Techniques de réécriture et transformations

Horatiu CIRSTEA and Claude KIRCHNER

27 janvier 2007

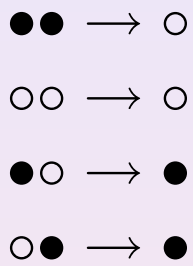
Mathematics is frequently described as “the science of pattern,” a characterisation that makes more sense than most, both of pure mathematics, but also of the ability of mathematics to connect to the world teeming with patterns, symmetries, regularities, and uniformities

Jon Barwise
Lawrence Moss

Roadmap

A simple game

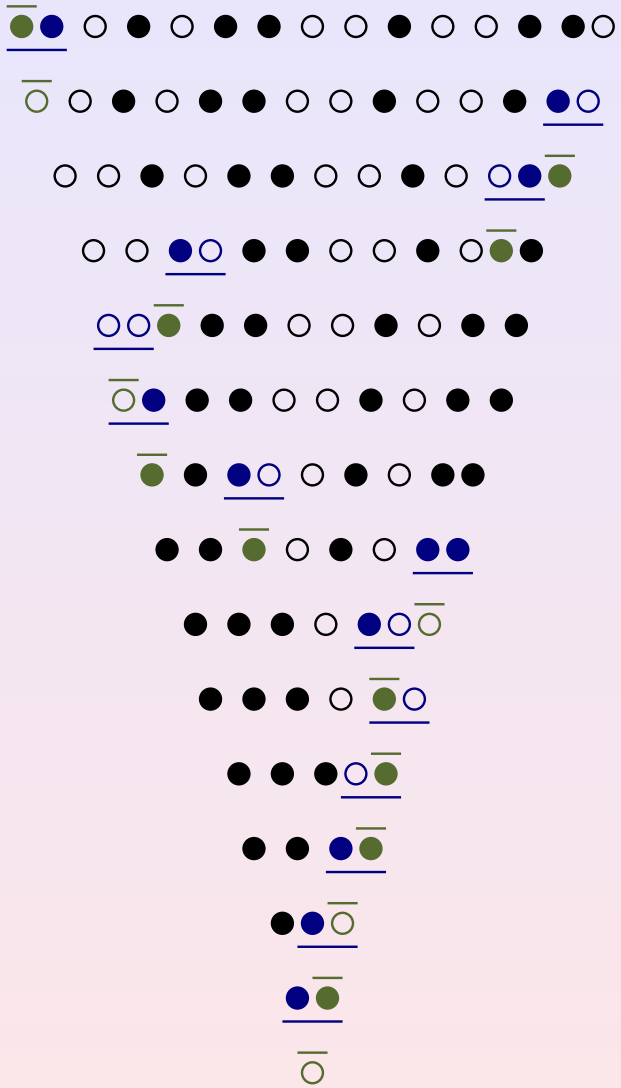
The rules of the game :



A starting point :

Who wins ?

➥ Who puts the last white ?

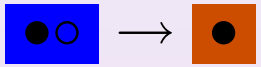


May I always win ? Does the game terminate ? Do we always get the same result ?

What are the basic operations that have been used ?

1– Matching

The data : 

The rewrite rule : 

2– Compute what should be substituted

The lefthand side : 

3– Replacement

The new generated data : 

Note that the last list is a NEW object.

Addition in Peano arithmetic

Peano gives a meaning to addition by using the following axioms :

$$0 + x = x$$
$$s(x) + y = s(x + y)$$

What's the result of $s(s(0)) + s(s(0))$?

$$\begin{aligned} s(s(0)) + s(s(0)) &= s(s(0) + s(s(0))) \\ &= s(s(0 + s(s(0)))) \\ &= s(s(s(s(0)))) \\ &= s(0) + s(s(s(0))) \\ &= 0 + 0 + 0 + s(s(s(s(0)))) \\ &= \dots \end{aligned}$$

Is there a *better* result ?

Addition in Peano arithmetic

Compute a result by turning the equalities into rewrite rules :

$$0 + x \to x$$
$$s(x) + y \to s(x + y)$$

$s(s(0)) + s(s(0)) \to s(s(0) + s(s(0))) \to$
$s(s(0 + s(s(0)))) \to$
$s(s(s(s(0))))$

Is this computation *terminating*,

is there always a *result* (e.g. an expression without $+$)

is such a result *unique* ? ? ?

What are the basic operations that have been used ?

1– Matching

The data : $s(\,s(0)\,) + s(s(0))$

The rewrite rule : $s(\,x\,) + y \longrightarrow s(x + y)$

2– Compute what should be substituted

The instanciated lhs : $s(\,s(0) + s(s(0))\,)$

3– Replacement

The new generated data : s(s(0)+s(s(0)))

Note that this last entity is a NEW object.

Fibonacci

$$
\begin{array}{lll}
[\alpha] & \mathit{fib}(0) & \rightarrow \quad 1 \\
[\beta] & \mathit{fib}(1) & \rightarrow \quad 1 \\
[\gamma] & \mathit{fib}(n) & \rightarrow \quad \mathit{fib}(n-1) + \mathit{fib}(n-2)
\end{array}
$$

$\mathit{fib}(3) \rightarrow \mathit{fib}(2) + \mathit{fib}(1)$

$\mathit{fib}(2) + \mathit{fib}(1) \rightarrow \mathit{fib}(2) + 1$

$\mathit{fib}(2) + 1 \rightarrow \mathit{fib}(1) + \mathit{fib}(0) + 1$

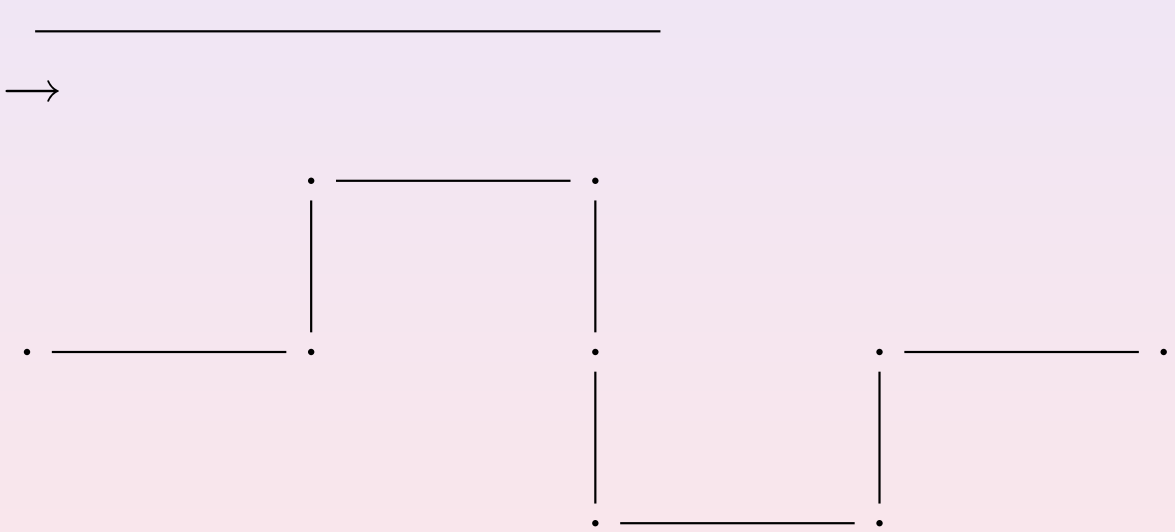
$\mathit{fib}(1) + \mathit{fib}(0) + 1 \rightarrow 1 + \mathit{fib}(0) + 1$

$\ldots$

Finally $\mathit{fib}(3) = 3$, $\mathit{fib}(4) = 5$, ...

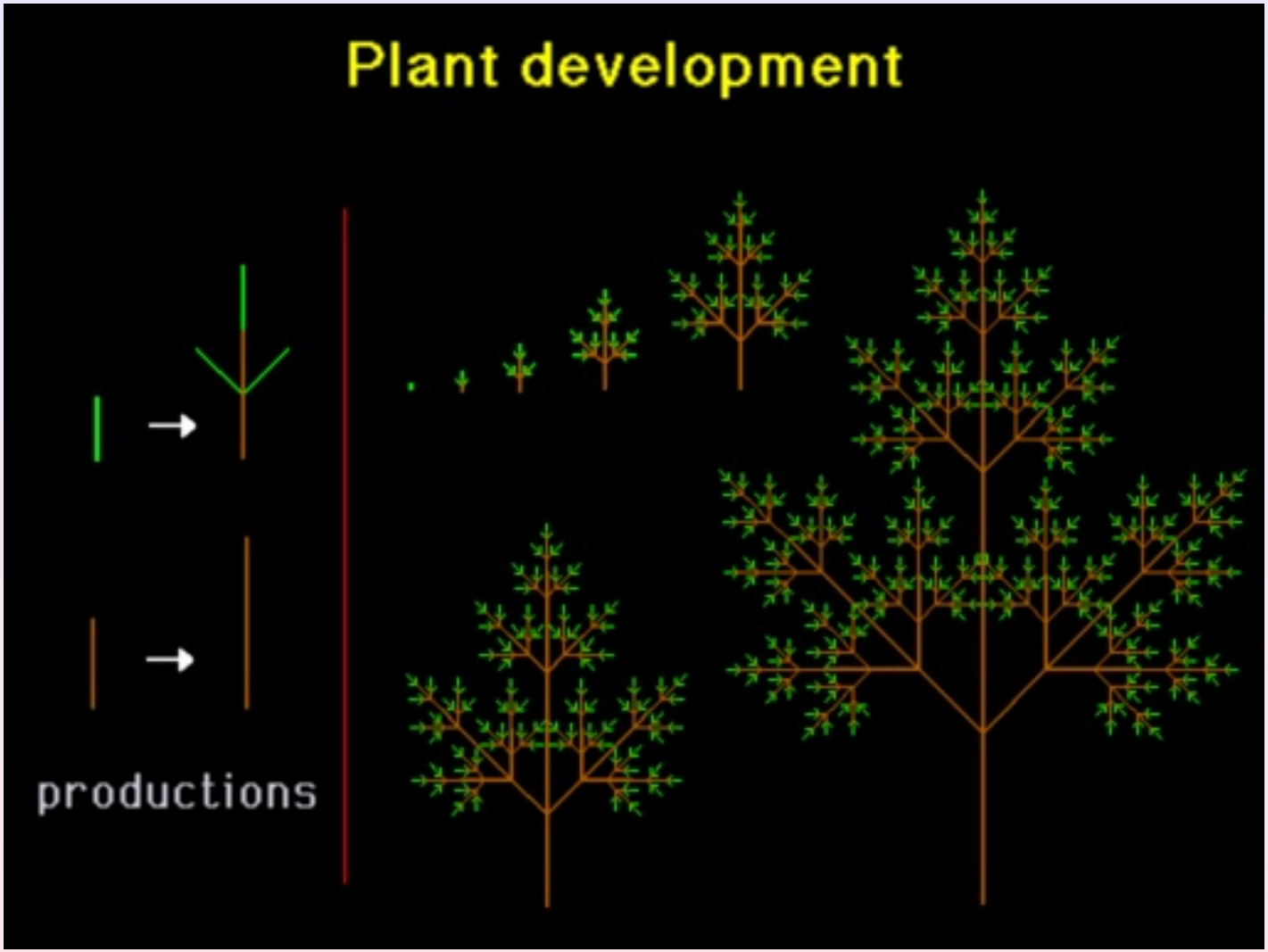
Graphical Rewriting

$$F \rightarrow F + F - F - FF + F + F - F$$



L-systems (Lindenmeier)

Ecological Rewriting



http ://algorithmicbotany.org/

Sorting by rewriting

```
rules for List
  X, Y : Nat ; L L' L'' : List;
   hd (X L) => X ;                          tl (X L) => L ;
   sort nil => nil .
   sort (L X L' Y L'') =>  sort (L Y L' X L'')  if Y < X .
end

sort (6 5 4 3 2 1) => ...

sorts NeList List ;   subsorts Nat < NeList < List ;
operators
  nil : List ;
  @ @ : (List List) List       [associative id: nil] ;
  @ @ : (NeList List) NeList   [associative] ;
  hd @ : (NeList) Nat ;
  tl @ : (NeList) List ;
  sort @ : (List) List ;
end
```

On what objects can rewriting act ?

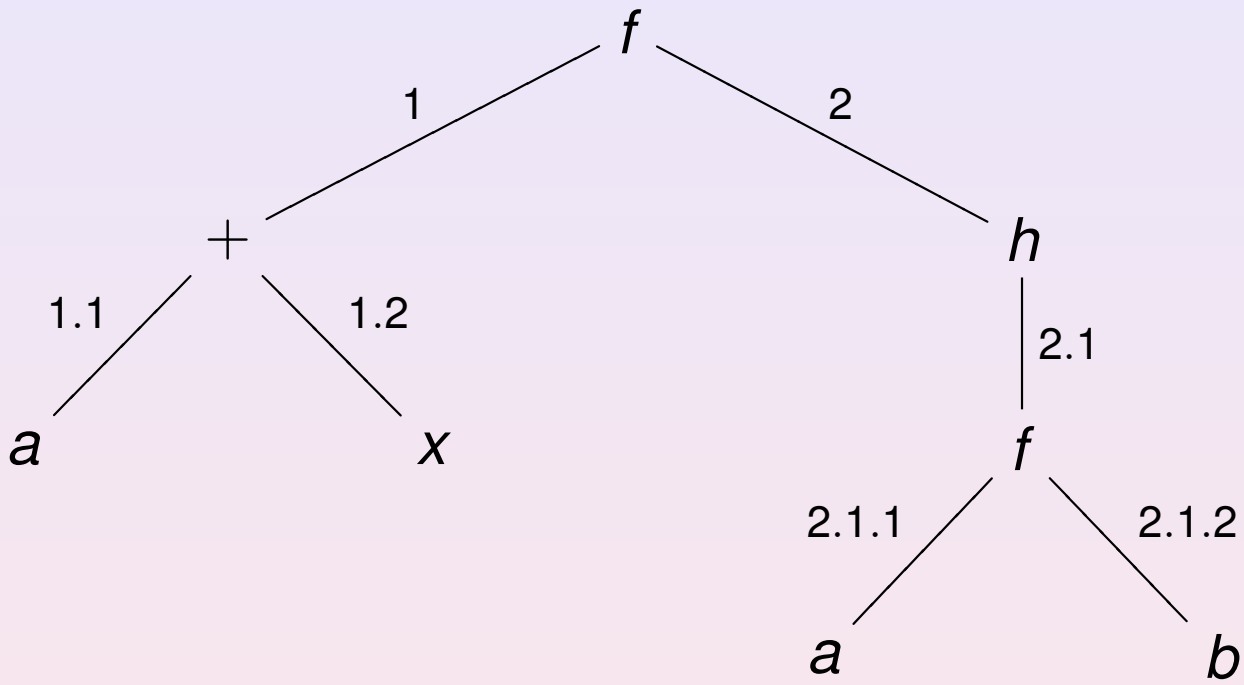
It can be defined on

- terms like $s(s(0)) + s(s(0))$
- strings like “What is rewriting ?” (`sed` performs string rewriting)
- graphs
- sets
- multisets
- . . .

We will “restrict” here to first-order terms.

Terms as trees

$t = f(a + x, h(f(a, b)))$ is represented by :



$|t|$ is the size of t i.e. the cardinality of $\mathcal{D}om(t)$.

$$|f(a + x, h(f(a, b)))| = 8$$

$\mathcal{V}ar(t)$ denotes the set of variables in t.

Formally

Matching

Finding a substitution σ such that

$$\sigma(l) = t$$

is called the matching problem from l to t.

This is denoted $l \ll^? t$

It is decidable in linear time in the size of t.

It induces a relation on terms called subsumption

Matching : A rule based description

Delete
$$t \ll^? t \wedge P \rightarrow P$$

Decomposition
$$f(t_1, \ldots, t_n) \ll^? f(t'_1, \ldots, t'_n) \wedge P \rightarrow \bigwedge_{i=1,\ldots,n} t_i \ll^? t'_i \wedge P$$

SymbolClash
$$f(t_1, \ldots, t_n) \ll^? g(t'_1, \ldots, t'_m) \wedge P \rightarrow \mathit{Fail} \quad \text{if } f \neq g$$

SymbolVariableClash
$$f(t_1, \ldots, t_n) \ll^? x \wedge P \rightarrow \mathit{Fail} \quad \text{if } x \in \mathcal{X}$$

MergingClash
$$x \ll^? t \wedge x \ll^? t' \wedge P \rightarrow \mathit{Fail} \quad \text{if } t \neq t'$$

Find a match

$$x+(y*3) \ll^? 1+(4*3)$$

$$\rightarrow_{\text{Decomposition}} x \ll^? 1 \ \wedge \ y*3 \ll^? 4*3$$

$$\rightarrow_{\text{Decomposition}} x \ll^? 1 \ \wedge \ y \ll^? 4 \ \wedge \ 3 \ll^? 3$$

$$\rightarrow_{\text{Delete}} x \ll^? 1 \ \wedge \ y \ll^? 4$$

$$x+(y*y) \ll^? 1+(4*3)$$

$$\rightarrow_{\text{Decomposition}} x \ll^? 1 \ \wedge \ y*y \ll^? 4*3$$

$$\rightarrow_{\text{Decomposition}} x \ll^? 1 \ \wedge \ y \ll^? 4 \ \wedge \ y \ll^? 3$$

$$\rightarrow_{\text{MergingClash}} \mathit{Fail}$$

Matching rules

Does it terminate ?
Do we always get the same result ?

Theorem The normal form by the rules in Match, of any matching problem $t \ll^? t'$ such that $\mathcal{V}ar(t) \cap \mathcal{V}ar(t') = \emptyset$, exists and is unique.

1. If it is *Fail*, then there is no match from t to t′.
2. If it is of the form $\bigwedge_{i \in I} x_i \ll^? t_i$ with $I \neq \emptyset$, the substitution $\sigma = \{x_i \mapsto t_i\}_{i \in I}$ is the unique match from t to t′.
3. If it is empty then t and t′ are identical : $t = t'$.

Definition of rewriting

It relies on 5 notions :

- The objects : terms and rewrite rules
- The actions
  - matching
  - substitutions
  - replacement

and, given a rule and a term, it consists in :

- finding a subterm of the term
- that matches the left hand side of the rule
- and replacing that subterm by the right hand side of the rule instanciated by the match

Formally

Conditional rules

$$l \longrightarrow r \quad \textbf{if} \quad c$$

- $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$,
- c a boolean term
- $\mathcal{V}ar(r) \cup \mathcal{V}ar(c) \subseteq \mathcal{V}ar(l)$

The rule applies on a term t provided the matching substitution σ allows $c\sigma$ to reduce to $true$.

Applying a conditional rewrite rule

$$
\begin{array}{rcll}
even(0) & \longrightarrow & true & \\
even(s(x)) & \longrightarrow & odd(x) & \\
odd(x) & \longrightarrow & true & \textbf{if} \quad not(even(x)) \\
odd(x) & \longrightarrow & false & \textbf{if} \quad even(x)
\end{array}
$$

$$even(s(0)) \longrightarrow odd(0) \longrightarrow false$$

Generalized rules

Expressiveness of rewriting

[Max Dauchet 1989]
A Turing machine can be simulated by a single rewrite rule.

This unique rewrite rule can further be left linear and regular !

On the use of term rewriting

- for programming (ASF, ELAN, MAUDE, ML, OBJ, Stratego, TOM, ...)
- for proving (Completion procedures, proof systems, ...)
- for solving (Constraint manipulations, ...)
- for verifying (exhaustive (intelligent) search)

Matching modulo

Finding a substitution σ such that

$$\sigma(l) = t$$

is called the matching problem from l to t (denoted $l \ll^? t$).

Finding a substitution σ such that

$$\sigma(l) =_E t$$

is called the matching problem from l to t (denoted $l \ll^?_E t$).

Examples (commutative symbol(s))

$\mathcal{F} = \{a(0), b(0), c(0), f(2), g(2), h(1)\}$
f is assumed to be commutative (the other symbols have no property).

$$C(f): \quad \forall x, y \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \quad f(x, y) = f(y, x)$$

- $f(a, b) = f(b, a)$ — yes
- $g(f(a, b), a) = g(f(b, a), a)$ — yes
- $g(f(a, b), a) = g(a, f(b, a))$ — no
- $f(a, f(a, b)) = f(f(b, a), a)$ — yes
- $f(a, f(b, c)) = f(f(c, b), a)$ — yes
- $f(f(a, b), c) = f(a, f(b, c))$ — no

Matching modulo C : examples

Solve the following problems :

- $f(x, y) \ll^?_C f(a, b)$
  $\sigma = \{x \mapsto a, y \mapsto b\}$
  $\sigma = \{x \mapsto b, y \mapsto a\}$

- $f(y, f(x, x)) \ll^?_C f(f(f(a, b), f(b, a)), f(b, a))$
  $\sigma = \{x \mapsto a, y \mapsto f(f(a, b), f(b, a))\}$
  $\sigma = \{x \mapsto b, y \mapsto f(f(a, b), f(b, a))\}$
  $\sigma = \{x \mapsto f(a, b), y \mapsto f(a, b)\}$

Matching modulo C : A rule based description

Delete		$t \ll^? t \wedge P$	
	$\mapsto\!\!\!\!\rightarrow$	P	
Decomposition		$f(t_1, \ldots, t_n) \ll^? f(t'_1, \ldots, t'_n) \wedge P$	
	$\mapsto\!\!\!\!\rightarrow$	$\bigwedge_{i=1,\ldots,n} t_i \ll^? t'_i \wedge P$	
SymbolClash		$f(t_1, \ldots, t_n) \ll^? g(t'_1, \ldots, t'_m) \wedge P$	
	$\mapsto\!\!\!\!\rightarrow$	Fail	*if* $f \neq g$
SymbolVariableClash		$f(t_1, \ldots, t_n) \ll^? x \wedge P$	
	$\mapsto\!\!\!\!\rightarrow$	Fail	*if* $x \in \mathcal{X}$
MergingClash		$x \ll^? t \wedge x \ll^? t' \wedge P$	
	$\mapsto\!\!\!\!\rightarrow$	Fail	*if* $t \neq t'$

Assume + commutative

$$\begin{array}{ll} C-Decomposition & t_1 + t_2' \ll^?_C t_1' + t_2' \ \wedge \ P \\ \mapsto\!\!\!\!\rightarrow & (t_1 \ll^?_C t_1' \wedge t_2 \ll^?_C t_2') \vee (t_1 \ll^?_C t_2' \wedge t_2 \ll^?_C t_1') \end{array}$$

Find a match

$$x*(3+y) \ll_C^? 1*(4+3)$$

$$\rightarrow_{Decomposition} x \ll_C^? 1 \ \wedge \ 3+y \ll_C^? 4+3$$

$$\rightarrow_{C(+)-Decomposition} x \ll_C^? 1 \wedge ((3 \ll_C^? 4 \wedge y \ll_C^? 3) \vee (3 \ll_C^? 3 \wedge y \ll_C^? 4))$$

$$\rightarrow_{MergingClash} x \ll_C^? 1 \ \wedge \ (\text{Fail} \vee (3 \ll_C^? 3 \ \wedge \ y \ll_C^? 4))$$

$$\rightarrow_{Delete} x \ll_C^? 1 \ \wedge \ (\text{Fail} \vee (y \ll_C^? 4))$$

$$\rightarrow_{Bool} x \ll_C^? 1 \ \wedge \ y \ll_C^? 4$$

Matching rules

Does it terminate ?
Do we always get the same result ?

Theorem The normal form by the rules in $Commutative - Match$, of any matching problem $t \ll^? t'$ such that $\mathcal{V}ar(t) \cap \mathcal{V}ar(t') = \emptyset$, exists and is unique.

1. If it is **Fail**, then there is no match from t to t'.
2. If it is of the form $\bigvee_{k \in K} \bigwedge_{i \in I} x_i^k \ll^?_C t_i^k$ with $I, K \neq \emptyset$, the substitutions $\sigma^k = \{x_i^k \mapsto t_i^k\}_{i \in I}$ are all the matches from t to t'.
3. If it is **empty** then t and t' are identical : $t = t'$.

Matching modulo associativity-commutativity (1)

$\cup$ is assumed to be an associative commutative (AC) symbol :

$\forall x, y, z,\ \cup(x, \cup(y, z)) = \cup(\cup(x, y), z)$ and $\forall x, y, \cup(x, y) = \cup(y, x)$.

$$\{i\} \cup s \ll^{?}_{AC} \{1\} \cup \{2\} \cup \{3\} \cup \{4\} \cup \{5\}$$

$$\begin{aligned} &\{1\} \cup \{2\} \cup \{3\} \cup \{4\} \cup \{5\} && =_{AC} \\ &\{2\} \cup \{3\} \cup \{4\} \cup \{5\} \cup \{1\} && =_{AC} \\ &\ldots \\ &\{5\} \cup \{1\} \cup \{2\} \cup \{3\} \cup \{4\} \end{aligned}$$

5 different and non *AC*-equivalent matches.

Rewriting modulo : definition

A **class rewrite system** R/A is composed of a set of rewrite rules R and a set of equalities A, such that A and R are disjoint sets.

$$
\begin{aligned}
x + 0 &\to x \\
x + (0 + y) &\to x + y \\
x + (-x) &\to 0 \\
x + ((-x) + y) &\to y \\
- - x &\to x \\
-0 &\to 0 \\
-(x + y) &\to (-x) + (-y)
\end{aligned}
$$

$$
\begin{aligned}
x + y &= y + x \\
(x + y) + z &= x + (y + z)
\end{aligned}
$$

$\rightarrow_{R/A}$

t (R/A)-rewrites to t' if $t =_A t_1 \rightarrow_R t_2 =_A t'$

To be more effective, consider any relation $\rightarrow_{RA}$ such that :

$$\rightarrow_R \subseteq \rightarrow_{RA} \subseteq \rightarrow_{R/A}$$

$\longrightarrow_{R,A}$

A term rewrite system R (a set of rewrite rules) determines a relation on terms denoted $\longrightarrow_{R,A}$ [Peterson & Stickel,81]

$$u \longrightarrow_{R,A} v$$

iff
there exist $l \to r \in R$, an occurrence ω in t, such that

$$u_{|\omega} =_A \sigma(l)$$

and

$$v = u[\sigma(r)]_\omega$$

USUALLY, when defining the rewriting relation, one requires the all rewrite rules satisfy $Var(r) \subseteq Var(l)$.

For example

Let $\cup$ be an AC symbol, such that

$$\{i\} \cup x \to i$$

$$\begin{aligned} \{1\} \cup \{2\} \cup \{3\} \cup \{4\} \cup \{5\} &=_{AC} \\ \{2\} \cup \{3\} \cup \{4\} \cup \{5\} \cup \{1\} &=_{AC} \\ \ldots & \\ \{5\} \cup \{1\} \cup \{2\} \cup \{3\} \cup \{4\} & \end{aligned}$$

Since this term matches the lefthand side of the rewriting rule in 5 different and non *AC*-equivalent ways, the rewrite rule applies in 5 different ways.

Examples

Assume $+$ to be AC (associative and commutative)

$$R = \{a + a \to a\}$$

R/E-rewrite the term $(a + c) + a$ a+c

R, E-rewrite the term $(a + c) + a$

$$R = \{a + a \to a \quad (a + a) + x \to a + x\}$$

R/E-rewrite the term $(a + c) + a$ a+c

R, E-rewrite the term $(a + c) + a$ a+c

Communication protocol : is this a good one ?

may be not . . .

Needham-Schroeder public-key protocol

The Needham-Schroeder public-key protocol aims to establish a mutual authentication between an initiator and a responder that communicate via an insecure network.

		Initiator	Responder	Net
1.	$A \rightarrow B : \{N_A, A\}_{K(B)}$	A^{SLEEP}	B^{SLEEP}	$\emptyset$
		A^{WAIT}	B^{SLEEP}	$\{N_A, A\}_{K(B)}$
2.	$B \rightarrow A : \{N_A, N_B\}_{K(A)}$	A^{WAIT}	$B^{\{N_A,A\}_{K(B)}}$	$\emptyset$
		A^{WAIT}	B^{WAIT}	$\{N_A, N_B\}_{K(A)}$
3.	$A \rightarrow B : \{N_B\}_{K(B)}$	$A^{\{N_A,N_B\}_{K(A)}}$	B^{WAIT}	$\emptyset$
		A^{COMMIT}	B^{WAIT}	$\{N_B\}_{K(B)}$
4.	N_B *is the session key*	A^{COMMIT}	B^{COMMIT}	$\emptyset$

Rewrite rules for the agents

➲ initiator starts the communication with a responder

```
[initiator-1]
x+SLEEP+resp   || E <> y+std+init || D <> I <> ls                    =>
x+WAIT+N(x,y) || E <> y+std+init || D <> I <>
                        x-->y:K(y)[N(x,y),DN,A(x)] & ls
end
```

Data structures

Rewrite rules for the agents

➲ responder reads the message and sends the acknowledgement

```
[responder-1]
E <> y+SLEEP+init   || D <> I <> w-->y:K(y)[N(n1,n3),N(n2,n4),A(z)] & ls=>
E <> y+WAIT+N(y,z) || D <> I <> y-->z:K(z)[N(n1,n3),N(y,z),A(y)] & ls
end
```

Rewrite rules for the agents

➲ initiator receives the acknowledgement and checks its validity

```
[initiator-2]
x+WAIT+N(x,v) || E <> D <> I <> w-->x:K(x)[N(x,v),N(n2,n4),A(z)] & ls =>
x+COMMIT+N(x,v) || E <> D <> I <> x-->v:K(v)[N(n2,n4),DN,DA] &ls
end

[initiator-2]
x+WAIT+N(x,v) || E <> D <> I <> w-->x:K(x)[N(n1,n3),N(n2,n4),A(z)] &ls =>
ERROR
      if x!=n1 or v!=n3
end
```

Rewrite rules for the intruder

➲ the intruder intercepts all the messages in the network but the messages generated by itself and stores or decrypts them.

```
[intruder-1]
   E <> D <> w#l#ll <> z-->x:K(w)[N(n1,n3),N(n2,n4),A(v)] & ls  =>
   E <> D <> w#N(n1,n3) | N(n2,n4) | l#ll <> ls
                  if w!=z
end

[intruder-1]
   E <> D <> w#l#ll <> !w-->x:K(w)[N(n1,n3),N(n2,n4),A(v)] & ls  =>
   E <> D <> w#N(n1,n3) | N(n2,n4) | l#ll <> ls
end
```

Rewrite rules for the intruder

➲ the nonces obtained previously by the intruder are used in order to generate fake messages that are sent to all the agents.

```
[intruder-4]
   E <> D <> w # resp | l # ll <> ls                              =>
   E <> D <> w # l # ll <> w-->y:K(y)[resp,DN,A(xadd)] & ls
               where (Agent)y+std+dn :=(extAgent) elemIA(D || E)
               where (Agent)xadd+std1+dn1 :=(extAgent) elemIA(D || E)
end
```

Generalized rules

The invariants

➲ **authenticity of the responder :** if an initiator `x` committed with a responder `y`, then `y` has really been involved in the protocol.

```
[attack-1] x+COMMIT+N(x,y) || E <> D <> i#l#ll <> ls =>
           ATTACK
                if y!=i
                if not(existAgent(y+WAIT+N(y,x),D)) and
                   not(existAgent(y+COMMIT+N(y,x),D))
end
```

➲ **authenticity of the initiator :** if a responder `y` committed with an initiator `x` then the initiator have committed as well with `y`.

```
[attack-2] E <> y+COMMIT+N(y,x) || D <> i#l#ll <> ls =>
           ATTACK
                if x!=i
                if not(existAgent(x+COMMIT+N(x,y),E))
end
```

The strategy

We apply repeatedly all the rewrite rules in any order and in all the possible ways until one of the attack rules can be applied.

```
[]attStrat  =>  repeat*(
                   dk(
                      attack-1, attack-2,
                      intruder-1, intruder-2, intruder-3, intruder-4,
                      initiator-1, initiator-2, responder-1, responder-2
                   )
                );
                attackFound
end
```

where

```
[attackFound]   ATTACK   =>   ATTACK              end
```

The attack

I.1.	$A \rightarrow I$	$: \{N_A, A\}_{K(I)}$
II.1.	$I(A) \rightarrow B$	$: \{N_A, A\}_{K(B)}$
II.2.	$B \rightarrow I(A)$	$: \{N_A, N_B\}_{K(A)}$
I.2.	$I \rightarrow A$	$: \{N_A, N_B\}_{K(A)}$
I.3.	$A \rightarrow I$	$: \{N_B\}_{K(I)}$
II.3.	$I(A) \rightarrow B$	$: \{N_B\}_{K(B)}$

The corrected protocol

1. $A \rightarrow B : \{N_A, A\}_{K(B)}$
2. $B \rightarrow A : \{N_A, N_B, B\}_{K(A)}$
3. $A \rightarrow B : \{N_B\}_{K(B)}$

Modified rule : `initiator-2`

```
[initiator-2]
x+WAIT+N(x,v) || E <> D <> I <> w-->x:K(x)[N(x,v),N(n2,n4),A(v)] & ls  =>
x+COMMIT+N(x,v) || E <> D <> I <> x-->v:K(v)[N(n2,n4),DN,DA] & ls
end
```

Think abstractly

The properties of this relation could be studied in an abstract way :

$\Rightarrow$ Abstract rewrite systems

Showing normalization

A (partial) order on $\mathcal{T}$ is a reflexive, antisymetric and transitive relation.

An ordering is **total** on $\mathcal{T}$ when two terms are always comparable

$>$ is well-founded or Noetherian on $\mathcal{T}$ if there is no infinite decreasing sequence on $\mathcal{T}$:

$$t_1 > t_2 > t_3 > \dots$$

Theorem
Consider an ARS $(\mathcal{A},\rightarrow)$.
$\rightarrow$ is terminating
iff
there exists a well-founded (partial) order $>$ on $\mathcal{T}$ and a mapping ϕ s.t.
for all rewrite rule $a \rightarrow a'$ implies $\phi(a) > \phi(a')$.

Example

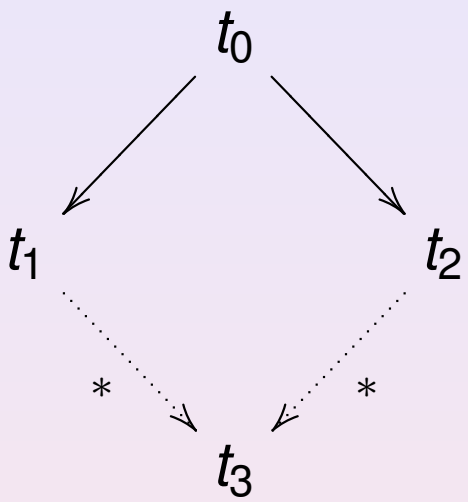
Use the order $(>,\mathbb{N})$ which is well-founded.

Several choices for strings $\mathcal{A} = (\bullet \mid \circ)^*$

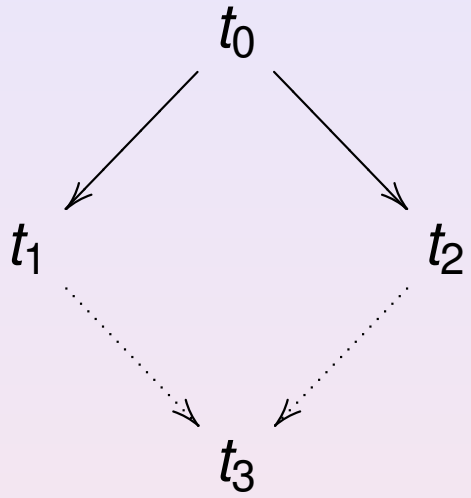
- $\phi(w)$ = number of $\bullet$
  works for all $\bullet$-decreasing reductions
- $\phi(w)$ = number of $\circ$
  works for all $\circ$-decreasing reductions
- $\phi(w)$ = number of $\bullet$ and $\circ$
  works for all length-decreasing reductions

Definitions (◂ Relathionships)

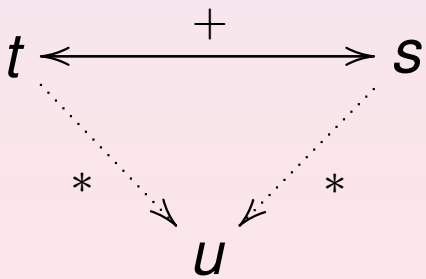
Localy confluent (LC)



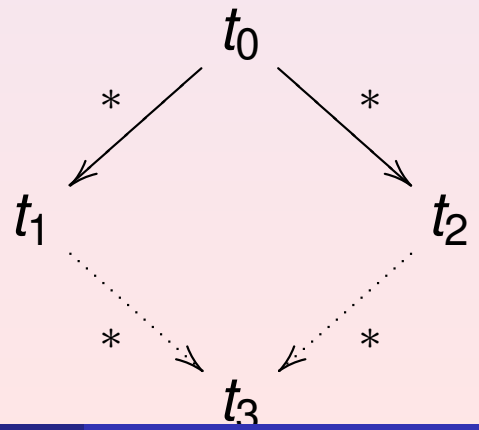
Diamond property (DP)



Church Rosser (CR)

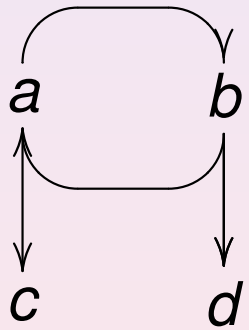


Confluent (C)



Local versus global confluence

1. $C \Rightarrow LC$
2. $LC \Rightarrow C$?
   - Consider four distinct elements a, b, c, d of $\mathcal{T}$ and the relation :
     $a \longrightarrow b$
     $b \longrightarrow a$
     $a \longrightarrow c$
     $b \longrightarrow d$



Newman's lemma

[Newman 1942]

Provided the relation $\rightarrow$ is terminating

then

$\rightarrow$ is confluent iff it is locally confluent

Proof :

- locally confluent if confluent
  ➥ obvious
- confluent if locally confluent
  ➥ ?

Termination

R (or $\rightarrow_R$) terminates
iff all derivation issued from any term terminate.

Termination implies the existence of normal form(s) for any term.

Termination is in general undecidable
but interesting sufficient condition can be found.

Proving termination could be tricky . . .

$$f(a, b, x) \rightarrow f(x, x, x)$$

is terminating

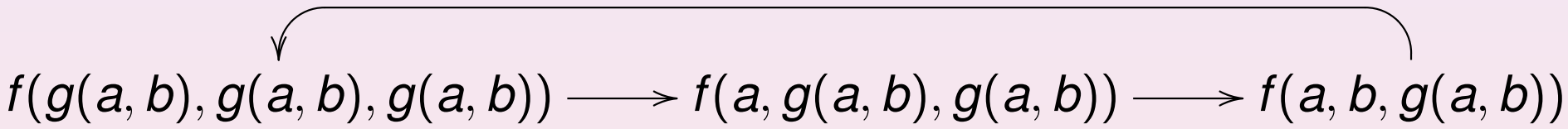
$$\begin{aligned} g(x, y) &\rightarrow x \\ g(x, y) &\rightarrow y, \end{aligned}$$

is terminating

Is the union terminating ?

$$\begin{aligned} f(a,b,x) &\to f(x,x,x) \\ g(x,y) &\to x \\ g(x,y) &\to y, \end{aligned}$$

We have the derivation :



[Toyama 1986]

Orderings on terms

A Reduction ordering is an ordering on $\mathcal{T}$,
stable by context and substitution :
➥ for every context $C[\_]$ and for all substitutions σ,
if $t > s$ then $C[t] > C[s]$ and $\sigma(t) > \sigma(s)$.

Theorem R terminates iff there exists a well-founded reduction ordering $>$ s.t. for all rewrite rule $(l \to r) \in R$, $l > r$.

Example

The rules of the game :

$$\bullet\bullet \longrightarrow \circ$$
$$\circ\circ \longrightarrow \circ$$
$$\bullet\circ \longrightarrow \bullet$$
$$\circ\bullet \longrightarrow \bullet$$

$l > r$ if $|l| > |r|$

$l > r$ if $|l|_{\bullet\circ} > |r|_{\bullet\circ}$

($|t|_{\bullet\circ}$=number of $\bullet$ and $\circ$ of the term t built out of $\bullet$ and $\circ$)

$|f(f(x,x),y)| > f(y,y)$

but

$|f(f(x,x),f(x,x))| \not> f(f(x,x),f(x,x))$

Example

The rules of the game :

$$\bullet\bullet \longrightarrow \circ\circ$$
$$\circ\circ \longrightarrow \circ$$
$$\bullet\circ \longrightarrow \bullet$$
$$\circ\bullet \longrightarrow \bullet$$

$l > r$ if $|l|_{\bullet\circ} > |r|_{\bullet\circ}$

$$|\bullet\bullet|_{\bullet\circ} = 2 \not> 2 = |\circ\circ|_{\bullet\circ}$$

$l > r$ if $|l|_{\bullet} > |r|_{\bullet}$

$$|\circ\circ|_{\bullet} = 0 \not> 0 = |\circ|_{\bullet}$$

Example

The rules of the game :

$$\bullet\bullet \longrightarrow \circ\circ$$
$$\circ\circ \longrightarrow \bullet$$
$$\bullet\circ \longrightarrow \bullet$$
$$\circ\bullet \longrightarrow \bullet$$

$l > r$ if $|l|_{\bullet\circ} > |r|_{\bullet\circ}$

$$|\bullet\bullet|_{\bullet\circ} = 2 \not> 2 = |\circ\circ|_{\bullet\circ}$$

$l > r$ if $|l|_{\bullet} > |r|_{\bullet}$

$$|\circ\circ|_{\bullet\circ+\bullet} = 2 \not> 2 = |\bullet|_{\bullet\circ+\bullet}$$

Lexicographical extensions

Let $>$ be an ordering on $\mathcal{T}$.
Its **lexicographical extension** $>^{lex}$ on $\mathcal{T}^n$ is defined as :

$$(s_1, \ldots, s_n) >^{lex} (t_1, \ldots, t_n)$$

if there exists i, $1 \leq i \leq n$ s.t. $s_i >_i t_i$, and $\forall j, 1 \leq j < i, s_j = t_j$.

If $>$ is well-founded on $\mathcal{T}$, then $>^{lex}$ is well-founded on $\mathcal{T}^n$.

FALSE for an infinite product of ordered sets :
$\mathcal{T} = \{a, b\}$ with $a < b$

$$b >^{lex} ab >^{lex} aab >^{lex} aaab >^{lex} \ldots$$

Well-founded reduction orderings

- Syntactic
  Based on the precedence concept (i.e. a partiel order $>_{\mathcal{F}}$ on $\mathcal{F}$)
  Example : Recursive or Lexicographic path ordering [Dershowitz, 82]
- Semantic
  Terms are interpreted in another structure where a well-founded ordering is known (e.g. the natural numbers)
  Example : Polynomial interpretations
- Combinations
  Ordering combining semantical and syntactical behavior
- Recursion analysis
  Induction, dependency pairs

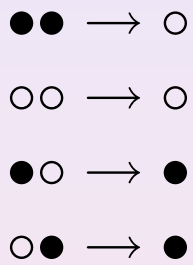
Confluence

Allows us to forget about non-determinism :

Whatever rewriting is done we will converge later.

Back with the simple game

The rules of the game :

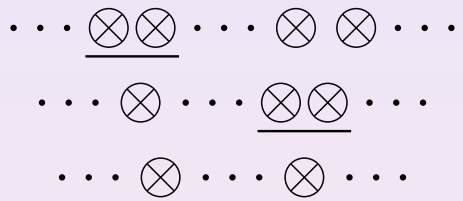


A starting point :

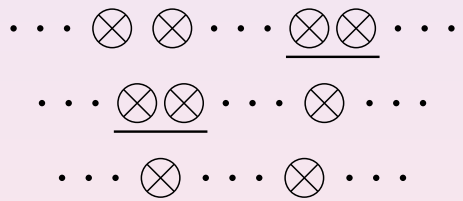
From a given start, is the result deterministic ?

Analysing the different cases

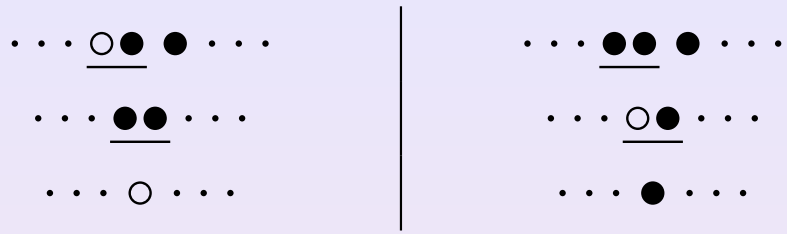
Disjoint redexes :



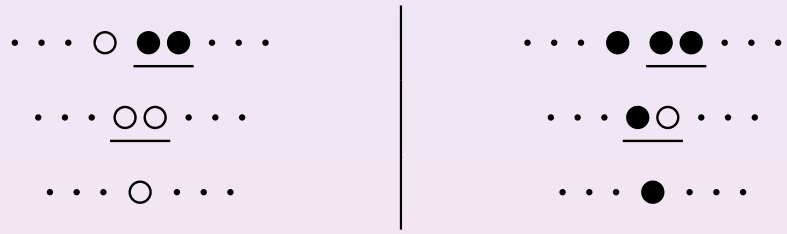
is the same as :



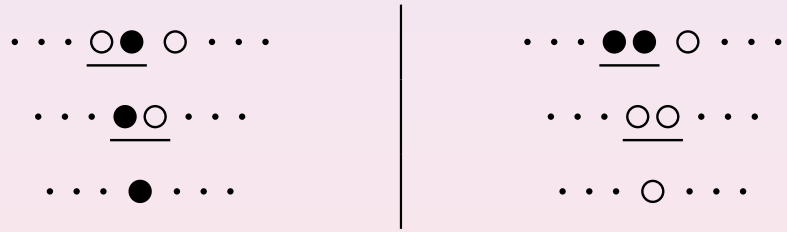
No disjoint redexes (central black) :



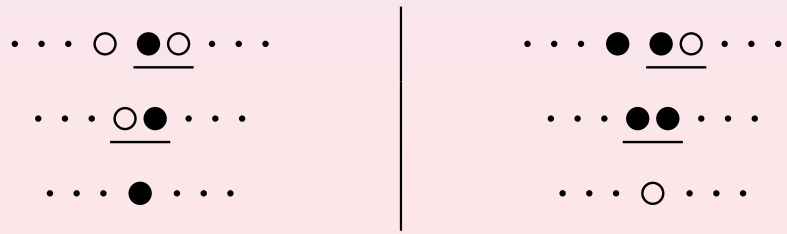
but



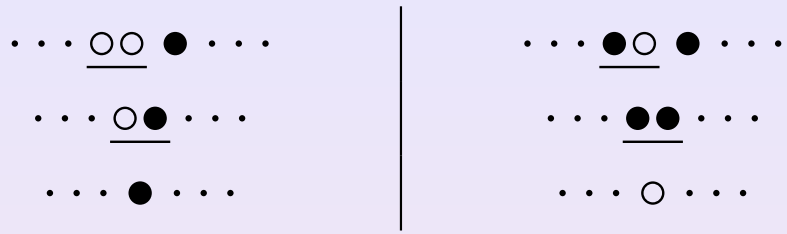
or



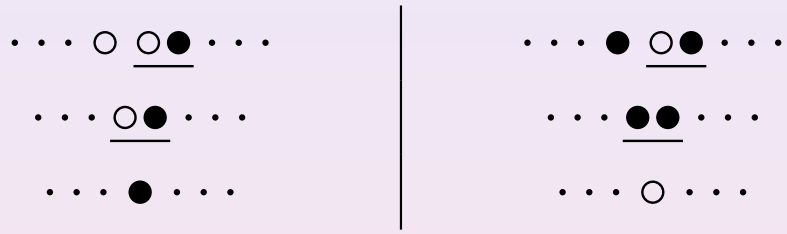
but



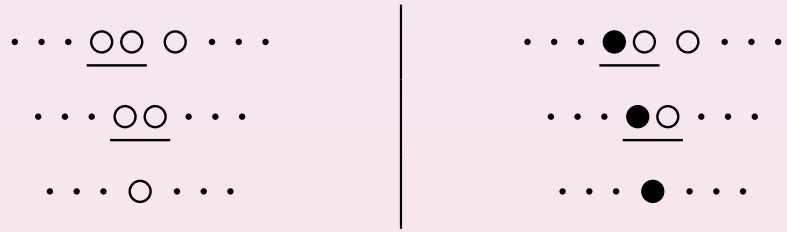
No disjoint redexes (central white) :



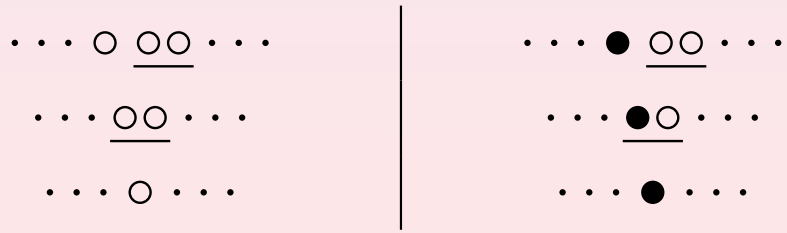
but



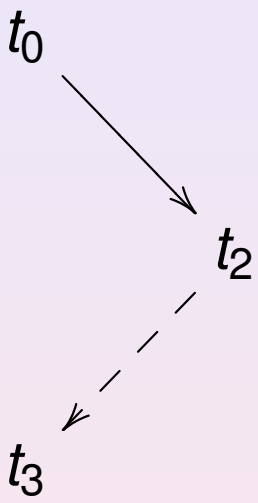
or



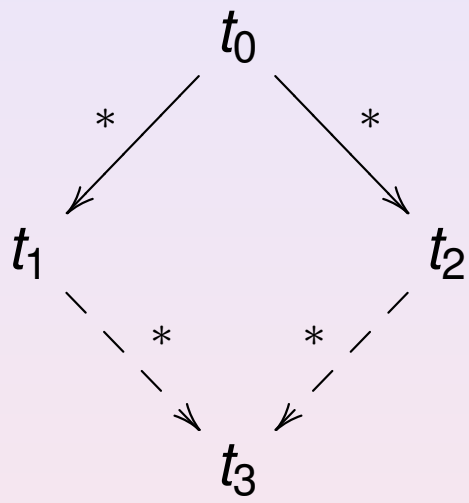
but



Thus :



but what about :



Confluence

➥ Undecidable in general, confluence is decidable for finite and terminating rewrite systems.

➥ Assuming termination of the rewrite relation, its confluence is equivalent to the confluence of critical pairs.

➥ If a rewrite system is orthogonal (linear and non-overlapping), then it is confluent.

Other systems

What if the system is non-teminating and non-orthogonal ?

Theorem Consider a reduction relation $\longrightarrow_R$ and let $\longrightarrow_D$ s.t.

$\longrightarrow_R \subseteq \longrightarrow_D \subseteq \overset{*}{\longrightarrow}_R$

$\longrightarrow_D$ has the diamond property

Then, $\longrightarrow_R$ is confluent.

References on rewriting modulo

- G. Huet. Confluent reductions : Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4) :797–821, October 1980.
- G. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28 :233–264, 1981.
- J.-P. Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, 15(4) :1155–1194, 1986.
- Enno Ohlebusch. Church-Rosser Theorems for Abstract Reduction Modulo an Equivalence Relation RTA, pages 17-31, LNCS 1379, 1998.
- Claude and Hélène Kirchner. Rewriting Solving Proving `www.loria.fr/~ckirchne/rsp.ps.gz`

Why a new calculus ?

Rewriting is nice, but

- the rewrite relation or the rewriting logic are difficult to control
- non-reducibility is impossible to express

Lambda-calculus is great, but

- lacks of discrimination capabilities

A “simple” λ-term...

$$(\lambda Y.\Big((\lambda y.(y x_{\perp}(\lambda X.X)))Y\Big))\Big((\lambda x_1.\lambda z_1\lambda z_2.(z_2 x_1))(\lambda u_1\lambda u_2.u_1)\Big)$$

. . . and its meaning

$$\overbrace{\underbrace{[\![\lambda Y.(f(X) \rightarrowtriangle X)\ Y]\!]}_{[\![f(X) \rightarrowtriangle X]\!]}}^{(\lambda Y.(\overbrace{(\lambda y.(y x_\perp (\lambda X.X)))}^{[\![f]\!]}\ Y))} \overbrace{\underbrace{[\![f(a)]\!]}_{[\![f]\!]}\ \underbrace{[\![a]\!]}_{(\lambda u_1 \lambda u_2.u_1)}}^{(\lambda x_1.\lambda z_1 \lambda z_2.(z_2 x_1))}$$

Simple encoding of rewriting in the λ-calculus

$$
\begin{array}{ll}
 & \overbrace{\llbracket \lambda Y.(f(X) \twoheadrightarrow X)\ Y \rrbracket}^{\llbracket f(X) \twoheadrightarrow X \rrbracket}\ \overbrace{\llbracket f(a) \rrbracket}^{\llbracket f \rrbracket\ \llbracket a \rrbracket} \\
 & (\lambda Y.(\overbrace{(\underline{\lambda y}.(y x_\perp (\lambda X.X)))}\ \underline{Y}))\ (\overbrace{(\lambda x_1.\lambda z_1 \lambda z_2.(z_2 x_1))}\overbrace{(\lambda u_1 \lambda u_2.u_1)}) \\
\mapsto_\beta & (\lambda Y.(Y x_\perp (\lambda X.X)))\ ((\underline{\lambda x_1}.\lambda z_1 \lambda z_2.(z_2 x_1))\underline{(\lambda u_1 \lambda u_2.u_1)}) \\
\mapsto_\beta & (\underline{\lambda Y}.(Y x_\perp (\lambda X.X)))\ \underline{(\lambda z_1 \lambda z_2.(z_2 (\lambda u_1 \lambda u_2.u_1)))} \\
\mapsto_\beta & (\underline{\lambda z_1} \lambda z_2.(z_2 (\lambda u_1 \lambda u_2.u_1)))\ \underline{x_\perp}\ (\lambda X.X) \\
\mapsto_\beta & (\underline{\lambda z_2}.(z_2 (\lambda u_1 \lambda u_2.u_1)))\ \underline{(\lambda X.X)} \\
\mapsto_\beta & (\underline{\lambda X}.X)\underline{(\lambda u_1 \lambda u_2.u_1)} \\
\mapsto_\beta & (\lambda u_1 \lambda u_2.u_1) \\
= & \llbracket a \rrbracket
\end{array}
$$

Term rewriting

$$f(x, y) \to x$$

$$f(a, b) \Longrightarrow_{\mathcal{R}} a$$

Rewriting calculus - abstraction

$$f(X, Y) \rightarrow X$$

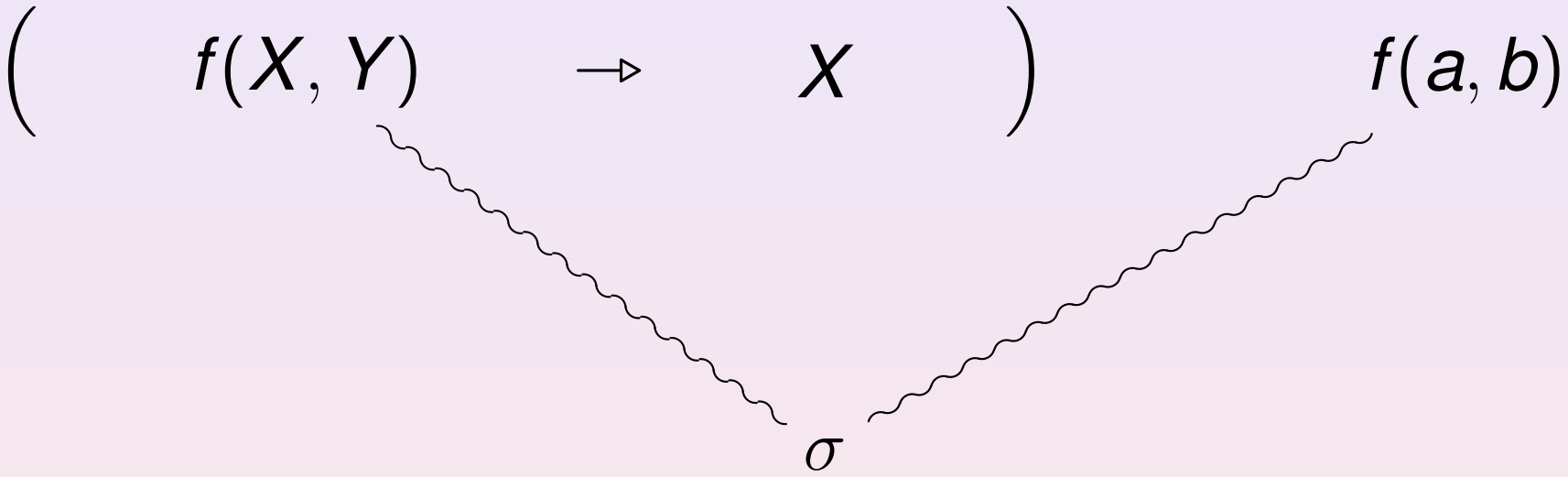
↪ Abstraction Operator

Rewriting calculus - application

$$\left(f(X, Y) \rightarrowtriangle X \right) \; f(a, b)$$

➥ Application Operator

Rewriting calculus - compute the substitution



Rewriting calculus - replacement

$$\left(\quad f(X, Y) \quad \rightarrowtriangle \quad X \quad \right) \quad f(a, b)$$

σ

$= \{X \mapsto a, Y \mapsto b\}$

Rewriting calculus - result

$$\left(\quad f(X, Y) \quad \rightarrowtriangle \quad X \quad \right) \quad f(a, b)$$

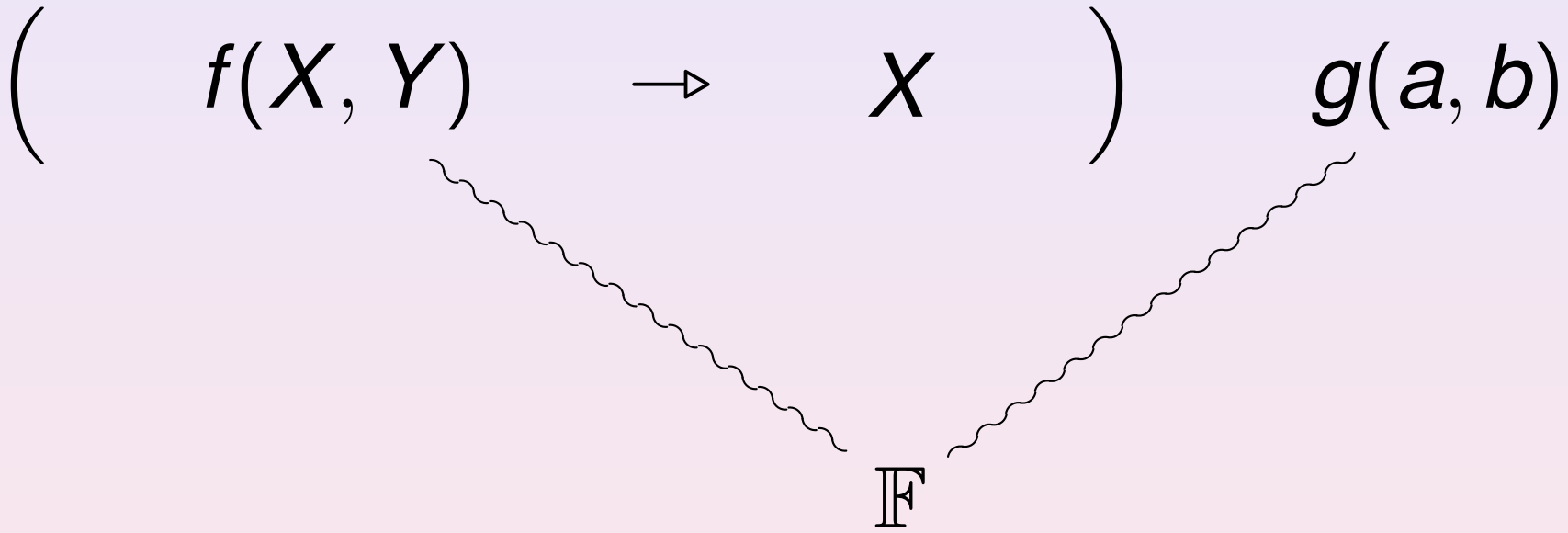
$$\mapsto \sigma(X) \textit{ i.e. } a$$

For the rewriting relation

$$f(x, y) \longrightarrow x$$

$$g(a, b) \Longrightarrow_{\mathcal{R}}$$

For the rewriting calculus



A calculus with more explicit features

Like for explicit substitution calculi, explicitation of all the ingredients is useful.
In “basic” rewriting calculus,

- rules are first class object
- application is explicit
- decision of redex reduction is explicit
- matching is a main explicit parameter
- results are first class

The Untyped Syntax

$$\mathcal{P} ::= \mathcal{T} \qquad \text{Patterns}$$

$$\mathcal{T} ::= \mathcal{X} \mid \mathcal{K} \mid \mathcal{P} \twoheadrightarrow \mathcal{T} \mid \mathcal{T}\ \mathcal{T} \mid \mathcal{T} \wr \mathcal{T} \qquad \text{Terms}$$

1. $T_1 \twoheadrightarrow T_2$ is a *rule abstraction* with pattern T_1 and body T_2
   ... the free variables of T_1 are bound in T_2
2. The terms can be also *structures* built using the symbol "$\wr$"
3. We work modulo the α*-convention* and the *hygiene-convention*

Some ρ-terms

- $(X \to X)\, a$ — similar to the λ-term $(\lambda x.x)\, a$
- $(X \to X\, X)\, (X \to X\, X)$ — the well-known λ-term $(\omega\omega)$
- $(a \to b)\, a$ — the application of the rule $a \to b$ to the term a
- $(f(X, Y) \to g(X, Y))\, f(a, b)$ — a classical rewrite rule application

The Simplest Reduction Semantics

$$(P \rightarrowtriangle A)\ B \ \rightarrow_{\rho} A\theta_{(P \lessdot B)} \qquad \text{if } P\theta =_{\mathbb{T}} B$$

Some ρ-reductions

- $(X \to X)\, a \quad \mapsto\!\!\!\!\to_{\rho} a$
- $(X \to (X\, X))\, (X \to (X\, X)) \quad \mapsto\!\!\!\!\to_{\rho} \{\omega\, \omega\} \mapsto\!\!\!\!\to_{\rho\omega} \ldots$
- $(a \to b)\, a \quad \mapsto\!\!\!\!\to_{\rho} b$
- $(f(X, Y) \to g(X, Y))\, (f(a, b)) \quad \mapsto\!\!\!\!\to_{\rho} g(a, b)$
- $(f(X, Y) \to g(X, Y))\, (g(a, b))$

Non unitary matching

$$\left(f_{AC}(X, Y) \twoheadrightarrow X \right) \quad f_{AC}(a, b)$$

$$\mapsto^{?} a$$
$$\mapsto^{?} b$$

Reduction produces structures

$$(P \rightarrowtriangle A)\, B \;\rightarrow_{\rho}\; A\theta_1 \wr \ldots \wr A\theta_n, \ldots$$

$$\text{with } \{\theta_1, \ldots, \theta_n, \ldots\} = \mathcal{S}ol(P \lessdot_{\mathbb{T}} B)$$

Nondeterminism

For the rewriting reduction

$$\begin{cases} f(x,y) \to x \\ f(x,b) \to b \end{cases}$$

$$f(x,y) \to x$$

$$f(a,b) \Longrightarrow_{\mathcal{R}} a$$

Non Determinism

$$f(a,b) \Longrightarrow_{\mathcal{R}} b$$

$$f(x,b) \to b$$

Basic ρ-calculus

$$(P \twoheadrightarrow A)\, B \;\to_{\rho}\; A\theta_1 \wr \ldots \wr A\theta_n, \ldots$$

$$\text{with } \{\theta_1, \ldots, \theta_n, \ldots\} = \mathcal{S}ol(P \ll_{\mathbb{T}} B)$$

$$(A \wr B)\, C \;\to_{\partial}\; A\, C \wr B\, C$$

Detecting matching failures : the symbol *stk*

1. The relation $P \not\sqsubseteq A$ detects (some) definitive matching failures :

   $\forall \sigma, \forall B$ s.t. $\sigma(A) \mapsto B$, $\nexists \tau$ s.t. $\tau(P) = B$

2. The relation $\rightarrow_{stk}$ treats matching failures uniformly :

$$
\begin{aligned}
(P \rightarrowtriangle B)\ A &\rightarrow_{stk} stk \qquad \text{if } P \not\sqsubseteq A \\
stk \wr A &\rightarrow_{stk} A \\
A \wr stk &\rightarrow_{stk} A \\
stk\ A &\rightarrow_{stk} stk
\end{aligned}
$$

Failures

$$\left(f(X,Y) \to X \wr f(X,c) \to c\right) f(a,b)$$

- $(f(X,Y) \to X)\, f(a,b) \wr (f(X,c) \to c)\, f(a,b)$
- $a \wr stk$
- a

ρ-calculus and objects

Object = record with an explicit account of self

$$[m_i = \varsigma(\mathcal{X}_i)\, T_i]^{i\in I} \quad \triangleq \quad (m_i \twoheadrightarrow \mathcal{X}_i \twoheadrightarrow T_i)^{i\in I}$$

Self-application = the application of an object to the object itself

$$T_1.T_2 \quad \triangleq \quad T_1\ T_2\ T_1$$

Ex : $T \triangleq a \twoheadrightarrow S \twoheadrightarrow b$. Then : $T.a \triangleq T\ a\ T \ \longmapsto\!\!\!\!\!\rightarrow_{\rho\mathfrak{dB}} \ (S \twoheadrightarrow b)\ T \ \longmapsto\!\!\!\!\!\rightarrow_{\rho\mathfrak{dB}} \ b$

Ex : $T \triangleq \omega \twoheadrightarrow S \twoheadrightarrow S.\omega$. Then :
$T.\omega \ \longmapsto\!\!\!\!\!\rightarrow_{\rho\mathfrak{dB}} \ (S \twoheadrightarrow S.\omega)\ T \ \longmapsto\!\!\!\!\!\rightarrow_{\rho\mathfrak{dB}} \ T.\omega \ \longmapsto\!\!\!\!\!\rightarrow_{\rho\mathfrak{dB}} \ \ldots$

Records

A “ping-pong” object

Let $T \triangleq (ping \twoheadrightarrow S \twoheadrightarrow S.pong \wr pong \twoheadrightarrow S \twoheadrightarrow S.ping)$

$$
\begin{array}{rcl}
T.ping & \triangleq & T\ ping\ T \\
& \mapsto_{\rho\omega} & ((ping \twoheadrightarrow S \twoheadrightarrow S.pong)\ ping \wr \\
& & (pong \twoheadrightarrow S \twoheadrightarrow S.ping)\ ping)\ T \\
& \mapsto_{\rho\omega} & ((S \twoheadrightarrow S.pong) \wr stk)\ T \\
& \mapsto_{\rho\omega} & (S \twoheadrightarrow S.pong)\ T \wr stk\ T \\
& \mapsto_{\rho\omega} & (S \twoheadrightarrow S.pong)\ T \wr stk \\
& \mapsto_{stk} & (S \twoheadrightarrow S.pong)\ T \\
& \mapsto_{\rho\omega} & T.pong \\
& \mapsto_{\rho\omega} & T.ping \\
& \mapsto_{\rho\omega} & \ldots
\end{array}
$$

Functional object update

Update $(a.m := b) \triangleq (a \wr m \rightarrow b)$

$$\begin{aligned} \mathit{Point} \triangleq\ & \mathit{val} \rightarrow S \rightarrow v(1,1) \wr \\ & \mathit{get} \rightarrow S \rightarrow S.\mathit{val} \wr \\ & \mathit{set} \rightarrow S \rightarrow v(X,Y) \rightarrow (S.\mathit{val} := S' \rightarrow v(X,Y)) \end{aligned}$$

Then :

$$\begin{aligned} \mathit{Point}.\mathit{get} &\longmapsto\!\!\!\!\!\rightarrow_{\rho\partial} v(1,1) \\ \mathit{Point}.\mathit{set}(v(2,2)) &\longmapsto\!\!\!\!\!\rightarrow_{\rho\partial} \mathit{Point} \wr (\mathit{val} \rightarrow S' \rightarrow v(2,2)) \\ \mathit{Point}.\mathit{set}(v(2,2)).\mathit{get} &\longmapsto\!\!\!\!\!\rightarrow_{\rho\partial} v(1,1) \wr v(2,2) \end{aligned}$$

Imperative object update

$Kill_m$ rule :

$$kill_m \triangleq (m \rightarrow X \wr Y) \rightarrow Y$$

Update $(a.m := b) \triangleq (kill_m(a) \wr m \rightarrow b)$

Then :

$$\begin{aligned} Point_l.get &\longmapsto\!\!\!\!\!\rightarrow_{\rho\partial\delta} v(1,1) \\ Point_l.set(v(2,2)) &\longmapsto\!\!\!\!\!\rightarrow_{\rho\partial\delta} val \rightarrow S' \rightarrow v(2,2) \wr get \rightarrow \ldots \wr set \rightarrow \ldots \\ Point_l.set(v(2,2)).get &\longmapsto\!\!\!\!\!\rightarrow_{\rho\partial\delta} v(2,2) \end{aligned}$$

Daemon

Inheritance

(Well-typed) Encoding of Rewriting in the ρ-calculus

- rewrite rules and their application,
  - ρ-abstractions and applications (Simple Encoding)
- a construction grouping together a set of rewrite rules,
  - structures and objects
- an iteration operator that applies *repeatedly* a set of rewrite rules,
  - self application
- an operator testing if a set of rewrite rules is *applicable* to a term.
  - the symbol *stk*

Encoding rewriting derivations

1. A rewrite system $\mathcal{R}$ can be represented as the structure containing all the rules

$$\big(f(X,Y) \rightharpoonup X \wr f(X,b) \rightharpoonup b\big)$$

2. Derivations can be simply encoded

$$\big(f(X,Y) \rightharpoonup X \wr f(X,b) \rightharpoonup b\big)\ f(a,b)$$

➥ $(f(X,Y) \rightharpoonup X)\ f(a,b) \wr (f(X,b) \rightharpoonup b)\ f(a,b)$

➥ $a \wr b$

Theorem : If $T_1 \longmapsto_{\mathcal{R}} T_2$, then $\exists\ T_{\mathcal{R}}$ such that $T_{\mathcal{R}}\ T_1 \mapsto\!\!\!\!\twoheadrightarrow_{\rho\delta} T_2$

Encoding rewriting - Addition over Peano integers

$$plus \triangleq \left[\begin{array}{l} S \to add(0, y) \to y \wr \\ S \to add(suc(x), y) \to suc\big((S\ S)\ add(x, y)\big) \end{array} \right]$$

$$(plus\ \ plus)\ \ add(N, M) \ \mapsto\!\!\!\!\!\!\rightarrow_{\beta \cup stk}\ \ M + N$$

Fill in the blanks with your favorite rewrite system...provided it is convergent and ground reducible if you want completeness.

$$func \triangleq \left[\begin{array}{l} S \to\ len([\,]) \to\ 0 \ \ \wr \\ S \to\ len(Cons(x, l)) \to\ suc\big((S\ S)\ \ len(l)\big) \end{array} \right]$$

General encoding

ρ-calculus Contributors

Clara Bertolissi
Horatiu Cirstea
Germain Faure
Claude Kirchner
Luigi Liquori
Benjamin Wack

Other pattern calculi

Lambda Calculus with Patterns
Vincent van Oostrom

Pure pattern calculus
Barry Jay and Delia Kesner

. . .

Rewriting

- useful concept
  Booksss
- implemented tool ASF, ELAN, LPG, Maude, Stratego, TOM, ...
- active area of research RTA, IFIP WG1.6, RULE, WRLA, ...
- active area of application and transfert XML, Semantic Web, IlogRule, RuleBase, ...

(Some) Additional Recommended Readings

- L'intelligence et le calcul (may be translated to English ?)
  Jean-Paul Delahaye
  Look also at his web page
- Term Rewriting Systems
  Terese (M. Bezem, J. W. Klop and R. de Vrijer, eds.)
  Cambridge Univerty press, 2002
- Term *Re*writing and *all That*
  Franz Baader and Tobias Nipkow
  Cambridge Univerty press, 1998
- The Rewriting Calculus Home page
  `rho.loria.fr`
- Repository of Lectures on Rewriting and Related Topics
  `qsl.loria.fr`
- The rewriting and IFIP WG1.6 page
  `rewriting.loria.fr`

More on rewriting ...

Signature and first-order terms

$\mathcal{F}_0$ a set of symbols of arity 0 (the constants)

$\mathcal{F}_i$ a set of symbols of arity i

$\mathcal{F} = \cup_n \mathcal{F}_n$

$\mathcal{X}$ a set of arity 0 symbols called **variables**.

$\mathcal{T}(\mathcal{F}, \mathcal{X})$ is the smallest set such that :

- $\mathcal{X} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$,
- $\forall f \in \mathcal{F}, \forall t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X}) : \quad f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

$\mathcal{T}(\mathcal{F}, \emptyset) = \mathcal{T}(\mathcal{F})$ is the set of **ground terms**.

Terms as mappings : $(\mathbf{N}, .) \rightarrow \mathcal{F}$

$t = f(a + x, h(f(a, b)))$ is represented by :

position	$\mapsto$	*symbol*
$\wedge$	$\mapsto$	f
1	$\mapsto$	$+$
1.1	$\mapsto$	a
1.2	$\mapsto$	x
2	$\mapsto$	h
2.1	$\mapsto$	f
2.1.1	$\mapsto$	a
2.1.2	$\mapsto$	b

$\mathcal{D}om(t) = \{\wedge, 1, 1.1, 1.2, 2, 2.1, 2.1.1, 2.1.2\}$

Examples and (some) terminology

With the following signature :

$\mathcal{F} = \{f, a\}$ with $arity(f) = 2$, $arity(a) = 0$, $x, y, z \in \mathcal{X}$:

what are the following terms ?

$f(a, a)$ is ground,

$f(x, f(a, x))$ is not linear but

$f(x, f(y, z))$ is linear

What about the following terms ?

$f(a, a, a)$ is ill-formed (since f is of arity 2)

a is correct

$x(a)$ is ill-formed (since all variables are assumed of arity 0)

f is ill-formed (since f is of arity 2)

Subterms

$t[s]_\omega$ denotes the term t with s as **subterm** at **position** (or **occurrence**) ω.
$t|_\omega$ denotes the subterm at occurrence ω.

$$f(a + x, h(f(a, b)))|_2 = h(f(a, b))$$

Simple questions—

What is $f(f(a,b),g(a))|_{1.1}$? — a

What is $f(f(a,b),g(a))|_{\Lambda}$? — $f(f(a,b),g(a))$

What is $f(f(a,b),g(a))|_{1.2}$? — b

What is the arity of f just above ? — 2

What is the arity of a just above ? — 0

What are the variables of $f(f(a,b),g(a))|_{1.2}$? — $\emptyset$

What are the variables of $f(f(x,x),g(a))|_{1.2}$? — $\{x\}$

What are the variables of $f(f(x,x),g(a))$? — $\{x\}$

Back

Substitution

A substitution σ is a mapping from the set ot variables to the set of terms :

$$\sigma : \mathcal{X} \mapsto \mathcal{T}(\mathcal{F}, \mathcal{X})$$

It is extended as a morphism from terms to terms :

$$\sigma : \mathcal{T}(\mathcal{F}, \mathcal{X}) \mapsto \mathcal{T}(\mathcal{F}, \mathcal{X})$$

$$\sigma(f(t_1, t_2)) = f(\sigma(t_1), \sigma(t_2))$$

If $\sigma = \{x \mapsto a, y \mapsto f(a, g(z)), z \mapsto g(z)\}$, then $\sigma(f(x, f(a, z))) = f(a, f(a, g(z)))$.

Back

Term subsumption

$$s \ll t \Leftrightarrow \sigma(s) = t$$

Vocabulary :
t is called an **instance** of s
s is said **more general** than t or
s **subsumes** t
σ is a **match** from s to t.
$\ll$ is a quasi-ordering on terms called **subsumption**.

$$f(x, y) \ll f(f(a, b), h(y))$$

Theorem : [Huet78]
Up to renaming, the subsumption ordering on terms is well-founded.

Notice that

$s \leq t \quad \not\Rightarrow \quad f(u,s) \leq f(u,t)$
since
$x \leq a$ but $f(x,x) \not\leq f(x,a)$

$s \leq t \quad \not\Rightarrow \quad \sigma(s) \leq \sigma(t)$
since
$x \leq a$ but $(x \mapsto b)x \not\leq (x \mapsto b)a$

Back

Formally

t rewrites to t' using the rule $l \longrightarrow r$ if

$$t_{|p} = \sigma(l) \quad \text{and} \quad t' = t[\sigma(r)]_p$$

This is denoted

$$t \longrightarrow_p^{l \longrightarrow r} t'$$

Rewrite relation

A term rewrite system R (a set of rewrite rules) determines a relation on terms denoted $\longrightarrow_R$:

$$u \to_R v$$

iff
there exist t, $l \to r \in R$, an occurrence ω in t, such that
$u = t[\sigma(l)]_\omega$
and
$v = t[\sigma(r)]_\omega$

$$t[\sigma(l)]_\omega \to_R t[\sigma(r)]_\omega$$

USUALLY, when defining the rewriting relation, one requires the all rewrite rules satisfy $Var(r) \subseteq Var(l)$.

Simple examples —

Consider the rewrite system R :

$$\begin{array}{lcl} x + x & \rightarrow & x \\ (a + x) + y & \rightarrow & y + x \end{array}$$

How many redexes are in $(a + a) + (a + a)$? — 4

Is $((a + a) + (a + a), a)$ in the transitive closure of $\rightarrow$? — yes

Is (a, a) in the transitive closure of $\rightarrow$? — no

Is there any infinite derivation starting from a finite tree using R ? — no

Back

Generalized rules

$$l \longrightarrow r \textbf{ where } p_1 := c_1 \ldots \textbf{ where } p_n := c_n$$

- $l, r, p_1, \ldots, p_n, c_1, \ldots, c_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$,
- $\mathcal{V}ar(p_i) \cap (\mathcal{V}ar(l) \cup \mathcal{V}ar(p_1) \cup \cdots \cup \mathcal{V}ar(p_{i-1})) = \emptyset$,
- $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l) \cup \mathcal{V}ar(p_1) \cup \cdots \cup \mathcal{V}ar(p_n)$
- $\mathcal{V}ar(c_i) \subseteq \mathcal{V}ar(l) \cup \mathcal{V}ar(p_1) \cup \cdots \cup \mathcal{V}ar(p_{i-1})$.

where *true* := c is equivalently written **if** c.
p_i is oftern reduced to a variable x.

Generalized rule application

$$l \to r \textbf{ where } p_1 := c_1 \ldots \textbf{ where } p_n := c_n$$

To apply this rewrite rule on t, the matching substitution σ from l to t (i.e. such that $l\sigma = t$) is successively composed with each match μ_i from p_i to $c_i\sigma\mu_1 \ldots \mu_{i-1}$, for all $i = 1, \ldots, n$.

$$move(S) \to C(x, y) \textbf{ where } < x, y >:= position(S) \textbf{ if } x = y$$

◂ Back to rewriting

◂ Back to NSPK

Abstract rewrite systems

- **Consider a set $\mathcal{T}$**
- **Consider a binary relation $\longrightarrow$ on $\mathcal{T}$ (one-step reduction)**
  - $a \longrightarrow b$: b is the reduct of a
- **Induced relations**
  - transitive closure : $\overset{+}{\longrightarrow}$
  - transitive reflexive closure : $\overset{*}{\longrightarrow}$
  - symetric closure : $\longleftrightarrow$

Normalization

Consider an ARS $(\mathcal{T}, \rightarrow)$

- An element $t \in \mathcal{T}$ is a **$\rightarrow$-normal form** if there exists no $t' \in \mathcal{T}$ such that $t \rightarrow t'$.

- The relation $\rightarrow$ is **terminating** (or **strongly normalizing**, or **noetherian**) if every reduction sequence is finite.
  .
  $a \rightarrow a$ is not terminating

- The relation $\rightarrow$ is **weakly normalizing** (or weakly terminating) if every element $t \in \mathcal{T}$ has a normal form.
  .
  $a \rightarrow a \quad a \rightarrow b$ is weakly terminating

- The relation $\rightarrow$ has the **unique normal form property** if for any $t, t' \in \mathcal{T}$, $t \overset{*}{\longleftrightarrow} t'$ and t, t' are normal forms imply $t = t'$.

Back

Noetherian induction : a fondamental tool

Let $(\mathcal{T}, >)$ be an ordered set s.t. $>$ is well-founded.

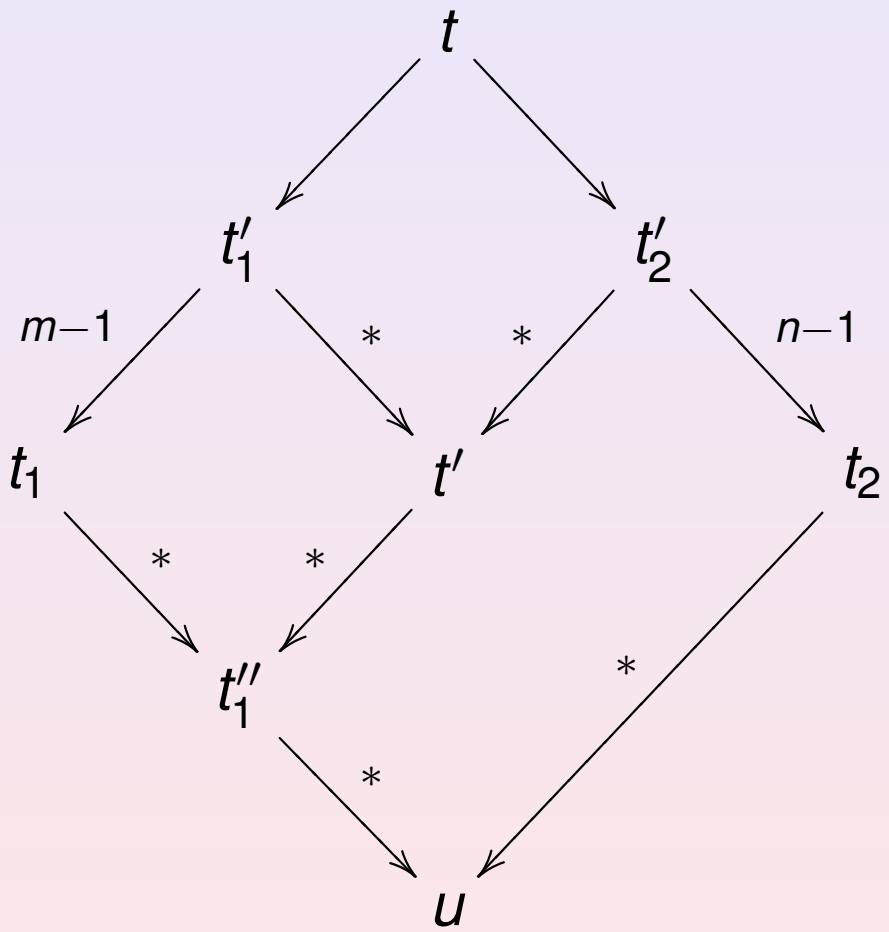
Let $\mathcal{P}$ be a proposition :

1. $\forall t \in \mathcal{T},\ [\forall t' \in \{t' \mid t > t'\},\ \mathcal{P}(t')] \Rightarrow \mathcal{P}(t)$
2. $\mathcal{P}(t)$ is provable for all minimal element t,

then $\forall t \in \mathcal{T}, \mathcal{P}(t)$.

Noetherian induction : a fondamental tool

Consider $(\mathcal{T}, \rightarrow)$



Back

Multiset extensions

Let $>$ an ordering on $\mathcal{T}$.

Its (strict) multiset extension $>^{mult}$ is defined by : $\mathcal{M} >^{mult} \mathcal{N}$ if :

- $\mathcal{M} \neq \mathcal{N}$, and
- $\mathcal{N}(t) > \mathcal{M}(t)$ implies $\exists t' \in \mathcal{T}$ such that $t' > t$ and $\mathcal{M}(t') > \mathcal{N}(t')$.

Its (strict) multiset extension denoted $>^{mult}$ is defined by :

$$\mathcal{M} = \{s_1, \ldots, s_m\} >^{mult} \mathcal{N} = \{t_1, \ldots, t_n\}$$

if there exist $i \in \{1, \ldots, m\}$ and $1 \leq j_1 < \ldots < j_k \leq n$ with $k \geq 0$, such that :

- $s_i > t_{j_1}, \ldots, s_i > t_{j_k}$ and,
- either $\mathcal{M} - \{s_i\} >^{mult} \mathcal{N} - \{t_{j_1}, \ldots, t_{j_k}\}$ or the multisets $\mathcal{M} - \{s_i\}$ and $\mathcal{N} - \{t_{j_1}, \ldots, t_{j_k}\}$ are equals.

Multiset extensions - Examples

if $>$ is well-founded on $\mathcal{T}$, then $>^{mult}$ is well-founded on $\mathcal{T}^n$.

$\{3,3,1,2\} >^{mult} \{3,1\}$
$\{3,3,1,2\} >^{mult} \{3,2,2,2,2\}$
$\{3,3,1,2\} >^{mult} \{3,0\} >^{mult} \{3\} >^{mult} \{\}$.

◂ Back to LPO

Back

Data structures

Data structures

- The agent

```
@ + @ + @  : ( AgentId SWC Nonce ) Agent;


@          : ( Agent ) listAgent;
@ || @     : ( listAgent  listAgent ) listAgent (AC);
```

- The messages

```
@-->@:@[@,@,@] : ( AgentId AgentId Key Nonce Nonce Address ) message


@          : ( message ) network;
@ & @      : ( network  network ) network (AC);
```

- The intruders

```
@ # @ # @ : ( AgentId listNonce network ) intruder;
```

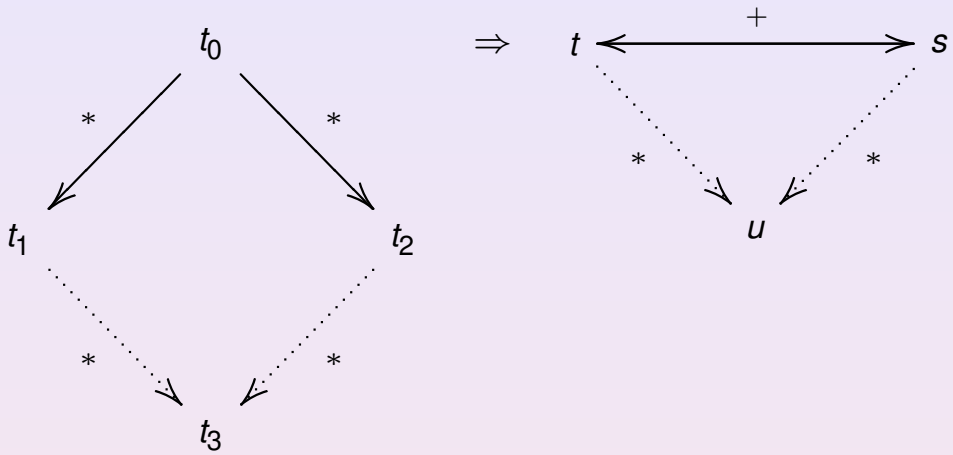
- The global state

```
@ <> @ <> @ <> @ : ( listAgent listAgent intruder network ) state;
```

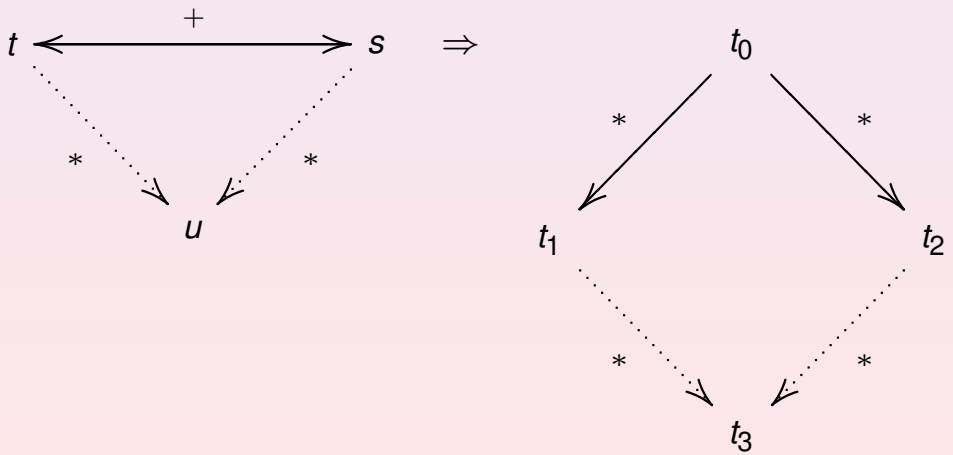
◂ Back to NSPK

Relationship between properties

1. $C \Rightarrow CR$ (by induction)



2. $CR \Rightarrow C$



◂ Back

Building reduction orderings using interpretations

Consider a homomorphism τ from ground terms to $(\mathcal{A}, >)$ with $>$ a well-founded ordering and let f_τ denote the image of $f \in \mathcal{F}$ using τ ; τ and $>$ are constrained to satisfy the monotonicity condition :

$$\forall a, b \in \mathcal{A}, \forall f \in \mathcal{F}, \quad a > b \text{ implies } f_\tau(\ldots, a, \ldots) > f_\tau(\ldots, b, \ldots).$$

Then the ordering $>_\tau$ defined by :

$$\forall s, t \in \mathcal{T}(\mathcal{F}), \; s >_\tau t \text{ if } \tau(s) > \tau(t),$$

is well-founded.

Building reduction orderings using interpretations

Then the ordering $>_\tau$ is extended by defining

$$\forall s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X}),\ s >_\tau t \text{ if } \nu(\tau(s)) > \nu(\tau(t))$$

for all assignment ν of values in $\mathcal{A}$ to variables of $\tau(s)$ and $\tau(t)$.
Because $>$ is assumed to be well-founded, a rewrite system is terminating if one can find $\mathcal{A}, \tau$ and $>$ as defined above.

Example

Is the reduction induced by $i(f(x,y)) \to f(f(i(x),y),y)$ terminating ?

$$\begin{array}{rcl} \tau(i(x)) & = & \tau(x)^2 \\ \tau(f(x,y)) & = & \tau(x) + \tau(y) \end{array} \qquad \begin{array}{rcl} \tau(x) & = & x \\ \tau(y) & = & y \end{array}$$

Monotonicity : $a > b$ implies $f_\tau(a) > f_\tau(b)$
(each function is increasing on natural numbers)

$$\begin{array}{rcl} \tau(i(f(x,y))) & = & (x+y)^2 = x^2 + y^2 + 2xy \\ \tau(f(f(i(x),y),y)) & = & x^2 + 2y \end{array}$$

For any assignment of positive natural numbers n and m to the variables x and y : $n^2 + m^2 + 2nm > n^2 + 2m$

Another example

Is the following system terminating ?

$$\begin{aligned}
\ominus \ominus x &\to x \\
\ominus(x \oplus y) &\to (\ominus x) \oplus (\ominus y) \\
\ominus(x \otimes y) &\to (\ominus x) \otimes (\ominus y) \\
x \otimes (y \oplus z) &\to (x \otimes y) \oplus (x \otimes z) \\
(x \oplus y) \otimes z &\to (x \otimes z) \oplus (y \otimes z)
\end{aligned}$$

Interpretation :

$$\begin{aligned}
\tau(\ominus x) &= 2^{\tau(x)} \\
\tau(x \oplus y) &= \tau(x) + \tau(y) + 1 \\
\tau(x \otimes y) &= \tau(x)\tau(y) \\
\tau(c) &= 3
\end{aligned}$$

Back

Lexicographic Path Ordering (LPO)

For a given precedence on $\mathcal{F}$,

$$s = f(s_1, .., s_n) >_{lpo} t = g(t_1, \ldots, t_m)$$

if at least one of the following condition is satisfied :

1. $f = g$ and $(s_1, \ldots, s_n) >_{lpo}^{lex} (t_1, \ldots, t_m)$ and $\forall j \in \{1, \ldots, m\}, s >_{lpo} t_j$
2. $f >_{\mathcal{F}} g$ and $\forall j \in \{1, \ldots, m\}, s >_{lpo} t_j$
3. $\exists i \in \{1, \ldots, n\}$ s.t either $s_i >_{lpo} t$, or $s_i = t$.

Theorem **LPO is a simplification ordering**
i.e. a reduction ordering that contains the subterm ordering.

Extension of LPO

The definition of the ordering can be extended to terms with variables by adding the following conditions :

1. two different variables are incomparable,
2. a function symbol and a variable are incomparable.

A typical LPO example

Termination of the Ackermann function :

$$\begin{aligned} ack(0, y) &\rightarrow succ(y) \\ ack(succ(x), 0) &\rightarrow ack(x, succ(0)) \\ ack(succ(x), succ(y)) &\rightarrow ack(x, ack(succ(x), y)). \end{aligned}$$

With $ack >_{\mathcal{F}} succ$, we can show that

$$\begin{aligned} ack(0, y) &>_{lpo} succ(y) \\ ack(succ(x), 0) &>_{lpo} ack(x, succ(0)) \\ ack(succ(x), succ(y)) &>_{lpo} ack(x, ack(succ(x), y)). \end{aligned}$$

Multiset extensions : ◂ MPO

Back

Critical pair

A non-variable term t' and a term t overlap if there exists a position ω in t such that $t_{|\omega}$ and t' are unifiable (with $t_{|\omega}$ not a variable).

Where does $(x + y) + z$ and $(x' + y') + z'$ overlap ?

Where does $0 + x \to x$ and $s(x) + y \to s(x + y)$ superpose ?

How about $H \vdash \overline{\overline{P}} \to H \vdash P$ and $H \wedge P \vdash P \to T$?

Critical Pairs

Superposition

$$\mathbf{l_1} \to \mathbf{r_1} \qquad \mathbf{l_2[u]} \to \mathbf{r_2}$$
$$\mathbf{l_2}[r_1]\sigma = \mathbf{r_2}\sigma$$

u is a non-variable sub-term of l_2
σ is the $mgu(u, l_1)$

Critical Pair Lemma

R is locally confluent iff all critical pair satisfies :

$$\mathbf{l}_2[r_1]\sigma \xrightarrow{*}_R \otimes \ {}_R \xleftarrow{*} r_2\sigma$$

Back

Orthogonal systems

A rewrite system that is both linear (the left-hand side of each rule is a linear term) and non-overlapping is called orthogonal.

Theorem **If a rewrite system is orthogonal, then it is confluent.**

Linearity is needed for non-terminating rewriting system :

$$\begin{aligned} d(x,x) &\to t \\ d(x,c(x)) &\to f \\ a &\to c(a) \end{aligned}$$

Back

Definitions

The rewriting relation RA is

- **Church-Rosser modulo A** if

$$=_{R \cup A} \subseteq \xrightarrow{*}_{RA} \circ =_A \circ \xleftarrow{*}_{RA} .$$

- **confluent modulo A** if

$$\xleftarrow{*}_{RA} \circ \xrightarrow{*}_{RA} \subseteq \xrightarrow{*}_{RA} \circ =_A \circ \xleftarrow{*}_{RA}$$

- **coherent modulo A** if

$$\xleftarrow{*}_{RA} \circ =_A \subseteq \xrightarrow{*}_{RA} \circ =_A \circ \xleftarrow{*}_{RA}$$

- **locally coherent with R modulo A** if

$$\xleftarrow{}_{RA} \circ \xrightarrow{}_{R} \subseteq \xrightarrow{*}_{RA} \circ =_A \circ \xleftarrow{*}_{RA}$$

- **locally coherent with A modulo A** if

$$\xleftarrow{}_{RA} \circ =_A \subseteq \xrightarrow{*}_{RA} \circ =_A \circ \xleftarrow{*}_{RA}$$

Good news

If R/A is terminating, the following properties are equivalent :

1. $\rightarrow_{RA}$ is Church-Rosser modulo A.
2. $\rightarrow_{RA}$ is confluent modulo A and $\rightarrow_{RA}$ is coherent modulo A.
3. $\rightarrow_{RA}$ is locally coherent with R modulo A and locally coherent with A modulo A.
4. $\forall t, t', t =_{R \cup A} t'$ iff $t \downarrow_{RA} =_A t' \downarrow_{RA}$.

Back

ρ-calculus and records

Record = structure composed of rewriting rules

$$\begin{array}{rcll} (T_i)^{i=1\ldots n} & \triangleq & T_1 \wr \ldots \wr T_n & (n \in \mathbb{N}) \\ [m_i = T_i]^{i \in I} & \triangleq & (m_i \twoheadrightarrow T_i)^{i \in I} & \\ [cx = 0, cy = 0] & \triangleq & (cx \twoheadrightarrow 0 \wr cy \twoheadrightarrow 0) & \end{array}$$

Record selection = the application of the record to the label

$$\begin{array}{rcl} (cx \twoheadrightarrow 0 \wr cy \twoheadrightarrow 0)\, cx & \mapsto_{\delta} & (cx \twoheadrightarrow 0)\, cx \wr (cy \twoheadrightarrow 0)\, cx \\ & \mapsto_{\rho\circ\delta} & 0 \wr (cy \twoheadrightarrow 0)\, cx \\ & \mapsto_{stk} & 0 \wr stk \\ & \mapsto_{stk} & 0 \end{array}$$

Back

The object Daemon : methods as first-class entities

$$Daemon \triangleq set \to S \to X \to (X \wr set \to S' \to Y \to (Y \wr S'))$$
$$Daemon.set(x \to S \to 3)$$

$$\begin{aligned}
&\triangleq && Daemon\ set\ Daemon\ (x \to S \to 3)\\
&\mapsto\!\!\!\!\!\rightarrow_{\rho\partial\delta} && (S \to X \to (X \wr set \to S' \to Y \to (Y \wr S')))\ Daemon\ (x \to S \to 3)\\
&\mapsto\!\!\!\!\!\rightarrow_{\rho\partial\delta} && (X \to (X \wr set \to S' \to Y \to (Y \wr S')))\ (x \to S \to 3)\\
&\mapsto\!\!\!\!\!\rightarrow_{\rho\partial\delta} && \underbrace{x \to S \to 3 \wr set \to S' \to Y \to (Y \wr S')}_{obj}
\end{aligned}$$

$$obj.set(y \to S \to 4) \mapsto\!\!\!\!\!\rightarrow_{\rho\partial\delta}$$
$$(y \to S \to 4 \wr x \to S \to 3 \wr set \to S' \to Y \to (Y \wr S'))$$

Back

Inheritance in the ρ-calculus (Abadi & Cardelli encoding of classes-as-objects)

$$
\begin{aligned}
\mathit{PClass} \triangleq\ & \mathit{new} \to S \to (\mathit{val} \to S' \to (S.\mathit{preval})\ S' \wr \\
& \qquad \mathit{get} \to S' \to (S.\mathit{preget})\ S' \wr \\
& \qquad \mathit{set} \to S' \to (S.\mathit{preset})\ S') \wr \\
& \mathit{preval} \to S \to S' \to v(1,1) \wr \\
& \mathit{preget} \to S \to S' \to S'.\mathit{val} \wr \\
& \mathit{preset} \to S \to S' \to v(X,Y) \to (S'.\mathit{val} := S'' \to v(X,Y))
\end{aligned}
$$

Then :

$$\mathit{PClass.new} \mapsto\!\!\!\!\to_{\rho\delta} \mathit{Point}$$

◂ Back

Failures

$$\Big(f(X,Y) \to X \wr f(X,c) \to c\Big)\ f(a,b)$$

$\hookrightarrow$ $(f(X,Y) \to X)\ f(a,b) \wr (f(X,c) \to c)\ f(a,b)$

$\hookrightarrow$ $a \wr (f(X,c) \to c)\ f(a,b)$

Failure semantics : the symbol *stk*

$$\frac{\forall \theta_1, \theta_2,\ \forall B',\ B\theta_1 \mapsto\!\!\!\!\rightarrow_{\rho\delta} B' \ \Rightarrow\ P\theta_2 \not\equiv B'}{(P \rightarrowtriangle A)\ B \mapsto_{stk} stk}$$

Failure definition

$$
\begin{array}{lcll}
stk & \not\sqsubseteq & g\,\overline{B} & \text{if } g \not\equiv stk \\
stk & \not\sqsubseteq & Q \rightarrowtriangle B & \\
f\,P_1 \ldots P_m & \not\sqsubseteq & g\,B_1 \ldots B_n & \text{if } f \not\equiv g \vee n \neq m \vee \exists i,\ P_i \not\sqsubseteq B_i \\
f\,\overline{P} & \not\sqsubseteq & stk & \\
f\,\overline{P} & \not\sqsubseteq & Q \rightarrowtriangle B & \\
f\,\overline{P} & \not\sqsubseteq & (Q \rightarrowtriangle A)\,B & \text{if } Q \not\sqsubseteq B \vee f\,\overline{P} \not\sqsubseteq A
\end{array}
$$

Properties

Correction of $\not\sqsubseteq$:
For all P and A, if $P \not\sqsubseteq A$ then $\forall \theta_1, \theta_2,\ \forall A',\ A\theta_1 \mapsto^{stk}_{\rho\delta} A' \ \Rightarrow\ P\theta_2 \not\equiv A'$.

Stability of $\not\sqsubseteq$
The relation $P \not\sqsubseteq A$ is stable by substitution and reduction of A.

Handling matching failures

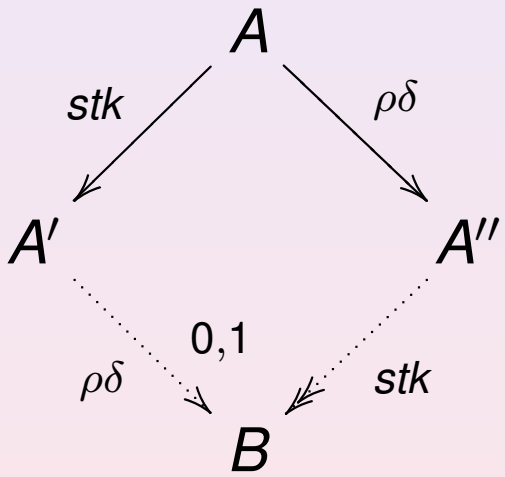
The relation $\rightarrow_{stk}$ treats matching failures uniformly :

$$
\begin{aligned}
(P \rightarrowtriangle B)\, A &\rightarrow_{stk} stk \qquad \text{if } P \not\sqsupseteq A \\
stk \wr A &\rightarrow_{stk} A \\
A \wr stk &\rightarrow_{stk} A \\
stk\ A &\rightarrow_{stk} stk
\end{aligned}
$$

Properties

Confluence and termination of $\mapsto_{stk}$:
The relation $\mapsto_{stk}$ is confluent and terminating.

$\mapsto_{\rho\delta}$ and $\mapsto_{stk}$ commute



Confluence of $\mapsto_{\rho\delta}^{stk}$
The relation $\mapsto_{\rho\delta}^{stk}$ is confluent.

New reduction : $\mapsto_{\rho\delta}^{stk} = \mapsto_{stk} \cup \mapsto_{\rho\delta}$

$$\Big(f(X,Y) \to X \wr \qquad f(X,c) \to c\Big) \qquad f(a,b)$$

- $(f(X,Y) \to X)\ f(a,b) \wr (f(X,c) \to c)\ f(a,b)$
- $a \wr stk$
- a

No cheating

$$plus \triangleq (rec\ z) \to \left(\wr \quad \begin{array}{l} (add\ 0\ y) \to y \\ (add\ (S\ x)\ y) \to S\ (z\ (rec\ z)\ (add\ x\ y)) \end{array} \right)$$

Reduction

$$
\begin{aligned}
& plus\ (rec\ plus)\ (add\ \overline{n}\ \overline{m}) \\
\mapsto_{\rho\delta} \quad & ((add\ 0\ y) \to y)\ (add\ \overline{n}\ \overline{m}) \\
& \wr\ ((add\ (S\ x)\ y) \to S\ (plus\ (rec\ plus)\ (add\ x\ y)))\ (add\ \overline{n}\ \overline{m}) \\
\mapsto_{\rho} \quad & ((add\ 0\ y) \to y)\ (add\ \overline{n}\ \overline{m}) \\
& \wr\ S\ (plus\ (rec\ plus)\ (add\ \overline{n-1}\ \overline{m})) \\
\mapsto_{stk} \quad & S\ (plus\ (rec\ plus)\ (add\ \overline{n-1}\ \overline{m})) \\
\mapsto_{\rho\delta} \quad & S\ (((add\ 0\ y) \to y)\ (add\ \overline{n-1}\ \overline{m}) \\
& \wr\ ((add\ (S\ x)\ y) \to S\ (plus\ (rec\ plus)\ (add\ x\ y)))\ (add\ \overline{n-1}\ \overline{m})) \\
& \vdots
\end{aligned}
$$

Reduction

$$
\begin{array}{ll}
 & \vdots \\
\mapsto_{\rho\delta}^{stk} & S\,(\ldots(\\
 & \quad S\,(((add\;0\;y) \rightarrow\!\!\triangleright\; y)\;(add\;0\;\overline{m})\\
 & \quad \wr\;((add\;(S\;x)\;y) \rightarrow\!\!\triangleright\; S\,(plus\;(rec\;plus)\;(add\;x\;y)))\;(add\;0\;\overline{m}))\\
 &)\ldots) \\
\mapsto_{\rho\delta} & S\,(\ldots(\\
 & \quad S\,(\overline{m}\\
 & \quad \wr\;((add\;(S\;x)\;y) \rightarrow\!\!\triangleright\; S\,(plus\;(rec\;plus)\;(add\;x\;y)))\;(add\;0\;\overline{m}))\\
 &)\ldots) \\
\mapsto_{stk} & S\,(\ldots(S\,\overline{m})\ldots) \\
\equiv & \overline{m+n}
\end{array}
$$

First

$$\mathit{first}(A_1, A_2, \ldots, A_n) \triangleq$$

$$x \rightarrowtriangle ((\mathit{stk} \rightarrowtriangle A_n\, x \wr y \rightarrowtriangle y)\, (\ldots (\mathit{stk} \rightarrowtriangle A_2\, x \wr y \rightarrowtriangle y)\, (A_1\, x)))$$

For any term B, the term $\mathit{first}(A_1, \ldots, A_n)\, B$ evaluates like $A_i\, B$ if

$$\begin{cases} \forall j < i, & A_j\, B & \overset{\mathit{stk}}{\underset{\rho\delta}{\longmapsto\!\!\!\!\rightarrow}} & \mathit{stk} \\ & A_i\, B & \overset{\mathit{stk}}{\underset{\rho\delta}{\longmapsto\!\!\!\!\rightarrow}} & f\ \overline{B} \end{cases}$$

First

If $A_i\ B \mapsto^{stk}_{\rho\delta} stk$

$$(stk \to A_{i+1} B \wr y \to y)\ (A_i\ B) \mapsto^{stk}_{\rho\delta} (stk \to A_{i+1} B)\ stk \wr (y \to y)\ stk \mapsto^{stk}_{\rho\delta} A_{i+1} B \wr stk \mapsto_{stk} A_{i+1} B$$

If $A_i\ B \mapsto^{stk}_{\rho\delta} f\ \overline{B}$ then $stk \not\sqsupseteq f\ \overline{B}$ and thus

$$(stk \to A_{i+1} B \wr y.y)\ (A_i\ B) \mapsto^{stk}_{\rho\delta} (stk \to A_{i+1} B)\ (f\overline{B}) \wr (y \to y)\ (f\ \overline{B}) \mapsto^{stk}_{\rho\delta} stk \wr f\ \overline{B} \mapsto_{stk} f\ \overline{B}$$

$first(A_1, \ldots, A_n, y \to y)\ B$ returns $A_i\ B$ or B if all $A_i\ B$ fail.

Encoding

$$(\!|\mathcal{R}|\!) \triangleq (rec\ z) \rightarrow first \left(\begin{array}{l} l_1 \rightarrow z\ (rec\ z)\ r_1, \\ \ldots \\ l_n \rightarrow z\ (rec\ z)\ r_n, \\ (a_1\ \overline{x}) \rightarrow z\ (Rec\ z)\ (a_1\ \overline{z\ (rec\ z)\ x}), \\ \ldots \\ (a_m\ \overline{x}) \rightarrow z\ (Rec\ z)\ (a_m\ \overline{z\ (rec\ z)\ x}) \end{array} \right)$$

$$\wr\ (Rec\ z) \rightarrow first \left(\begin{array}{l} l_1 \rightarrow z\ (rec\ z)\ r_1, \\ \ldots \\ l_n \rightarrow z\ (rec\ z)\ r_n, \\ y \rightarrow y \end{array} \right)$$

Innermost strategy

$$(\!|\mathcal{R}|\!) \triangleq (rec\ z) \to first \left(\begin{array}{l} (a_1\ \overline{x}) \to z\ (Rec\ z)\ (a_1\ \overline{z\ (rec\ z)\ x}), \\ \ldots \\ (a_m\ \overline{x}) \to z\ (Rec\ z)\ (a_m\ \overline{z\ (rec\ z)\ x}), \\ l_1 \to z\ (rec\ z)\ r_1, \\ \ldots \\ l_n \to z\ (rec\ z)\ r_n \end{array} \right) \wr (Rec\ z) \to first \left(\begin{array}{l} l_1 \to z\ (rec\ z)\ r_1, \\ \ldots \\ l_n \to z\ (rec\ z)\ r_n, \\ y \to y \end{array} \right)$$

Back

Correctness and completeness of the encoding

1. For all TRS $\mathcal{R}$, for all algebraic terms t and t',

$$(\!|\mathcal{R}|\!)\ (\mathit{rec}\ (\!|\mathcal{R}|\!))\ (\!|t|\!) \mapsto^{stk}_{\rho\delta} (\!|t'|\!) \quad \Rightarrow \quad t \underset{\mathcal{R}}{\longmapsto\!\!\!\!\rightarrow} t' \text{ and } t' \text{ in } \mathcal{R}\text{-normal form.}$$

2. For all convergent TRS $\mathcal{R}$, for all algebraic terms t and t' such that t' is in $\mathcal{R}$-normal form,

$$t \underset{\mathcal{R}}{\longmapsto\!\!\!\!\rightarrow} t' \quad \Rightarrow \quad (\!|\mathcal{R}|\!)\ (\mathit{rec}\ (\!|\mathcal{R}|\!))\ (\!|t|\!) \mapsto^{stk}_{\rho\delta} (\!|t'|\!)$$

Counterexamples

1. Non confluent $\mathcal{R} = \{random \to 0, random \to 1\}$

$$(\!|\mathcal{R}|\!)\ (rec\ (\!|\mathcal{R}|\!))\ random \mapsto^{stk}_{\rho\delta} 0$$

but no corresponding reduction in ρ-calculus for

$$random \to_{\mathcal{R}} 1$$

2. Non terminating $\mathcal{R}$

$$\mathcal{R} = \left\{ \begin{array}{rcl} g(x) & \to & g(Sx) \\ g(x) & \to & 0 \\ f(0) & \to & 0 \end{array} \right.$$

$f(g(0)) \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} f(g(S^n 0)) \to_{\mathcal{R}} f(0) \to_{\mathcal{R}} 0$
but

$$(\!|\mathcal{R}|\!)\ (rec\ (\!|\mathcal{R}|\!))\ (f(g\,0)) \mapsto^{stk}_{\rho\delta} \ldots (\!|\mathcal{R}|\!)\ (rec\ (\!|\mathcal{R}|\!))\ f((\!|\mathcal{R}|\!)\ (rec\ (\!|\mathcal{R}|\!))\ g(S^n 0)) \mapsto^{stk}_{\rho\delta} \ldots$$

Back