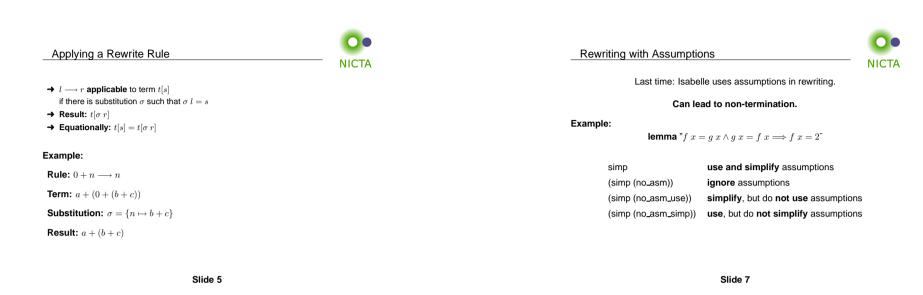




Slide 4



Conditional Term Rewriting	NICTA
vrite rules can be conditional:	NICIA
write rules can be conditional:	
$\llbracket P_1 \dots P_n \rrbracket \Longrightarrow l = r$	
<b>ipplicable</b> to term $t[s]$ with $\sigma$ if	
$\sigma \ l = s$ and	
$\sigma P_1, \ldots, \sigma P_n$ are provable by rewriting.	

NICTA

Preprocessing (recursive) for maximal simplification power:

 $\neg A \quad \mapsto \quad A = False$   $A \longrightarrow B \quad \mapsto \quad A \Longrightarrow B$   $A \land B \quad \mapsto \quad A, B$   $\forall x. \ A \ x \quad \mapsto \quad A \ ?x$   $A \quad \mapsto \quad A = True$ 

Preprocessing

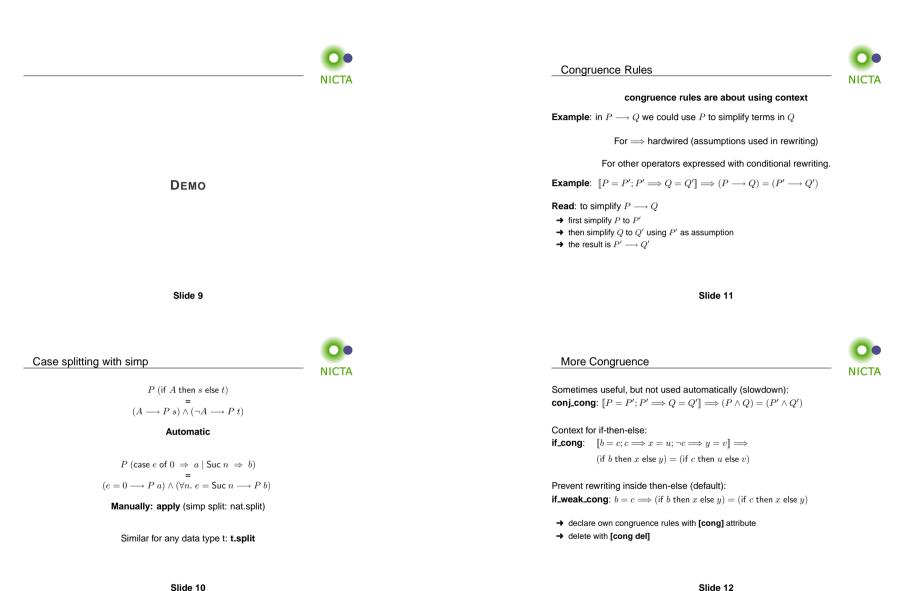
Example:

 $(p \longrightarrow q \land \neg r) \land s$  $\mapsto$ 

 $p \Longrightarrow q = True$  r = False s = True

Slide 6

Slide 8



Slide 12

Ordered rewriting			)
<b>Problem:</b> $x + y \longrightarrow y + x$ does not terminate			
Solution: use permutative rules only if term becomes lexicographically smaller.			
<b>Example:</b> $b + a \rightsquigarrow a + b$ but not $a + b \rightsquigarrow b + a$ .			
<ul> <li>For types nat, int etc:</li> <li>lemmas add_ac sort any sum (+)</li> <li>lemmas times_ac sort any product (*)</li> </ul>		Dемо	
<b>Example:</b> apply (simp add: add_ac) yields $(b+c) + a \rightsquigarrow \cdots \rightsquigarrow a + (b+c)$			
Slide 13		Slide 15	
AC Rules	NICTA	Back to Confluence	)
Example for associative-commutative rules:Associative: $(x \odot y) \odot z = x \odot (y \odot z)$ Commutative: $x \odot y = y \odot x$		Last time: confluence in general is undecidable. But: confluence for terminating systems is decidable! Problem: overlapping lhs of rules.	
These 2 rules alone get stuck too early (not confluent). Example: $(z \odot x) \odot (y \odot v)$		<b>Definition:</b> Let $l_1 \longrightarrow r_1$ and $l_2 \longrightarrow r_2$ be two rules with disjoint variables. They form a <b>critical pair</b> if a non-variable subterm of $l_1$ unifies with $l_2$ .	

Example:	$(z \odot x) \odot (y \odot v)$
We want:	$(z\odot x)\odot (y\odot v)=v\odot (x\odot (y\odot z))$
We get:	$(z\odot x)\odot (y\odot v)=v\odot (y\odot (x\odot z))$

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

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

Slide 14

Slide 16

(1)+(3)  $\{x \mapsto g z\}$   $a \stackrel{(1)}{\leftarrow} f g t \stackrel{(3)}{\longrightarrow} b$ (3)+(2)  $\{z \mapsto y\}$   $b \stackrel{(3)}{\leftarrow} f g t \stackrel{(2)}{\longrightarrow} b$ 

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

Example:

## Completion

**NICTA** 

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

Slide 17

## Example:

(1)+(3)  $\{x \mapsto g z\}$   $a \stackrel{(1)}{\leftarrow} f g t \stackrel{(3)}{\longrightarrow} b$ shows that a = b (because  $a \stackrel{\leftrightarrow}{\longrightarrow} b$ ), so we add  $a \longrightarrow b$  as a rule

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

Orthogonal Rewriting Systems



**Definitions:** A **rule**  $l \rightarrow r$  is **left-linear** if no variable occurs twice in l. A **rewrite system** is **left-linear** if all rules are.

A system is orthogonal if it is left-linear and has no critical pairs.

Orthogonal rewrite systems are confluent

Application: functional programming languages

Slide 19





→ Conditional term rewriting

- → Congruence rules
- → AC rules
- → More on confluence

**DEMO: WALDMEISTER** 



Slide 20

10