O • NICTA

COMP 4161 NICTA Advanced Course

Advanced Topics in Software Verification

Toby Murray, June Andronick, Gerwin Klein



Slide 1

Content	NICTA
→ Intro & motivation, getting started	[1]
→ Foundations & Principles	
 Lambda Calculus, natural deduction 	[1,2]
Higher Order Logic	[3 ^a]
Term rewriting	[4]
➔ Proof & Specification Techniques	
 Inductively defined sets, rule induction 	[5]
 Datatypes, recursion, induction 	[6, 7]
 Hoare logic, proofs about programs, C verification 	[8 ^b ,9]
(mid-semester break)	
 Writing Automated Proof Methods 	[10]
 Isar, codegen, typeclasses, locales 	[11 ^c ,12]

a a1 due; ba2 due; ca3 due

Slide 2

Last Time

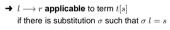


NICTA

- → Equations and Term Rewriting
- → Confluence and Termination of reduction systems
- → Term Rewriting in Isabelle

Slide 3

Applying a Rewrite Rule



→ Result: $t[\sigma r]$

→ Equationally: $t[s] = t[\sigma r]$

Example:

Rule: $0 + n \longrightarrow n$

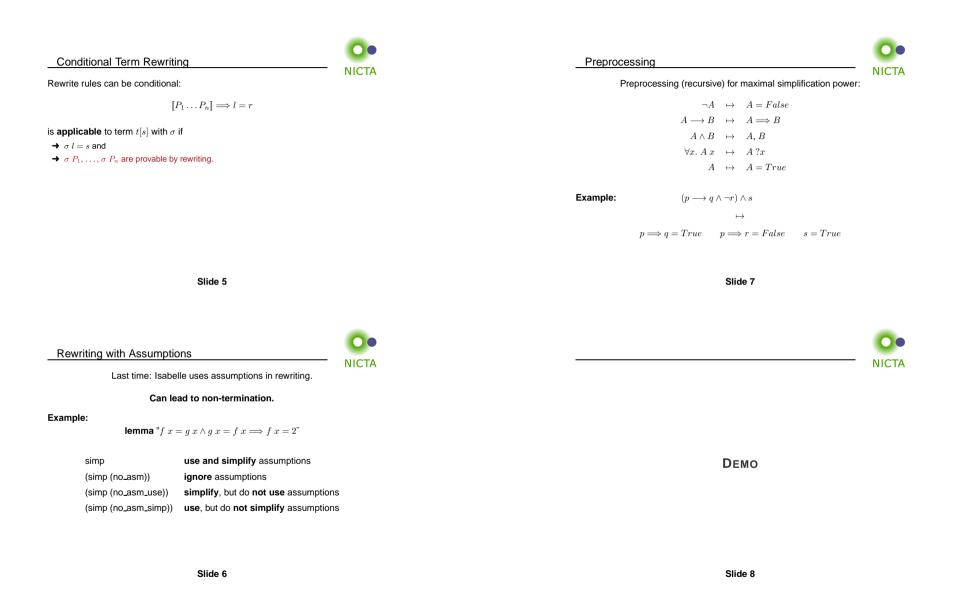
Term: a + (0 + (b + c))

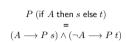
Substitution: $\sigma = \{n \mapsto b + c\}$

Result: a + (b + c)

Slide 4

1





Automatic

 $\begin{array}{l} P \ (\mathsf{case} \ e \ \mathsf{of} \ 0 \ \Rightarrow \ a \ | \ \mathsf{Suc} \ n \ \Rightarrow \ b) \\ = \\ (e = 0 \longrightarrow P \ a) \land (\forall n. \ e = \mathsf{Suc} \ n \longrightarrow P \ b) \end{array}$

Manually: apply (simp split: nat.split)

Similar for any data type t: t.split

Slide 9

	0.
Congruence Rules	NICTA
congruence rules are about using context	
Example : in $P \longrightarrow Q$ we could use P to simplify terms in Q	
For \Longrightarrow hardwired (assumptions used in rewriting)	
For other operators expressed with conditional rewriting.	
$\textbf{Example:} [\![P=P';P' \Longrightarrow Q=Q']\!] \Longrightarrow (P \longrightarrow Q) = (P' \longrightarrow Q')$	

Read: to simplify $P \longrightarrow Q$

Case splitting with simp

- → first simplify P to P'
- → then simplify Q to Q' using P' as assumption
- \Rightarrow the result is $P' \longrightarrow Q'$



More Congruence

Sometimes useful, but not used automatically (slowdown): **conj_cong**: $\llbracket P = P'; P' \Longrightarrow Q = Q' \rrbracket \Longrightarrow (P \land Q) = (P' \land Q')$

Context for if-then-else: **if_cong**: $[b = c; c \Longrightarrow x = u; \neg c \Longrightarrow y = v] \Longrightarrow$ (if b then x else y) = (if c then u else v)

Prevent rewriting inside then-else (default): **if_weak_cong**: $b = c \implies$ (if b then x else y) = (if c then x else y)

- → declare own congruence rules with [cong] attribute
- → delete with [cong del]
- → use locally with e.g. apply (simp cong: <rule>)

Slide 11

Ordere	d rewriting	— ı
Problem: a	$x + y \longrightarrow y + x$ does not terminate	
Solution:	use permutative rules only if term becomes	
	lexicographically smaller.	
Example:	$b + a \rightsquigarrow a + b$ but not $a + b \rightsquigarrow b + a$.	
For types n	at, int etc:	
 lemma 	s add_ac sort any sum (+)	
 lemma 	s times_ac sort any product (*)	
Example:	apply (simp add: add_ac) yields	
Example.		

NICTA

AC Rules

Example for associative-commutative rules: Associative: $(x \odot y) \odot z = x \odot (y \odot z)$ Commutative: $x \odot y = y \odot x$

. .

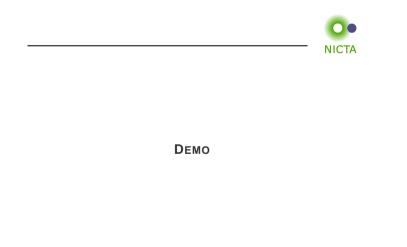
These 2 rules alone get stuck too early (not confluent).

 $\begin{array}{lll} \mbox{Example:} & (z \odot x) \odot (y \odot v) \\ \mbox{We want:} & (z \odot x) \odot (y \odot v) = v \odot (x \odot (y \odot z)) \\ \mbox{We get:} & (z \odot x) \odot (y \odot v) = v \odot (y \odot (x \odot z)) \end{array}$

We need: AC rule $x \odot (y \odot z) = y \odot (x \odot z)$

If these 3 rules are present for an AC operator Isabelle will order terms correctly

Slide 13



Slide 14

Back to Confluence

Last time: confluence in general is undecidable. But: confluence for terminating systems is decidable! Problem: overlapping lhs of rules.

Definition:

Let $l_1 \longrightarrow r_1$ and $l_2 \longrightarrow r_2$ be two rules with disjoint variables. They form a **critical pair** if a non-variable subterm of l_1 unifies with l_2 .

Example:

Rules: (1) $f x \longrightarrow a$ (2) $g y \longrightarrow b$ (3) $f (g z) \longrightarrow b$ Critical pairs:

(1)+(3)	$\{x \mapsto g \ z\}$	$a \xleftarrow{(1)}$	f(g z)	$\stackrel{(3)}{\longrightarrow} b$
(3)+(2)	$\{z\mapsto y\}$	$b \stackrel{(3)}{\longleftarrow}$	f~(g~y)	$\stackrel{(2)}{\longrightarrow} f \; b$

Slide 15

Completion



NICTA

(1) $f x \longrightarrow a$ (2) $g y \longrightarrow b$ (3) $f (g z) \longrightarrow b$ is not confluent

But it can be made confluent by adding rules!

How: join all critical pairs

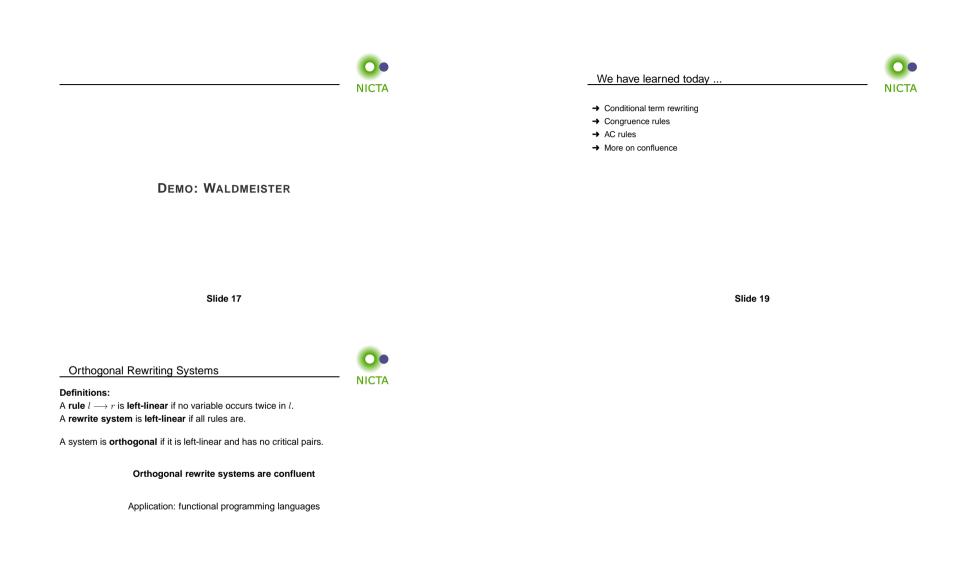
Example:

(1)+(3) $\{x \mapsto g \ z\}$ $a \stackrel{(1)}{\longleftarrow} f(g \ z) \stackrel{(3)}{\longrightarrow} b$

shows that a = b (because $a \stackrel{*}{\longleftrightarrow} b$), so we add $a \longrightarrow b$ as a rule

This is the main idea of the Knuth-Bendix completion algorithm.

Slide 16



Slide 18