.\, M)\,N \to_\beta M[x := N]$$

lambda calculus

introduced by Alonzo Church 1936

lambda calculus

terms:

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

rule:

$$(\lambda x.\, M)\,N \to_\beta M[x := N]$$

reduction may be infinite:

$$(\lambda x.\, x\,x)\,(\lambda x.\, x\,x) \to_\beta (\lambda x.\, x\,x)\,(\lambda x.\, x\,x)$$

expressivity

both CL and lambda calculus are Turing complete

extending lambda calculus

- lambda calculus with pairing
- lambda-calculus with arithmetic
- lambda calculus with explicit substitutions
- lambda calculus with patterns
- ...

extending CL

to orthogonal (first-order) term rewriting systems (OTRSs), e.g.:

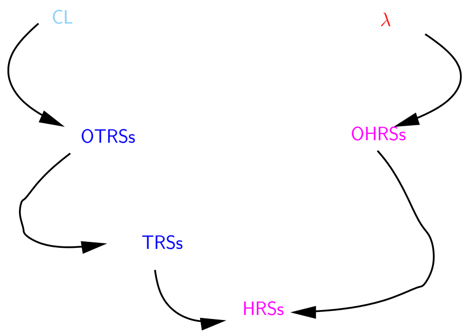
$$\begin{array}{lcl} \mathsf{map}\ F\ \mathsf{nil} & \to & \mathsf{nil} \\ \mathsf{map}\ F\ (\mathsf{cons}\ H\ T) & \to & \mathsf{cons}\ (F\ H)\ (\mathsf{map}\ F\ T) \end{array}$$

to (first-order) term rewriting systems (TRSs), e.g.:

$$\begin{array}{lcl} \mathsf{eq}(X, X) & \to & \mathsf{true} \\ \mathsf{por}(\mathsf{true}, Z) & \to & \mathsf{true} \\ \mathsf{por}(Z, \mathsf{true}) & \to & \mathsf{true} \\ \mathsf{por}(\mathsf{false}, \mathsf{false}) & \to & \mathsf{false} \end{array}$$

towards higher-order rewriting

both patterns and bound variables



higher-order rewriting: rewriting with bound variables

- adds bound variables to first-order rewriting
- adds expressive power and algebraic data-types to λ-calculus

Part II

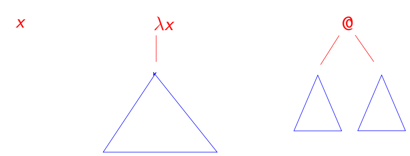
lambda calculus

this morning

- lambda calculus
- towards higher-order rewriting
- termination
- confluence

untyped lambda calculus: terms

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



untyped lambda calculus: reduction

rule:

$$(\lambda x.\, M)\, N \to_\beta M[x := N]$$

example:

$$(\lambda x.\, x\, x)\,(\lambda x.\, x\, x) \to_\beta (\lambda x.\, x\, x)\,(\lambda x.\, x\, x)$$

data types in lambda calculus: example

define:

$$\begin{array}{lcl} \mathsf{true} & = & \lambda x.\, \lambda y.\, x \\ \mathsf{false} & = & \lambda x.\, \lambda y.\, y \\ \mathsf{not} & = & \lambda x.\, x\, \mathsf{false}\, \mathsf{true} \end{array}$$

then:

$$\begin{array}{ll} \mathsf{not}\ \mathsf{true} & = \\ (\lambda x.\, x\, \mathsf{false}\, \mathsf{true})\ \mathsf{true} & \to_\beta \\ \mathsf{true}\ \mathsf{false}\ \mathsf{true} & \to_\beta^* \\ \mathsf{false} & \end{array}$$

untyped lambda calculus: properties

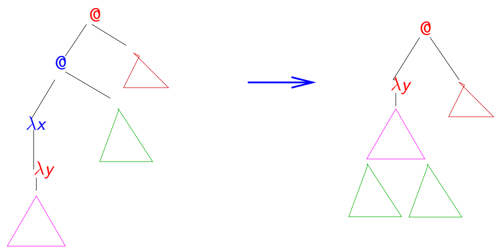
- it is Turing complete
- β-reduction is confluent
- every β-reduction can be transformed into one where redexes are contracted in a standard (left-to-right) way
- all developments terminate

creation of redexes

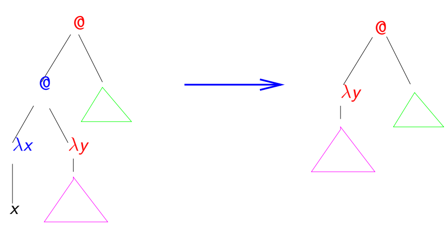
the three ways of creating redexes (Jean-Jacques Lévy):

1. $(\lambda x.\, \lambda y. M)\, P\, Q \to_\beta (\lambda y.\, M[x := P])\, Q$
2. $(\lambda x.\, x)\,(\lambda y.\, M)\, N \to_\beta (\lambda y.\, M)\, N$
3. $(\lambda z.\, C[z\, Q])\,(\lambda x.\, P) \to C[(\lambda x.\, P)\, Q]$

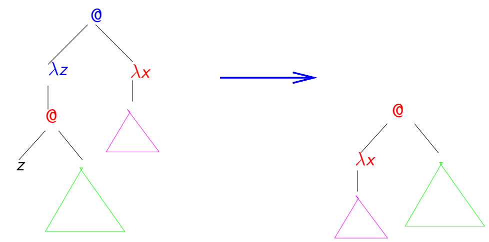
creation of redexes: 1



creation of redexes: 2



creation of redexes: 3



development: idea

only residuals of redexes in the initial term are contracted

so created redexes are not contracted

development: example

$$
\begin{array}{rl}
(\lambda x.\, f\,x\,x)\,((\lambda y.\, y)\,a) & \to_\beta \\
f\,((\lambda y.\, y)\,a)\,((\lambda y.\, y)\,a) & \to_\beta \\
f\,a\,((\lambda y.\, y)\,a) & \to_\beta \\
f\,a\,a &
\end{array}
$$

finiteness of developments

infinite behaviour is caused by created redexes

development: non-example

$$
\begin{array}{rl}
(\lambda x.\, \lambda y.\, f\,x\,y)\,a\,b & \to_\beta \\
(\lambda y.\, f\,a\,y)\,b & \to_\beta \\
f\,a\,b &
\end{array}
$$

finiteness of superdevelopments

infinite behaviour is caused by the 3rd type of created redexes

developments: via the underlined lambda calculus

terms:

$$M, N ::= x \mid \lambda x.\, M \mid (M\, N) \mid (\underline{\lambda} x.\, M)N$$

rule:

$$(\underline{\lambda} x.\, M)N \to_{\underline{\beta}} M[x := N]$$

reduction:

$$\begin{array}{rl} (\underline{\lambda} x.\, f\, x\, x)\, ((\underline{\lambda} y.\, y)\, a) & \to_{\underline{\beta}} \\ f\, ((\underline{\lambda} y.\, y)\, a)\, ((\underline{\lambda} y.\, y)\, a) & \to_{\underline{\beta}} \\ f\, a\, ((\underline{\lambda} y.\, y)\, a) & \to_{\underline{\beta}} \\ f\, a\, a & \end{array}$$

the underlined lambda calculus is terminating (1)

theorem: σ terminating $\Rightarrow$ M^σ terminating

proof: Induction on the definition of terms.

1. $M = x$
   variables are terminating and σ is terminating

the underlined lambda calculus is terminating (2)

theorem: σ terminating $\Rightarrow$ M^σ terminating

proof: Induction on the definition of terms.

1. $M = x$
2. $M = \lambda x.\, P$
   IH $\Rightarrow$ P^σ is terminating

the underlined lambda calculus is terminating (3)

theorem: σ terminating $\Rightarrow$ M^σ terminating

proof: Induction on the definition of terms.

1. $M = x$
2. $M = \lambda x.\, P$
3. $M = (P\, Q)$
   IH $\Rightarrow$ P^σ and Q^σ are terminating
   note: $\underline{\lambda} x.\, M$ is not an underlined term
   no interaction between P and Q

the underlined lambda calculus is terminating (4)

theorem: σ terminating $\Rightarrow$ M^σ terminating
proof: Induction on the definition of terms.

1. $M = x$
2. $M = \lambda x.\, P$
3. $M = (P\, Q)$
4. $M = (\underline{\lambda} x.\, P)\, Q$
   - head redex is contracted
     IH $\Rightarrow$ $P^{\sigma[x:=Q^\sigma]} = P^\sigma[x := Q^\sigma]$ is terminating
   - head redex is not contracted
     IH $\Rightarrow$ P^σ and Q^σ are terminating

Part III

towards higher-order rewriting

exercise

how can redexes be created in CL?

simple types

$$A, B ::= b \mid (A \to B)$$

examples:
nat
nat $\to$ nat $\to$ nat
(nat $\to$ nat) $\to$ bool

simply typed terms

1. $x : A$
   if x is a variable of type A
2. $f : A$
   if f is a constant of type A
3. $\lambda x.\, M : A \to B$
   if $M : B$ and x a variable of type A
4. $(F\, M) : B$
   if $F : A \to B$ and $M : A$

there is no self-application as in $\lambda x.\, x\, x$

example: predicate logic

normal:

$$\forall x.\, \text{plays}(x)$$

HOS:

$$\forall\,($$

higher-order syntax

all bindings are expressed using λ

beta reduction

$$(\lambda x.\, M)N \rightarrow_{\beta} M[x := N]$$

but now β-reduction terminates
(more this afternoon !)

different approaches

do we want to see the substitution explicitly ?

YES
then we work modulo α

NO
then we work modulo $\alpha\beta$ (and η)

beta reduction implements substitution

$\forall x.\, P(x)$	$\forall\, (\lambda x.\, (P\, x))$
	$(\lambda P.\, \forall\, (\lambda x.\, (P\, x)))\, (\lambda u.\, \mathsf{plays}\, u)$
	$\downarrow$
$P(x) \rightsquigarrow \mathsf{plays}(x)$	$\forall\, (\lambda x.\, ((\lambda u.\, \mathsf{plays}\, u)\, x))$
	$\downarrow$
	$\forall\, (\lambda x.\, (\mathsf{plays}\, x))$
$\forall x.\, \mathsf{plays}(x)$	$\forall\, (\lambda x.\, (\mathsf{plays}\, x))$

higher-order rewriting: systems

- modulo α
  extensions of λ-calculus
  AFS (Jouannaud and Okada 1991)
- modulo $\alpha\beta\eta$

  CRS (Klop 1980)
  ERS (Khasidashvili 1990)
  HOER (Wolfram 1991)
  HRS (Nipkow 1991)

restricted eta expansion

the η-expansion rule:

$$M \to_{\overline{\eta}} \lambda x.\,(M\,x)$$

side conditions:

- types ok
- fresh variables
- do not create β-redexes

long normal form

a β-normal form:

$$\lambda x_1 \dots \lambda x_m.\,(c\,M_1 \dots M_n)$$

its long normal form:

$$\lambda x_1 \dots \lambda x_m\,\lambda x_{m+1} \dots \lambda x_p.\,(c\,M_1' \dots M_n'\,x_{m+1}' \dots x_p')$$

unique representatives

every $\beta\eta$ equivalence class has a unique representative:
its β long normal form

requirements on a rewrite rule $l \to r$?

as in TRSs:

- all free variables of r occur in l
- l is not a free variable

in addition:

- l and r have the same type
- l and r are in $\beta\overline{\eta}$ normal form

decidability

is it decidable whether a rule can be applied ?

are critical pairs decidable ?

unification is undecidable

higher-order unification problem:

$$g^{\sigma} =_{\beta\eta} l^{\tau}$$

result:

- undecidable (Huet 1976)

is matching decidable ?

higher-order matching problem:

$$s =_{\beta\eta} l^{\sigma}$$

results:

- 3rd order: decidable (Dowek 1994)
- 4th order: decidable (Padovani 2001)
- general: decidable (Stirling 2006)
- modulo β: undecidable (Loader 2003)

restriction of left-hand sides to patterns

unification is decidable for patterns (Miller 1991)

pattern:
all arguments of a free variable are different bound variables

not a pattern:

$$f(Z(Z(x)))$$

why is it called higher-order ?

order of a rewrite system:
maximum order of a free variable in a rule

order of a type:

1. $\text{order}(b) = 1$
2. $\text{order}(A_1 \to \ldots \to A_n \to b) = \max\{\text{order}(A_i)\} + 1$

Part IV

termination

order of a rewrite system: examples

0th order:

$$a \to b$$

1st order:

$$\text{plus}(0, Y) \to Y$$

2nd order:

$$\text{map}(\lambda x.\, F(x), \text{nil}) \to \text{nil}$$

3rd order:

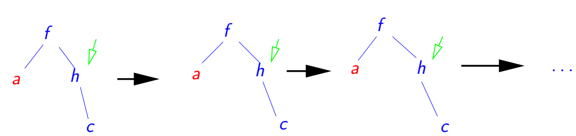
$$f(\lambda x.\, Z(x)) \to Z(\lambda u.\, a)$$

outermost-fair rewriting: idea

every outermost redex is eventually contracted

example of a non-outermost-fair reduction:

$$\begin{array}{lcl} a & \to & a \\ h(x) & \to & h'(x) \\ f(x, h'(y)) & \to & c \end{array}$$



outermost-fair rewriting is normalizing

for TRSs that are almost orthogonal
(O'Donnell 1977)

for HRSs that are weakly orthogonal and fully extended
(van Oostrom 1999)

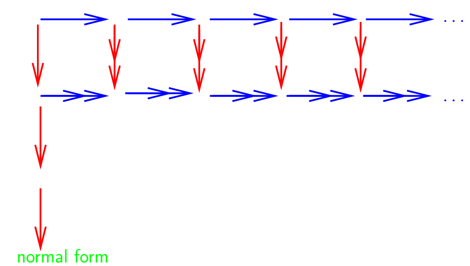
why is higher-order more difficult ?

a non-outermost redex can create an outermost redex

so we will need an additional restriction

proof idea

induction on the maximum length to normal form
outermost-fair is stable under projection

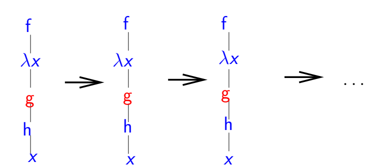


example

rules:

$$\begin{array}{lcl} \mathsf{f}(\lambda x.\, Z) & \to & \mathsf{a} \\ \mathsf{g}(X) & \to & \mathsf{g}(X) \\ \mathsf{h}(X) & \to & \mathsf{a} \end{array}$$

an infinite outermost-fair reduction of a normalizing term:



fully applied: idea

avoid the occur check:

a free variable has as arguments
all variables that are at that position bound

Part V

confluence

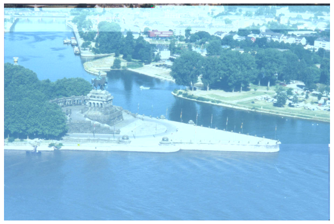
why is weakly orthogonal difficult ?

an outermost redex may (as such) disappear

- because it is contracted
- because it is not a redex anymore
- because it is not outermost anymore

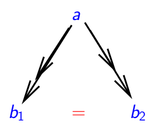
confluence

two coinitial rewrite sequences can be joined



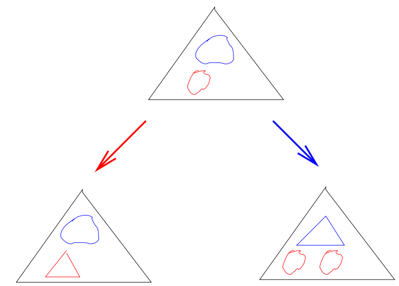
why confluence ?

confluence yields uniqueness of normal forms



orthogonality

rewrite steps do not destroy each other



methods to prove confluence of HRSs

- orthogonality ⇒ confluence
  (Aczel 1978, Klop 1980, Khasidashvili 1990)
- joinability of critical pairs ⇒ local confluence
  (Nipkow 1991)
  joinability of critical pairs and termination ⇒ confluence
- critical pairs development closed ⇒ confluence
  (van Oostrom 1996)

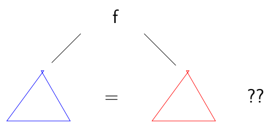
orthogonality: definition

rewrite steps do not destroy each other
is guaranteed by two conditions on the rewrite rules:

- left-hand sides are linear
- left-hand sides are non-overlapping

left-linearity: idea

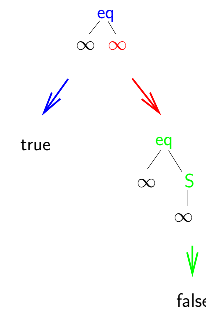
no check for equality



non-left-linearity may yield non-confluence

$$\begin{array}{lcl} \infty & \to & \mathsf{S}(\infty) \\ \mathsf{eq}(X,X) & \to & \mathsf{true} \\ \mathsf{eq}(X,\mathsf{S}(X)) & \to & \mathsf{false} \end{array}$$

is not confluent:



left-linearity: definition

a free variable occurs at most once in a left-hand side

non-overlapping: definition

two left-hand sides are not unifiable
on a non-variable position

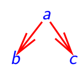
unification is decidable because left-hand sides are patterns

overlap may yield non-confluence

rules:

$$\begin{array}{ccc} a & \rightarrow & b \\ a & \rightarrow & c \end{array}$$

divergence:



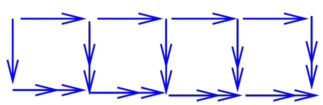
orthogonality implies confluence: TML proof method

define relation $\rightsquigarrow$ with

- $\rightsquigarrow$ has the diamond property
- $\rightsquigarrow^* = \rightarrow^*$

orthogonality implies confluence: proof methods

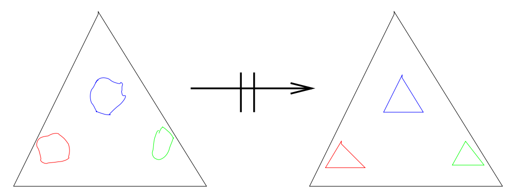
- using developments and the parallel moves lemma



- Tait-Martin-Löf method using an inductive definition

what do we take for $\rightsquigarrow$?

for TRSs: parallel reduction $\mathrel{-\!\!\!\parallel\!\!\!\rightarrow}$

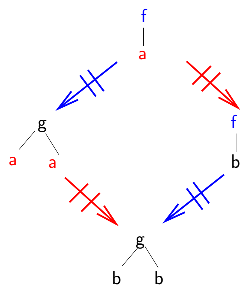


parallel reduction: example

rules:

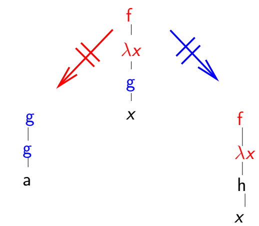
$$\begin{array}{lcl} \mathsf{f}(X) & \rightarrow & \mathsf{g}(X, X) \\ \mathsf{a} & \rightarrow & \mathsf{b} \end{array}$$

parallel step:



parallel reduction for HRSs: no diamond property

$$\begin{array}{lcl} \mathsf{f}(\lambda x.\, Z(x)) & \rightarrow & Z(Z(\mathsf{a})) \\ \mathsf{g}(X) & \rightarrow & \mathsf{h}(X) \end{array}$$



parallel reduction: not for HRSs

parallel reduction does not have the diamond property
because residuals of parallel redexes may be nested

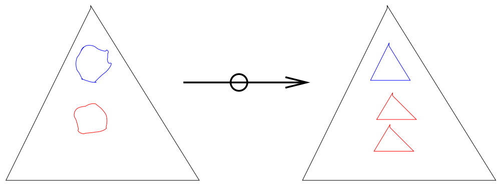
parallel reduction for lambda: no diamond property

divergence:

$$\begin{array}{lcl} (\lambda x.\, (\lambda y.\, x)\, \mathsf{I})\, (\mathsf{I}\, \mathsf{I}) & \mathrel{\|\!\!\!\longrightarrow} & (\lambda y.\, \mathsf{I}\, \mathsf{I})\, \mathsf{I} \\ (\lambda x.\, (\lambda y.\, x)\, \mathsf{I})\, (\mathsf{I}\, \mathsf{I}) & \mathrel{\|\!\!\!\longrightarrow} & (\lambda x.\, x)\, \mathsf{I} \end{array}$$

what do we take for $\rightsquigarrow$?

for HRSs: multistep reduction $\mathrel{-\!\!\circ\!\!\to}$



how do we proceed ?

- define (inductively) $\mathrel{-\!\!\circ\!\!\to}$
- prove $\mathrel{-\!\!\circ\!\!\to}^* = \to^*$
- define (inductively) s^* with $s \mathrel{-\!\!\circ\!\!\to} s^*$
- prove (inductively) if $s \mathrel{-\!\!\circ\!\!\to} s'$ then $s' \mathrel{-\!\!\circ\!\!\to} s^*$ (triangle)

multistep reduction: example

rule:

$$\mathsf{f}(X) \to \mathsf{g}(X)$$

multistep:

$$\mathsf{h}(\mathsf{f}(\mathsf{f}(a)), \mathsf{f}(\mathsf{f}(a))) \mathrel{-\!\!\circ\!\!\to} \mathsf{h}(\mathsf{g}(\mathsf{g}(a)), \mathsf{g}(\mathsf{g}(a)))$$

parallel step:

$$\mathsf{h}(\mathsf{f}(\mathsf{f}(a)), \mathsf{f}(\mathsf{f}(a))) \mathrel{-\!\!\circ\!\!\to} h(\mathsf{g}(\mathsf{f}(a)), \mathsf{g}(\mathsf{f}(a)))$$

multistep reduction for lambda calculus

1. variable
$$x \mathrel{-\!\!\circ\!\!\to} x$$
2. abstraction
$$\frac{M \mathrel{-\!\!\circ\!\!\to} M'}{\lambda x.\, M \mathrel{-\!\!\circ\!\!\to} \lambda x.\, M'}$$
3. application
$$\frac{M \mathrel{-\!\!\circ\!\!\to} M' \qquad N \mathrel{-\!\!\circ\!\!\to} N'}{(M\,N) \mathrel{-\!\!\circ\!\!\to} (M'\,N')}$$
4. beta
$$\frac{M \mathrel{-\!\!\circ\!\!\to} M' \qquad N \mathrel{-\!\!\circ\!\!\to} N'}{(\lambda x.\, M)\,N \mathrel{-\!\!\circ\!\!\to} M'[x := N']}$$

multistep reduction for lambda calculus: example

$(\lambda y.\, y\, y)\, x \mathrel{-\!\!\circ\!\!\to} x\, x$

$(\lambda z.\, z)\, a \mathrel{-\!\!\circ\!\!\to} a$

$(\lambda x.\, (\lambda y.\, y\, y)\, x)\, ((\lambda z.\, z)\, a) \mathrel{-\!\!\circ\!\!\to} a\, a$

multistep reduction for HRSs: example

rules:

$$\begin{array}{lcl} \mathsf{a} & \to & \mathsf{b} \\ \mathsf{f}(x) & \to & \mathsf{g}(x) \end{array}$$

multisteps:

$$\mathsf{a} \mathrel{-\!\!\circ\!\!\to} \mathsf{b}$$

$$\mathsf{f}(\mathsf{a}) \mathrel{-\!\!\circ\!\!\to} \mathsf{g}(\mathsf{b})$$

multistep reduction for HRSs

1. variable
$$\frac{\vec{s} \mathrel{-\!\!\circ\!\!\to} \vec{t}}{x(\vec{s}) \mathrel{-\!\!\circ\!\!\to} x(\vec{t})}$$
2. function
$$\frac{\vec{s} \mathrel{-\!\!\circ\!\!\to} \vec{t}}{\mathsf{f}(\vec{s}) \mathrel{-\!\!\circ\!\!\to} \mathsf{f}(\vec{t})}$$
3. abstraction
$$\frac{s \mathrel{-\!\!\circ\!\!\to} t}{\lambda x.\, s \mathrel{-\!\!\circ\!\!\to} \lambda x.\, t}$$
4. rule
$$\frac{\sigma \mathrel{-\!\!\circ\!\!\to} \tau}{l^{\sigma} \mathrel{-\!\!\circ\!\!\to} r^{\tau}}$$

difference between parallel reduction and multistep

parallel reduction:

$$l^{\sigma} \mathrel{-\!\!\|\!\!\to} r^{\sigma}$$

multistep:

$$\frac{\sigma \mathrel{-\!\!\circ\!\!\to} \tau}{l^{\sigma} \mathrel{-\!\!\circ\!\!\to} r^{\tau}}$$

multistep simulates rewriting

$$\mathrel{-\!\!\circ\!\!\to}^* = \to^*$$

s^* for HRSs

intuition: contract all redexes in s
goal: universal common reduct

1. $(x(s_1, \ldots, s_n))^* = x(s_1^*, \ldots, s_n^*)$
2. $(\mathsf{f}(s_1, \ldots, s_n))^* = \mathsf{f}(s_1^*, \ldots, s_n^*)$
3. $(\lambda x.\, s)^* = \lambda x.\, s^*$
4. $(l^\sigma)^* = r^{\sigma^*}$

M^* for lambda calculus

intuition: contract all redexes in M
goal: universal common reduct

1. $x^* = x$
2. $(\lambda x.\, M)^* = \lambda x.\, M^*$
3. $(M\, N)^* = (M^*\, N^*)$
4. $((\lambda x.\, M)\, N)^* = M^*[x := N^*]$

triangle: s^* is a universal common reduct

lemma:
if $s \mathrel{-\!\!\circ\!\!\to} s'$ then $s' \mathrel{-\!\!\circ\!\!\to} s^*$
proof:
induction on the definition of $\mathrel{-\!\!\circ\!\!\to}$

$$\begin{tikzcd}
s \arrow[d, "\circ"] \arrow[dr, "\circ"] & \\
s' \arrow[r, "\circ"] & s^*
\end{tikzcd}$$

orthogonality implies confluence

- multistep reduction instead of parallel reduction
- alternative: use developments instead of inductive definitions
- extends to the weakly orthogonal case

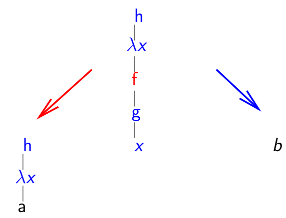
critical pairs: idea

critical pairs arise from most general overlaps

rules:

$$\begin{array}{lcl} f(X) & \to & a \\ h(\lambda x.\, f(g(x))) & \to & b \end{array}$$

critical pair:



exercise

is it possible to define s^* for parallel reduction ?

prove the diamond property of $-\!\!\circ\!\!\twoheadrightarrow$ without s^*

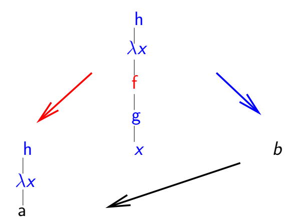
critical pairs: lemma

all critical pairs joinable $\Rightarrow$ local confluence

rules:

$$\begin{array}{lcl} f(X) & \to & a \\ h(\lambda x.\, f(g(x))) & \to & b \\ b & \to & h(\lambda x.\, a) \end{array}$$

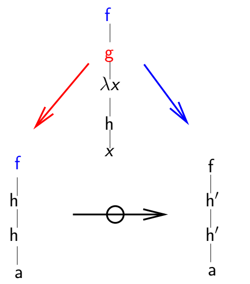
joinable critical pair:



development closed: idea

rules:

$$\begin{array}{lcl} f(g(\lambda x.\, Z(x))) & \to & f(Z(Z(a))) \\ g(\lambda x.\, h'(x)) & \to & h(h(a)) \\ h(X) & \to & h'(X) \end{array}$$



concluding remarks

some concepts for higher-order rewriting:

- multistep reduction instead of parallel reduction
- fully applied rewrite rules

some results for higher-order rewriting:

- critical pair lemma
- orthogonality yields confluence
- standardization
- recursive path ordering
- outermost-fair rewriting is normalizing

development closed: theorem

for HRSs:
all critical pairs development closed $\Rightarrow$ confluence