Rewriting, Explicit Substitutions and Normalisation

XXXVI Escola de Verão do MAT
Universidade de Brasilia

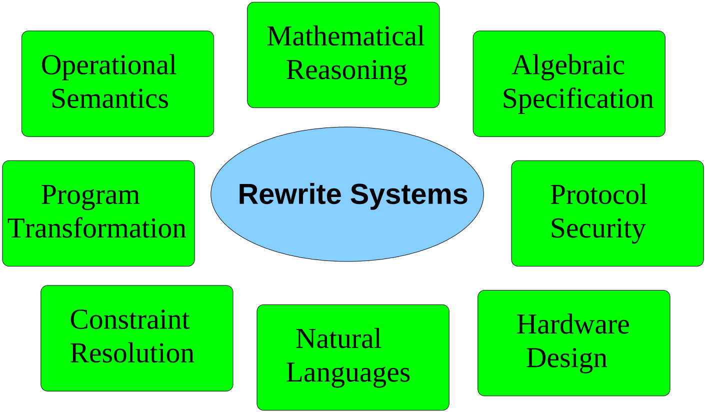
Part 1/3

Eduardo Bonelli

LIFIA (Fac. de Informática, UNLP, Arg.) and CONICET
eduardo@lifia.info.unlp.edu.ar

February, 2006

Applications



Example - Arithmetic

Natural numbers as Peano numerals: 0, s(0), s(s(0)), etc.

Rewrite system

$$\begin{array}{lcl} a(x,0) & \rightarrow & x \\ a(x,s(y)) & \rightarrow & s(a(x,y)) \\ m(x,0) & \rightarrow & 0 \\ m(x,s(y)) & \rightarrow & a(m(x,y),x) \end{array}$$

Reduction sequence

$$\begin{array}{lcl} \text{“}2+2\text{''} = \underline{a(s(s(0)),s(s(0)))} & \rightarrow & s(\underline{a(s(s(0)),s(0))}) \\ & \rightarrow & s(s(\underline{a(s(s(0)),0)})) \\ & \rightarrow & s(s(s(s(0)))) \end{array}$$

Example - Negation Normal Form

Rewrite system

$$
\begin{array}{lcl}
x \implies y & \to & \neg x \vee y \\
\neg(x \wedge y) & \to & \neg x \vee \neg y \\
\neg(x \vee y) & \to & \neg x \wedge \neg y \\
\neg\neg x & \to & x
\end{array}
$$

Reduction sequence

$$
\begin{array}{lcl}
\neg(\underline{\neg(x \implies y)} \vee z) & \to & \neg(\underline{\neg(\neg x \vee y)} \vee z) \\
& \to & \neg((\underline{\neg\neg x} \wedge \neg y) \vee z) \\
& \to & \underline{\neg((x \wedge \neg y) \vee z)} \\
& \to & \underline{\neg(x \wedge \neg y)} \wedge \neg z \\
& \to & (\neg x \vee \underline{\neg\neg y}) \wedge \neg z \\
& \to & (\neg x \vee y) \wedge \neg z
\end{array}
$$

Example - Combinatory Logic

Rewrite system

$$
\begin{array}{lcl}
((((S \cdot x) \cdot y) \cdot z) & \rightarrow & ((x \cdot z) \cdot (y \cdot z)) \\
((K \cdot x) \cdot y) & \rightarrow & x \\
(I \cdot x) & \rightarrow & x
\end{array}
$$

Reduction sequence

$$
\begin{array}{lcl}
\underline{(((S \cdot I) \cdot I) \cdot x)} & \rightarrow & (\underline{(I \cdot x)} \cdot (I \cdot x)) \\
 & \rightarrow & (x \cdot \underline{(I \cdot x)}) \\
 & \rightarrow & {\color{red}(x \cdot x)}
\end{array}
$$

Example - Functional Programming

Rewrite system

$$\begin{array}{lcl} map(\lambda x.M, nil) & \to & nil \\ map(\lambda x.M, cons(X, T)) & \to & cons(M\{x/X\}, map(\lambda x.M, T)) \end{array}$$

Reduction sequence ($[n]$ abbreviates $cons(n, nil)$)

$$\begin{array}{rl} & \underline{map(\lambda x.cons(x, nil), cons(1, (cons(2, nil))))} \\ \to & cons([1], \underline{map(\lambda x.cons(x, nil), cons(2, nil))}) \\ \to & cons([1], cons([2], \underline{map(\lambda x.cons(x, nil), nil)})) \\ \to & cons([1], cons([2], nil)) \end{array}$$

Example - Object Oriented Programming

Terms

$$
\begin{array}{lll}
M & ::= & x & \text{variable} \\
& | & [l_i = \varsigma(x_i)N_i \ ^{i \in \{1..n\}}] & \text{object} \\
& | & M.l & \text{method invocation} \\
& | & M.l \leftharpoonup \varsigma(x)N & \text{method update}
\end{array}
$$

Rewrite system ($o = [l_i = \varsigma(x_i)N_i \ ^{i \in \{1..n\}}]$ and $j \in 1..n$)

$$
\begin{array}{lcl}
o.l_j & \to & N_j\{x_j/o\} \\
o.l_j \leftharpoonup \varsigma(x)N & \to & [l = \varsigma(x)N, l_i = \varsigma(x_i)N_i \ ^{i \in \{1..n\} \setminus \{j\}}]
\end{array}
$$

Reduction sequence

$$
\begin{array}{lcl}
\underline{[l = \varsigma(y)(y.l \leftharpoonup \varsigma(x)x)].l} & \to & \underline{[l = \varsigma(y)(y.l \leftharpoonup \varsigma(x)x)].l \leftharpoonup \varsigma(x)x} \\
& \to & [l = \varsigma(x)x]
\end{array}
$$

Bibliography - Rewriting

1. TERM REWRITING SYSTEMS, TERESE, Cambridge Tracts in Theoretical Computer Science, Vol. 55, CUP, 2003.
2. TERM REWRITING AND ALL THAT, Franz Baader, Tobias Nipkow, CUP, 1998.
3. ADVANCED TOPICS IN REWRITING, Enno Ohlebusch, Springer, 2002.

More information at the rewriting home page

`http://rewriting.loria.fr`

Bibliography - Lambda Calculus

THE LAMBDA CALCULUS: ITS SYNTAX AND SEMANTICS, Henk Barendregt, North Holland, 1984.

An Aside

Definitions

Examples

Highlights or comments

Thm/Lemma/Proof

Statement of Thm/Lemma and proofs

Structure of Today's Talk

1. Abstract Reduction Systems
2. First-Order Rewriting
3. Lambda Calculus

Introduction

An Abstract Reduction System (ARS) is a structure $\langle A, \{\rightarrow_\alpha \mid \alpha \in I\}\rangle$ where

- A is a set
- $\{\rightarrow_\alpha \mid \alpha \in I\}$ is a family of binary relations on A indexed by I

- The relations $\rightarrow_\alpha$ are called reduction relations
- In the case of just one reduction relation we write $\rightarrow$

Examples

1. $$\begin{tikzcd} \bullet & \bullet \arrow[l, "\alpha"'] \arrow[r, bend right, "\alpha"'] & \bullet \arrow[l, bend right, "\beta"'] \end{tikzcd}$$

2. $A = \{\bullet, \circ\}^+$ and $u \to v$ for $u, v \in A$ if
   - $u = u_1 l u_2$ and $v = u_1 r u_2$
   - $\langle l, r \rangle$ is one of

$$\begin{array}{l} \langle \bullet \circ \,,\, \circ \circ \circ \bullet \rangle \\ \langle \circ \bullet \,,\, \bullet \rangle \\ \langle \bullet \bullet \,,\, \circ \circ \circ \circ \rangle \\ \langle \circ \circ \,,\, \circ \rangle \end{array}$$

Reduction

- A reduction sequence or derivation w.r.t. $\to_\alpha$ is a finite or infinite sequence $a_0 \to_\alpha a_1 \to_\alpha a_2 \to_\alpha \ldots$
- A reduction step is an occurrence of $\to_\alpha$ in a reduction sequence

Recall from above

$\langle \bullet \circ \, , \, \circ \circ \circ \bullet \rangle$

$\langle \circ \bullet \, , \, \bullet \rangle$

$\langle \bullet \bullet \, , \, \circ \circ \circ \circ \rangle$

$\langle \circ \circ \, , \, \circ \rangle$

$$\bullet \circ \bullet \to \circ \circ \circ \bullet \bullet \to \circ \circ \circ \circ \circ \circ \circ \to \to \to \to \to \to \circ$$

Notation

$\rightarrow_\alpha^=$ reflexive closure of $\rightarrow_\alpha$
$\rightarrow_\alpha^+$ transitive closure of $\rightarrow_\alpha$
$\twoheadrightarrow_\alpha$ reflexive, transitive closure of $\rightarrow_\alpha$

Note: $a \twoheadrightarrow_\alpha b$ iff there is a finite reduction sequence

$$a = a_0 \rightarrow_\alpha a_1 \rightarrow_\alpha \ldots \rightarrow_\alpha a_n = b$$

$\bullet \circ \bullet \twoheadrightarrow \circ$ since

$$\bullet \circ \bullet \rightarrow \circ \circ \circ \bullet \bullet \rightarrow \circ \circ \circ \circ \circ \circ \circ \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \circ$$

Confluence

Diamond Property (DP)

$$
\begin{tikzcd}
 & a \arrow[dl] \arrow[dr] & \\
b \arrow[dr, dotted] & & c \arrow[dl, dotted] \\
 & d &
\end{tikzcd}
$$

$\forall\ a, b, c$ s.t. $a \longrightarrow b$ and $a \longrightarrow c$,
$\exists d$ s.t. $b \longrightarrow d$ and $c \longrightarrow d$

Weak Church Rosser (WCR)

$$
\begin{tikzcd}
 & a \arrow[dl] \arrow[dr] & \\
b \arrow[dr, dotted, two heads] & & c \arrow[dl, dotted, two heads] \\
 & d &
\end{tikzcd}
$$

$\forall\ a, b, c$ s.t. $a \longrightarrow b$ and $a \longrightarrow c$,
$\exists d$ s.t. $b \twoheadrightarrow d$ and $c \twoheadrightarrow d$

Confluence

Confluence or Church-Rosser (CR)

$$\begin{tikzcd} & a \arrow[dl, two heads] \arrow[dr, two heads] & \\ b \arrow[dr, dotted, two heads] & & c \arrow[dl, dotted, two heads] \\ & d & \end{tikzcd}$$

$\forall\ a, b, c$ s.t. $a \twoheadrightarrow b$ and $a \twoheadrightarrow c$, $\exists d$ s.t. $b \twoheadrightarrow d$ and $c \twoheadrightarrow d$

Normalization

Consider an ARS $(A, \rightarrow)$.

- $a \in A$ is a normal form if there exists no b s.t. $a \rightarrow b$
- $a \in A$ is weakly normalizing if $a \twoheadrightarrow b$ for b a normal form; $\rightarrow$ is weakly normalizing (WN) if every $a \in A$ is weakly normalizing
- $a \in A$ is strongly normalizing if every reduction sequence starting from a is finite; $\rightarrow$ is strongly normalizing (SN) if every $a \in A$ is strongly normalizing

WN, $\neg$SN

$$
\begin{tikzcd}
\bullet & \bullet \arrow[l] \arrow[r, bend right] & \bullet \arrow[l, bend right]
\end{tikzcd}
$$

Interrelation between Properties

CR $\Longrightarrow$ WCR (trivial)

Lemma

WCR $\not\Rightarrow$ CR

Proof (counterexample - Hindley)

$$\bullet \longleftarrow \bullet \leftrightarrows \bullet \longrightarrow \bullet$$

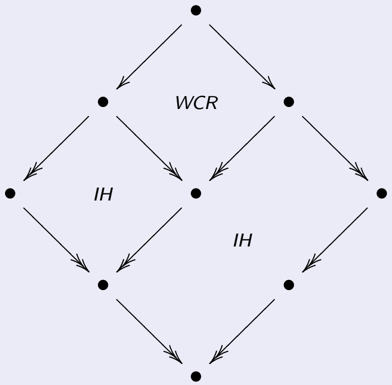
Interrelation between Properties

Thm (Newman's Lemma)

WCR and SN $\Longrightarrow$ CR

Proof [Huet1980]

By well-founded induction



Terms

Σ set of function symbols equipped with an arity n ($n \in \mathbb{N}$)
$\mathcal{X}$ set of variables
$\mathcal{T}(\Sigma)$ set of Σ-terms over $\mathcal{X}$

$$\frac{x \in \mathcal{X}}{x \in \mathcal{T}(\Sigma)} \qquad \frac{f \in \Sigma \text{ of arity } n \quad M_1, \ldots, M_n \in \mathcal{T}(\Sigma)}{f(M_1, \ldots, M_n) \in \mathcal{T}(\Sigma)}$$

- $\mathrm{Var}(M)$ denotes the variables in M
- M is closed if $\mathrm{Var}(M) = \emptyset$

Terms - Example

Signature

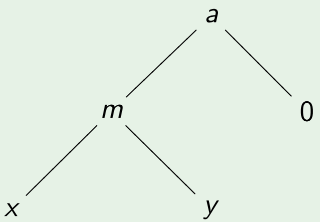
$$\begin{array}{llll} 0 & \text{(arity 0)} & s & \text{(arity 1)} \\ a & \text{(arity 2)} & m & \text{(arity 2)} \end{array}$$

Terms

$$s(0) \quad a(s(0), 0) \quad a(m(x, y), 0)$$

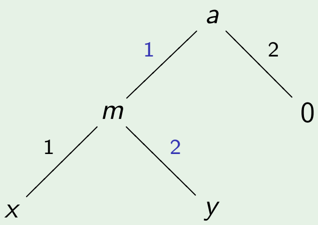
Terms as Trees

The term tree of $a(m(x, y), 0)$



Positions

- $\mathbb{N}^\star$ set of positions, where a position is a sequence of natural numbers $i_1\, i_2\, \ldots\, i_n$ (Note: we use ϵ for the empty sequence)
- Example: ϵ, 13, 249 (Note: We only use sequences of single digit numbers to avoid ambiguities)
- pos(M): Positions of the term tree of M



$$M = a(m(x,y),0)$$
$$\text{pos}(M) = \{\epsilon, 1, 2, 11, 12\}$$

Positions - Concatenation

Concatenation of positions

$$\begin{aligned} \epsilon \cdot q &= q \\ (i\,p) \cdot q &= i\,(p \cdot q) \end{aligned}$$

Prefix preorder

$p \preceq q$ (“p is a prefix of q”) iff $\exists r \in \mathbb{N}^\star \; p \cdot r = q$

$\epsilon \preceq p$, for all p
$1 \preceq 122$
$21 \preceq 213$
$21 \parallel 22$ (disjoint positions)

Subterms at a position

$M|_p$: Subterm of M at position $p \in \text{pos}(M)$

$$\frac{}{M|_\epsilon = M} \qquad \frac{M_i|_q = N \quad i \in \{1..n\}}{f(M_1, \ldots, M_n)|_{i\,q} = N}$$

$a(m(x,y),0)|_1 = m(x,y) \quad a(m(x,y),0)|_{1\,2} = y$

Substitution

A substitution is a map $\sigma : \mathcal{T}(\Sigma) \to \mathcal{T}(\Sigma)$ which satisfies

$$\sigma(f(M_1, \ldots, M_n)) = f(\sigma(M_1), \ldots, \sigma(M_n))$$

- We usually write M^σ instead of $\sigma(M)$
- $\sigma = \{x_1/M_1, \ldots, x_n/M_n\}$ determines a unique substitution (the expected one)

If $M = f(x, g(y))$ and $\sigma = \{x/g(a), y/f(x,x)\}$, then

$$M^\sigma = f(g(a), g(f(x,x)))$$

Unification

Terms M, N are said to be unifiable iff there exists a substitution σ (unifier) s.t. $M^\sigma = N^\sigma$

1. x is always unifiable with any M (provided that $x \notin \mathrm{Var}(M)$)
2. $f(x, g(x, a))$ is unifiable with $f(f(a), y)$ with unifier $\sigma = \{x/f(a), y/g(f(a), a)\}$
3. $f(x, g(x, a))$ and $f(f(a), g(b, a))$ are not unifiable

Preorder on Substitutions

Composition of substitutions σ, τ, written $\sigma \circ \tau$,

$$M^{\sigma \circ \tau} = (M^{\tau})^{\sigma}$$

Subsumption (σ is more general than τ)

$$\sigma \leq \tau \text{ iff } \exists \upsilon \text{ s.t. } \upsilon \circ \sigma = \tau$$

Note: $\leq$ is a preorder on substitutions (upto renaming)

Most General Unifier

Thm

If M, N are unifiable, then there exists a most general unifier (MGU) of M, N. Furthermore, this MGU is unique upto renaming.

Unification Algorithm (Martelli-Montanari)

E finite set of matching equations

$$
\begin{array}{lcl}
\{f(M_1,\ldots,M_n) \doteq f(N_1,\ldots,N_n)\} \cup E & \Longrightarrow & \{M_1 \doteq N_1,\ldots,M_n \doteq N_n\} \cup E \\
\{f(M_1,\ldots,M_n) \doteq g(N_1,\ldots,N_m)\} \cup E & \Longrightarrow & \textbf{fail} \\
\{x \doteq x\} \cup E & \Longrightarrow & E \\
\{f(M_1,\ldots,M_n) \doteq x\} \cup E & \Longrightarrow & \{x \doteq f(M_1,\ldots,M_n)\} \cup E \\
\{x \doteq f(M_1,\ldots,M_n)\} \cup E & \Longrightarrow & \textbf{fail} \\
 & & \text{if } x \in \mathsf{Var}(M_1,\ldots,M_n) \\
\{x \doteq M\} \cup E & \Longrightarrow & \{x \doteq M\} \cup E^{\{x/M\}} \\
 & & \text{if } x \notin \mathsf{Var}(M) \wedge x \in \mathsf{Var}(E)
\end{array}
$$

- To compute MGU of M and N, begin with $\{M \doteq N\}$ and apply rules repeatedly
- This process is CR and SN

Example

$$
\begin{array}{lcl}
a(x, 0) & \rightarrow & x \\
a(x, s(y)) & \rightarrow & s(a(x, y)) \\
m(x, 0) & \rightarrow & 0 \\
m(x, s(y)) & \rightarrow & a(m(x, y), x)
\end{array}
$$

Reduction Rule

A reduction rule for a signature Σ is a pair $\langle l, r\rangle$ of terms in $\mathcal{T}(\Sigma)$ such that

1. the left-hand side l is not a variable
2. every variable occurring in the right-hand side r occurs in l as well

- We often write $l \longrightarrow r$
- We sometimes give rules a name and write $\rho : l \longrightarrow r$
- We say ρ is left-linear if l contains at most one occurrence of any variable

Context

Context: Term over $\Sigma \cup \{\Box\}$. Special symbol $\Box$ denotes a hole.
If C is a context containing exactly n holes, then $C[M_1, \ldots, M_n]$ denotes the term resulting from replacing the holes of C from left to right with $M_1, \ldots, M_n$

- Unless stated, we restrict to contexts with exactly one occurence of $\Box$
- The p in $C[M]_p$ indicates $C\,|_p = \Box$

1. $a(m(s(\Box), x), 0)$
2. $a(0, \Box)$
3. $\Box$

Redex

A ρ-redex is an instance l^σ of the left-hand side of rule $\rho : l \longrightarrow r$ in a term M (source)

We use letters r, s for redexes

A redex is determined by

1. Pair of terms (source,target)
2. Rule name
3. Position
4. Substitution

In some cases, not all items are necessary

Redexes that have the same source are called coinitial

Redex - Example

$$\rho : f(x) \rightarrow x$$

Consider the term $f(f(y))$; it has two ρ-redexes

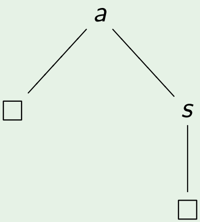
Source	$f(f(y))$	$f(f(y))$
Target	$f(y)$	$f(y)$
Rule	ρ	ρ
Position	ϵ	1
Subst	$\{x/f(y)\}$	$\{x/y\}$

Redex Patterns

The pattern of a rule $\rho : l \rightarrow r$ is l^{ϵ} where $x^{\epsilon} = \square$ for all variables x. The pattern of a ρ-redex is the pattern of ρ.
Let P be the pattern of a ρ-redex s. Then

1. $s = l^{\sigma} = P[x_1^{\sigma}, \ldots, x_n^{\sigma}]$ (note multiple holes) and
2. $x_1^{\sigma}, \ldots, x_n^{\sigma}$ are the arguments of s

Pattern of
$a(x, s(y)) \rightarrow s(a(x, y))$



Nested, Disjoint, Overlapping

Two coinitial redexes s and r are said to be

1. Disjoint: if their positions are disjoint
2. Nested (say s nests r): if r occurs in an argument of s
3. Overlapping: if their patterns share at least one symbol occurrence

Consider the TRS

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

overlapping	**nested**
f(g(g(a)))	f(g(g(a)))

We say f(g(g(a))) nests g(a)

Reduction Step

A reduction step according to the rule $\rho : l \to r$ consists of contracting a redex within an arbitrary context

$$C[l^\sigma] \to_\rho C[r^\sigma]$$

- Occasionally we write $C[l^\sigma] \to_s C[r^\sigma]$ (or even s) for this reduction step, where s is the ρ-redex l^σ in $C[l^\sigma]$
- If $s_1, \ldots, s_n$ are composable redexes we write $s_1; \ldots; s_n$ for the resulting derivation
- We sometimes give derivations names $d : s_1; \ldots; s_n$
- We write $|d|$ for the number of steps in d

Example

$$\begin{array}{llcl} \rho: & a(x,0) & \to & x \\ & a(x,s(y)) & \to & s(a(x,y)) \\ & m(x,0) & \to & 0 \\ & m(x,s(y)) & \to & a(m(x,y),x) \end{array}$$

Reduction step $(C = s(\Box),\ \sigma = \{x/s(s(0))\})$

$$s(\underline{a(s(s(0)),s(0))}) \quad \to_\rho \quad s(s(a(s(s(0)),0)))$$

Term Rewrite System

A Term Rewrite System is a pair $\mathcal{R} = \langle \Sigma, R \rangle$ of a signature Σ and a set of reduction rules R for Σ

The one-step reduction relation of $\mathcal{R}$ is defined as the union

$$\rightarrow = \bigcup \{ \rightarrow_s \mid M \rightarrow_s N,\ s \text{ a } \rho\text{-redex in } M, \rho \in R \}$$

Note:

- $\langle \mathcal{T}(\Sigma), \rightarrow \rangle$ is an ARS
- Thus all concepts of ARS are applicable to TRS

Confluence - Reminder

Confluence (CR)

$$
\begin{tikzcd}
 & a \arrow[dl, two heads] \arrow[dr, two heads] & \\
b \arrow[dr, two heads, dotted] & & c \arrow[dl, two heads, dotted] \\
 & d &
\end{tikzcd}
$$

$\forall\ a, b, c$ s.t. $a \twoheadrightarrow b$ and $a \twoheadrightarrow c$, $\exists d$ s.t. $b \twoheadrightarrow d$ and $c \twoheadrightarrow d$

Techniques for Proving Confluence

- Abstract: Formulated for Abstract Reduction Systems
- Concrete: Formulated for Term Rewrite Systems

Techniques for Proving Confluence

- Abstract: Confluence by
  - Strong confluence
  - Equivalence
  - ...
- Concrete: Confluence by
  - Critical pairs
  - Orthogonality
  - ...

Strong Confluence [Huet1980]

$$
\begin{tikzcd}
 & \bullet \arrow[dl] \arrow[dr] & \\
\bullet \arrow[dr, dotted, two heads] & & \bullet \arrow[dl, dotted, "="] \\
 & \bullet &
\end{tikzcd}
$$

$$
\begin{array}{lcl}
f(x,x) & \longrightarrow & g(x) \\
f(x,y) & \longrightarrow & g(y) \\
g(x) & \longrightarrow & f(x,a)
\end{array}
$$

Beware of asymmetry!

Equivalence

$\langle A, \rightarrow_A \rangle, \langle B, \rightarrow_B \rangle$ ARS

1. $\rightarrow_A \subseteq \rightarrow_B \subseteq \twoheadrightarrow_A$ and
2. $\rightarrow_B$ strongly confluent

Then $\rightarrow_A$ is confluent

Proof

1. (1) implies $\twoheadrightarrow_A = \twoheadrightarrow_B$
2. (2) implies $\rightarrow_B$ confluent
3. Result follows from (1),(2)

Equivalence

Let R be the TRS

$$\begin{array}{lcl} f(x) & \rightarrow & g(x,x) \\ g(x,y) & \rightarrow & f(y) \end{array}$$

Define $\rightrightarrows$ as

$$\frac{}{x \rightrightarrows x} \qquad \frac{M \rightrightarrows M'}{f(M) \rightrightarrows f(M')}$$

$$\frac{M \rightrightarrows M' \quad N \rightrightarrows N'}{g(M,N) \rightrightarrows g(M',N')}$$

1. Show $\rightarrow_R \subseteq \rightrightarrows \subseteq \twoheadrightarrow_R$
2. Show $\rightrightarrows$ is strongly confluent
3. Conclude R is confluent

Techniques for Proving Confluence

- Abstract: Confluence by
  - Strong confluence
  - Equivalence
  - ...
- Concrete: Confluence by
  - Critical pairs
  - Orthogonality
  - ...

Critical Pairs

Overlap between two left-hand sides of rewrite rules

$l \rightarrow r$ and $g \rightarrow d$ variable disjoint rewrite rules. A critical pair between them is a pair $\langle l^\sigma[d^\sigma]_p, r\sigma \rangle$ where

1. $p \in \mathrm{pos}(l)$ and $l\,|_p$ is not a variable
2. σ is a MGU of $l\,|_p$ and g

Note:

$$\begin{tikzcd}
 & l^\sigma = l^\sigma[g^\sigma]_p \arrow[dl] \arrow[dr] & \\
l^\sigma[d^\sigma]_p & & r^\sigma
\end{tikzcd}$$

Example

Rewrite system

$$
\begin{array}{lcl}
\neg(\mathit{true}) & \longrightarrow & \mathit{false} \\
\neg(\mathit{false}) & \longrightarrow & \mathit{true} \\
\neg(\neg(x)) & \longrightarrow & x \\
\mathit{and}(\mathit{true}, x) & \longrightarrow & x \\
\mathit{and}(\mathit{false}, x) & \longrightarrow & \mathit{false}
\end{array}
$$

Critical pairs (are there others?)

$$
\begin{tikzcd}
& \neg(\neg(\mathit{true})) \arrow[dl] \arrow[dr] & \\
\neg(\mathit{false}) & & \mathit{true}
\end{tikzcd}
\qquad
\begin{tikzcd}
& \neg(\neg(\mathit{false})) \arrow[dl] \arrow[dr] & \\
\neg(\mathit{true}) & & \mathit{false}
\end{tikzcd}
$$

Example

Rewrite system

$$x \oplus (y \oplus z) \quad \longrightarrow \quad (x \oplus y) \oplus z$$

Critical pairs

$$\begin{tikzcd}[column sep=small] & x \oplus (y \oplus (z \oplus w)) \arrow[dl] \arrow[dr] & \\ x \oplus ((y \oplus z) \oplus w) & & (x \oplus y) \oplus (z \oplus w) \end{tikzcd}$$

WCR by Critical Pairs

Thm

$\mathcal{R}$ is WCR iff every critical pair is joinable

Proof

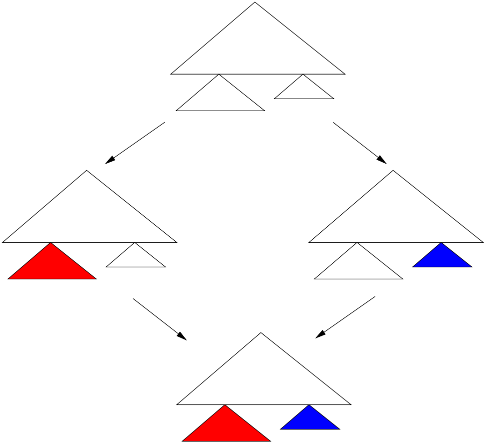
$\Rightarrow$) Trivial

$\Leftarrow$) Take an arbitrary peak

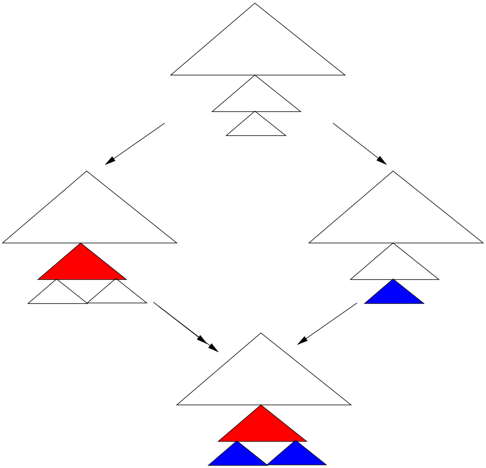
$$\begin{tikzcd} & M \arrow[dl, "s_0"'] \arrow[dr, "s_1"] & \\ P_0 & & P_1 \end{tikzcd}$$

and consider all possible cases in which this arises

s_0 and s_1 are disjoint - Direct

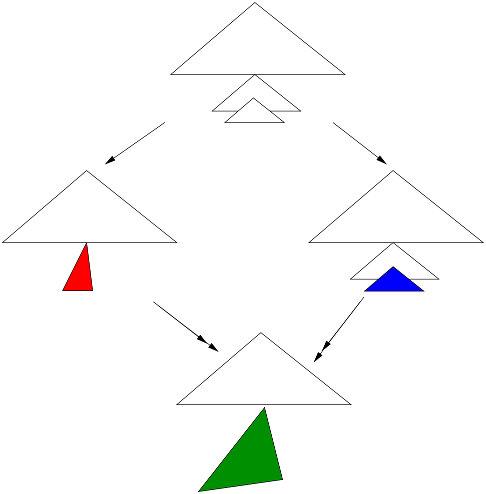


s_0 and s_1 are nested - Direct



The bottom-right arrow may have to perform multiple steps if the rewrite rules are not left-linear

s_0 and s_1 overlap - Use hypothesis



Decidable Case of Confluence

Thm

Let $\mathcal{R}$ be finite and SN. Then confluence is decidable.

Proof

1. Generate all critical pairs
2. For each critical pair $\langle u, v \rangle$ reduce u and v to their normal forms $\overline{u}, \overline{v}$
   1. if $\overline{u} \neq \overline{v}$ for some $\langle u, v \rangle$ then fail
   2. Otherwise, the system is confluent

Orthogonal TRS

A TRS is called orthogonal (OTRS) if it is

1. left-linear and
2. without critical pairs

Thm

Orthogonal TRS are confluent

- Note that, in contrast to the previous result, we do not require the TRS to be SN
- Proof relies on the fact that coinitial, diverging reduction steps can always be joined
- More on orthogonal TRS later

What is the Lambda Calculus

- A model of computation
- Concise and expressive
- Strong connections to proof theory and category theory
- Shown to be equivalent to Turing Machines
- Considered a suitable abstract model of programming languages

Lambda calculus
$+$
Your new programming construct
$=$
Good testbed

Informal Introduction

- Fundamental construction: abstraction

$$\lambda x.x + 1$$

  - Similar to $f(x) = x + 1$ except that it is "anonymous"

- Fundamental operation: application of functions to arguments

$$(\lambda x.x + 1)\,2$$

- Both of these combined in their purest form:
  - Everything is a function!

Syntax

λ-terms ($\mathcal{T}(\lambda)$)

$$
\begin{array}{lcll}
M & ::= & x & \text{variable} \\
 & \mid & M\,N & \text{application} \\
 & \mid & \lambda x.M & \text{abstraction}
\end{array}
$$

- In an abstraction
  - x is the (formal) parameter and M is the body
  - λx binds all occurrences of x in M not under another λx
  - notion of free and bound variables similar to that of first-order logic
  - free variables of M: $\mathsf{fv}(M)$
- In an application N is called an argument

Examples of λ-Terms

- $\lambda x.x$
- x
- $\lambda x.x\,x$ (self-application!)
- $\lambda x.\lambda y.x$
- $(\lambda x.x)(\lambda x.x)$
- $x\,y$
- $\lambda x.x = \lambda y.y$ (terms differing only in the name of bound variables are considered equal; this is called α-equivalence)

Reduction

$$\beta: \quad (\lambda x.M)\, N \quad \rightarrow \quad M\{x/N\}$$

1. Substitution: $M\{x/N\}$ denotes the term M where all free occurrences of x are replaced by N
2. Substitution may need to rename bound variables in order to avoid variable capture

$$\begin{array}{ll} (\lambda x.y)\{y/x\} = \lambda x.x & \text{No! Variable capture} \\ (\lambda x.y)\{y/x\} = \lambda z.x & \text{Rename. Ok!} \end{array}$$

We ignore extensionality in our presentation

$$\eta: \quad \lambda x.M\, x \quad \rightarrow \quad M \quad \text{if } x \notin \mathrm{fv}(M)$$

Example

1. $I\,y \to x\{x/y\} = y$ (where $I = \lambda x.x$)
2. $\Delta\,(I\,y) \to (I\,y)(I\,y)$ (where Δ is $\lambda x.x\,x$)
3. $\Delta\,(I\,y) \to \Delta y \to y\,y$
4. $(\lambda x.z)(I\,y) \to z$
5. $\omega\,\omega \to (x\,x)\{x/\omega\} = \omega\,\omega$ (where $\omega = \lambda x.x\,x$)
6. $(\lambda x.z)\,(\omega\,\omega) \to z$

Two Basic Properties

Lemma

β is not WN (hence not SN)

Proof (counterexample)

$\omega\,\omega \rightarrow \omega\,\omega \rightarrow \omega\,\omega \rightarrow \ldots$ (where $\omega = \lambda x.x\,x$)

Thm

β is confluent

Proof

Use confluence by equivalence technique

De Bruijn Indices ("I don't really like de Bruijn indices myself" (N. de Bruijn))

- The idea: replace variable names by reference to declaration point

$$\begin{array}{lll} \lambda x.x & \text{becomes} & \lambda 1 \\ \lambda x.\lambda y.x & \text{becomes} & \lambda\lambda 2 \end{array}$$

- Consequence: Renaming not necessary (replaced by index adjustment)

$$\lambda x.(\lambda y.\lambda z.y)\, x \to_{\beta} \lambda x.\lambda z.x$$

becomes

$$\lambda(\lambda\lambda 2)\, 1 \to_{\beta_{DB}} \lambda((\lambda 2)\{1/1\}) = \lambda\lambda 2\{2/2\} = \lambda\lambda 2$$

De Bruijn Indices

$$\beta_{DB} : \quad (\lambda M)\, N \quad \rightarrow \quad M\{\{1/N\}\}$$

- $M\{\{1/N\}\}$ is substitution on terms with indices
- β is isomorphic to β_{DB}
- β is easier for study purposes
- β_{DB} is easier for implementation

Lambda Calculus vs First-Order Rewriting

- First-Order Rewriting
  - ★ natural model of computation
  - ★ concise representation of algebraic data types
  - X functions are not treated as data
- Lambda Calculus
  - ★ natural model for reasoning about functions
  - ★ can encode programs, derivations, specifications
  - X inefficient representation of algebraic types
  - X more complex metatheory