



university of  
 groningen

# Basic Approaches to the Semantics of Computation (BaSC)

## Lecture 2: Inference and Unification

Jorge A. Pérez

Bernoulli Institute for Mathematics, Computer Science, and AI  
University of Groningen, Groningen, the Netherlands



(prod)

$$\frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \otimes E_1 \longrightarrow n} \quad n = n_0 \cdot n_1$$

$\overset{?}{\rightsquigarrow}$

(prod)

$$\frac{1 \oplus 2 \longrightarrow 3 \quad 3 \oplus 4 \longrightarrow 7}{(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21}$$



## Inference Rule

$$\begin{array}{c} (prod) \\ \frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \otimes E_1 \longrightarrow n} \quad n = n_0 \cdot n_1 \end{array}$$

$\overset{?}{\rightsquigarrow}$

## Rule Instance

$$\begin{array}{c} (prod) \\ \frac{1 \oplus 2 \longrightarrow 3 \quad 3 \oplus 4 \longrightarrow 7}{(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21} \end{array}$$

## Inference Rule

$$\begin{array}{c} (prod) \\ \frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \otimes E_1 \longrightarrow n} \quad n = n_0 \cdot n_1 \end{array}$$

$\rightsquigarrow ?$

## Rule Instance

$$\begin{array}{c} (prod) \\ \frac{1 \oplus 2 \longrightarrow 3 \quad 3 \oplus 4 \longrightarrow 7}{(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21} \end{array}$$

## Today:

- ▶ Goal-oriented (or bottom-up) derivations
- ▶ Signatures and substitutions
- ▶ Unification (key ideas)
- ▶ Inference rules, derivations, an inline notation
- ▶ Logic programming (key ideas)

# Applying SOS Rules



**Step 1.** A goal

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow m$$



**Step 2.** Take a rule

$$\begin{array}{c} (prod) \\ \frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \otimes E_1 \longrightarrow n} \quad n = n_0 \cdot n_1 \end{array}$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow m$$

# Applying SOS Rules



**Step 3.** Unify (if possible)

$$\begin{array}{c} (prod) \\ \frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \otimes E_1 \longrightarrow n} \quad n = n_0 \cdot n_1 \end{array}$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow m$$

$$E_0 = 1 \oplus 2$$

$$E_1 = 3 \oplus 4$$

$$n = m$$



## Step 4. Instantiate

$$\begin{array}{c} (prod) \\ (1 \oplus 2) \longrightarrow n_0 \quad (3 \oplus 4) \longrightarrow n_1 \\ \hline (1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow m \end{array} \quad m = n_0 \cdot n_1$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow m$$





## Step 5. Recursively solve sub-goals

(*prod*)

$$(1 \oplus 2) \longrightarrow n_0$$

$$(3 \oplus 4) \longrightarrow n_1$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow m$$

$$m = n_0 \cdot n_1$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow m$$



## Step 6. Combine results

*(prod)*

$$(1 \oplus 2) \longrightarrow 3$$

$$(3 \oplus 4) \longrightarrow 7$$

$$\frac{(1 \oplus 2) \longrightarrow 3 \quad (3 \oplus 4) \longrightarrow 7}{(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow m} \quad m = 3 \cdot 7$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow m$$



## Step 7. Return results

*(prod)*

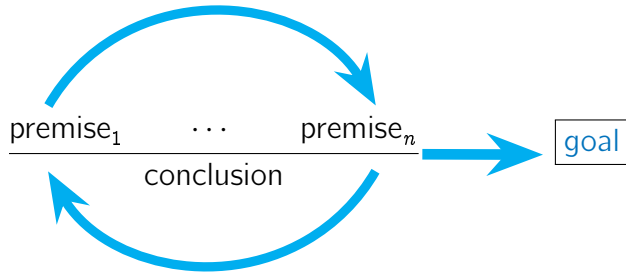
$$(1 \oplus 2) \longrightarrow 3$$

$$(3 \oplus 4) \longrightarrow 7$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21$$

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21$$

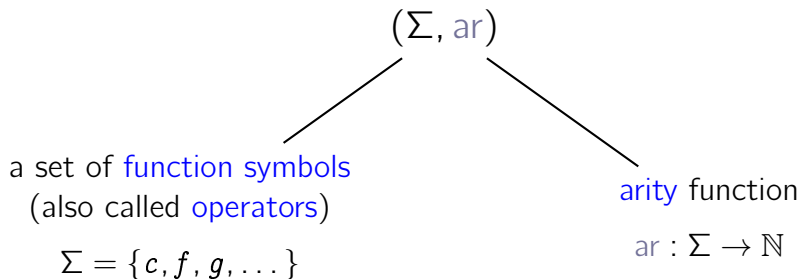
# Deduction Process: Goal-Oriented





# Signatures

# Unsorted Signature



Each symbol has an arity. Examples:

- ▶  $\text{ar}(c) = 0$ : constant
- ▶  $\text{ar}(f) = 1$ : unary
- ▶  $\text{ar}(g) = 2$ : binary
- ▶  $\text{ar}(h) = 3$ : ternary



- Given  $(\Sigma, ar)$ , we can define

$$\begin{aligned}\Sigma_n &\triangleq ar^{-1}(n) \\ &= \{f \in \Sigma \mid ar(f) = n\}\end{aligned}$$

That is,  $\Sigma_n$  denotes the set of operators of arity  $n$

- A signature can then be defined as a **family** of sets of operators, indexed by their arity:

$$\Sigma = \{\Sigma_n\}_{n \in \mathbb{N}}$$

# Terms Over a Signature



Consider given:

$$\Sigma = \{\Sigma_n\}_{n \in \mathbb{N}}$$

a signature

$$X = \{x, y, z, \dots\}$$

an infinite set of **variables**

Let  $T_{\Sigma, X}$  denote the set of all **terms** over  $\Sigma, X$ .



# Terms Over a Signature



Consider given:

$$\Sigma = \{\Sigma_n\}_{n \in \mathbb{N}}$$

a signature

$$X = \{x, y, z, \dots\}$$

an infinite set of **variables**

Let  $T_{\Sigma, X}$  denote the set of all **terms** over  $\Sigma, X$ . The least set such that:

- ▶ if  $x \in X$ , then  $x \in T_{\Sigma, X}$
- ▶ if  $c \in \Sigma_0$ , then  $c \in T_{\Sigma, X}$
- ▶ if  $f \in \Sigma_n$  and  $t_1, \dots, t_n \in T_{\Sigma, X}$ , then  $f(t_1, \dots, t_n) \in T_{\Sigma, X}$

# Terms Over a Signature



Consider given:

$$\Sigma = \{\Sigma_n\}_{n \in \mathbb{N}}$$

a signature

$$X = \{x, y, z, \dots\}$$

an infinite set of **variables**

Let  $T_{\Sigma, X}$  denote the set of all **terms** over  $\Sigma, X$ . The least set such that:

- ▶ if  $x \in X$ , then  $x \in T_{\Sigma, X}$
- ▶ if  $c \in \Sigma_0$ , then  $c \in T_{\Sigma, X}$
- ▶ if  $f \in \Sigma_n$  and  $t_1, \dots, t_n \in T_{\Sigma, X}$ , then  $f(t_1, \dots, t_n) \in T_{\Sigma, X}$

Put differently:

$$T_{\Sigma, X} \ni t ::= \underbrace{x}_{x \in X} \mid \underbrace{c}_{f \in \Sigma_n} \mid \underbrace{f(t_1, \dots, t_n)}_{f \in \Sigma_n}$$

# Variables



Assume

$$\Sigma = \{\Sigma_n\}_{n \in \mathbb{N}} \quad X = \{x, y, z, \dots\}$$

Given  $t \in T_{\Sigma, X}$ , we write  $\text{vars}(t)$  to denote the set of variables that appear in  $t$ .

$$\text{vars} : T_{\Sigma, X} \rightarrow \mathcal{P}(X)$$

$$\text{vars}(x) \triangleq \{x\}$$

$$\text{vars}(c) \triangleq \emptyset$$

$$\text{vars}(f(t_1, \dots, t_n)) \triangleq \bigcup_{i=1}^n \text{vars}(t_i)$$

# Variables



Assume

$$\Sigma = \{\Sigma_n\}_{n \in \mathbb{N}} \quad X = \{x, y, z, \dots\}$$

Given  $t \in T_{\Sigma, X}$ , we write  $\text{vars}(t)$  to denote the set of variables that appear in  $t$ .

$$\text{vars} : T_{\Sigma, X} \rightarrow \mathcal{P}(X)$$

$$\text{vars}(x) \triangleq \{x\}$$

$$\text{vars}(c) \triangleq \emptyset$$

$$\text{vars}(f(t_1, \dots, t_n)) \triangleq \bigcup_{i=1}^n \text{vars}(t_i)$$

We can also define **closed terms**:

$$T_{\Sigma} \triangleq \{t \in T_{\Sigma, X} \mid \text{vars}(t) = \emptyset\}$$

# Complete the Schema (1/2)



A signature:

$$\Sigma_0 = \{0\}$$

$$\Sigma_1 = \{\text{succ}\}$$

$$\Sigma_2 = \{\text{plus}\}$$

$$\Sigma_n = \emptyset \quad (\text{if } n > 2)$$

$t$	$t \in ?$	$\text{vars}(t)$
$0$	$\begin{array}{cc} \boxtimes & \square \\ T_\Sigma & T_{\Sigma, X} \end{array}$	$\emptyset$
$x$	$\begin{array}{cc} \square & \boxtimes \\ T_\Sigma & T_{\Sigma, X} \end{array}$	$\{x\}$
$\text{succ}(0)$	$\begin{array}{cc} \square & \square \\ T_\Sigma & T_{\Sigma, X} \end{array}$	
$\text{succ}(x)$	$\begin{array}{cc} \square & \square \\ T_\Sigma & T_{\Sigma, X} \end{array}$	
$\text{succ}(\text{plus}(0), x)$	$\begin{array}{cc} \square & \square \\ T_\Sigma & T_{\Sigma, X} \end{array}$	

## Complete the Schema (2/2)



$t$	$t \in ?$	$\text{vars}(t)$
$\text{plus}(\text{succ}(0), x)$	<div><input type="checkbox"/> <input type="checkbox"/> <math>T_\Sigma</math> <math>T_{\Sigma, X}</math></div>	
$\text{succ}(\text{succ}(0), \text{plus}(x))$	<div><input type="checkbox"/> <input type="checkbox"/> <math>T_\Sigma</math> <math>T_{\Sigma, X}</math></div>	
$\text{succ}(\text{plus}(w, z))$	<div><input type="checkbox"/> <input type="checkbox"/> <math>T_\Sigma</math> <math>T_{\Sigma, X}</math></div>	
$\text{plus}(\text{plus}(x, \text{succ}(y)), \text{plus}(0, \text{succ}(x)))$	<div><input type="checkbox"/> <input type="checkbox"/> <math>T_\Sigma</math> <math>T_{\Sigma, X}</math></div>	

# Substitutions



- ▶ A **substitution**  $\rho$  assigns terms to variables:  $\rho : X \rightarrow T_{\Sigma, X}$
- ▶ We only consider substitutions that are identity everywhere, except for a finite number of cases, written:

$$\rho = [x_1 = t_1, \dots, x_n = t_n]$$

all different

$$\rho(x) = \begin{cases} t_i & \text{if } x = x_i, \\ x & \text{otherwise.} \end{cases}$$



- ▶ A **substitution**  $\rho$  assigns terms to variables:  $\rho : X \rightarrow T_{\Sigma, X}$
- ▶ We only consider substitutions that are identity everywhere, except for a finite number of cases, written:

$$\rho = [x_1 = t_1, \dots, x_n = t_n]$$

all different

$$\rho(x) = \begin{cases} t_i & \text{if } x = x_i, \\ x & \text{otherwise.} \end{cases}$$

- ▶ Overloaded notation for the lifted function  $\rho : T_{\Sigma, X} \rightarrow T_{\Sigma, X}$
- ▶  $\rho(t)$  denotes the term obtained by simultaneous application of  $\rho$  to all variable occurrences in  $t$ . An alternative notation:  $t\rho$ .



# Example



Given

$$\rho \triangleq [x = \text{succ}(y), y = 0]$$
$$t \triangleq \text{plus}(\text{plus}(x, y), \text{succ}(x))$$

Then

$$t\rho =$$

# Example



Given

$$\begin{aligned}\rho &\triangleq [x = \text{succ}(y), y = 0] \\ t &\triangleq \text{plus}(\text{plus}(x, y), \text{succ}(x))\end{aligned}$$

Then

$$t\rho = \text{plus}(\text{plus}(\text{succ}(y), 0), \text{succ}(\text{succ}(y)))$$

# More General Than (mgt) Relation



- ▶ We say that  $t$  is **more general than**  $t'$ , written  $t$  **mgt**  $t'$ , if  $\exists \rho. t' = t\rho$ .
- ▶ If  $t$  **mgt**  $t'$  then  $t'$  is an **instance** of  $t$

## Example

mgt?

$\text{plus}(x, \text{succ}(y))$   
 $\text{plus}(0, x)$   
 $\text{plus}(y, 0)$   
 $\text{plus}(0, x), \text{plus}(y, 0)$

$\text{plus}(0, \text{succ}(\text{succ}(z)))$   
 $\text{plus}(y, 0)$   
 $\text{plus}(0, x)$   
 $\text{plus}(0, 0)$

# More General Than (mgt) Relation



- ▶ We say that  $t$  is **more general than**  $t'$ , written  $t$  mgt  $t'$ , if  $\exists \rho. t' = t\rho$ .
- ▶ If  $t$  mgt  $t'$  then  $t'$  is an **instance** of  $t$

## Example

	mgt?	
plus( $x$ , succ( $y$ ))	✓	plus(0, succ(succ( $z$ )))
plus(0, $x$ )		plus( $y$ , 0)
plus( $y$ , 0)		plus(0, $x$ )
plus(0, $x$ ), plus( $y$ , 0)		plus(0, 0)

# More General Than (mgt) Relation



- ▶ We say that  $t$  is **more general than**  $t'$ , written  $t$  mgt  $t'$ , if  $\exists \rho. t' = t\rho$ .
- ▶ If  $t$  mgt  $t'$  then  $t'$  is an **instance** of  $t$

## Example

	mgt?	
<code>plus(<math>x</math>, succ(<math>y</math>))</code>	✓	<code>plus(0, succ(succ(<math>z</math>)))</code>
<code>plus(0, <math>x</math>)</code>	✗	<code>plus(<math>y</math>, 0)</code>
<code>plus(<math>y</math>, 0)</code>		<code>plus(0, <math>x</math>)</code>
<code>plus(0, <math>x</math>), plus(<math>y</math>, 0)</code>		<code>plus(0, 0)</code>

# More General Than (mgt) Relation



- ▶ We say that  $t$  is **more general than**  $t'$ , written  $t$  mgt  $t'$ , if  $\exists \rho. t' = t\rho$ .
- ▶ If  $t$  mgt  $t'$  then  $t'$  is an **instance** of  $t$

## Example

	mgt?	
<code>plus(<math>x</math>, succ(<math>y</math>))</code>	✓	<code>plus(0, succ(succ(<math>z</math>)))</code>
<code>plus(0, <math>x</math>)</code>	✗	<code>plus(<math>y</math>, 0)</code>
<code>plus(<math>y</math>, 0)</code>	✗	<code>plus(0, <math>x</math>)</code>
<code>plus(0, <math>x</math>), plus(<math>y</math>, 0)</code>		<code>plus(0, 0)</code>

# More General Than (mgt) Relation



- ▶ We say that  $t$  is **more general than**  $t'$ , written  $t$  mgt  $t'$ , if  $\exists \rho. t' = t\rho$ .
- ▶ If  $t$  mgt  $t'$  then  $t'$  is an **instance** of  $t$

## Example

	mgt?	
<code>plus(<math>x</math>, succ(<math>y</math>))</code>	✓	<code>plus(0, succ(succ(<math>z</math>)))</code>
<code>plus(0, <math>x</math>)</code>	✗	<code>plus(<math>y</math>, 0)</code>
<code>plus(<math>y</math>, 0)</code>	✗	<code>plus(0, <math>x</math>)</code>
<code>plus(0, <math>x</math>), plus(<math>y</math>, 0)</code>	✓	<code>plus(0, 0)</code>



- ▶ mgt is transitive and reflexive
  - ▶  $t \text{ mgt } t$
  - ▶ if  $(t_1 \text{ mgt } t_2)$  and  $(t_3 \text{ mgt } t_3)$  then  $(t_1 \text{ mgt } t_3)$
- ▶ There are terms  $t \neq t'$  such that  $(t \text{ mgt } t')$  and  $(t' \text{ mgt } t)$ . Example:

$$\text{succ}(x) \quad \text{succ}(y)$$

- ▶ mgt extends to substitutions pointwise:

$$\rho \text{ mgt } \rho' \text{ if } \exists \rho''. \forall x. \rho'(x) = \rho''(\rho(x))$$



# The Unification Problem



The **unification problem** (syntactic, first-order) can be stated as follows:

Given a set of *potential* equalities

$$\mathcal{G} = \{ l_1 \stackrel{?}{=} r_1, \dots, l_n \stackrel{?}{=} r_n \}$$

where  $l_1, \dots, l_n, r_1, \dots, r_n \in T_{\Sigma, X}$ .

Can we find a  $\rho$  such that  $\forall i \in [1, n]. \rho(l_i) = \rho(r_i)$  ?

# The Unification Problem



The **unification problem** (syntactic, first-order) can be stated as follows:

Given a set of *potential* equalities

$$\mathcal{G} = \{ l_1 \stackrel{?}{=} r_1, \dots, l_n \stackrel{?}{=} r_n \}$$

where  $l_1, \dots, l_n, r_1, \dots, r_n \in T_{\Sigma, X}$ .

Can we find a  $\rho$  such that  $\forall i \in [1, n]. \rho(l_i) = \rho(r_i)$  ?

The set of **solutions** of  $\mathcal{G}$ :  $\text{sols}(\mathcal{G}) \triangleq \{ \rho \mid \forall i \in [1, n]. \rho(l_i) = \rho(r_i) \}$ .

# The Unification Problem



The **unification problem** (syntactic, first-order) can be stated as follows:

Given a set of *potential* equalities

$$\mathcal{G} = \{ l_1 \stackrel{?}{=} r_1, \dots, l_n \stackrel{?}{=} r_n \}$$

where  $l_1, \dots, l_n, r_1, \dots, r_n \in T_{\Sigma, X}$ .

Can we find a  $\rho$  such that  $\forall i \in [1, n]. \rho(l_i) = \rho(r_i)$  ?

The set of **solutions** of  $\mathcal{G}$ :  $\text{sols}(\mathcal{G}) \triangleq \{ \rho \mid \forall i \in [1, n]. \rho(l_i) = \rho(r_i) \}$ .

Another problem is finding the **most general** unifier for  $\mathcal{G}$ :

can we find a  $\rho \in \text{sols}(\mathcal{G})$  such that  $\rho \text{ mgt } \rho'$  for every  $\rho' \in \text{sols}(\mathcal{G})$ ?

# Unification Algorithm



**Idea** We iteratively reduce the set  $\mathcal{G}$  by solution-preserving transformations until

- either a solution is found or
- we can prove there is no solution.

**Note** A solution may no exist and even if exists it may not be unique.

# Algorithm: Termination Criteria



- ▶ We say  $\mathcal{G}$  and  $\mathcal{G}'$  are **equivalent** if  $\text{sols}(\mathcal{G}) = \text{sols}(\mathcal{G}')$ .
- ▶ Given  $\mathcal{G} = \{ l_1 \stackrel{?}{=} r_1, \dots, l_n \stackrel{?}{=} r_n \}$ , the algorithm terminates successfully when we reach:

$$\mathcal{G}' = \{ x_1 \stackrel{?}{=} t_1, \dots, x_k \stackrel{?}{=} t_k \} \quad \begin{array}{l} \text{such} \\ \text{that} \end{array} \quad \begin{array}{l} - \mathcal{G}' \text{ is equivalent to } \mathcal{G} \\ - \underbrace{\{x_1, \dots, x_k\}}_{\text{all different}} \cap \bigcup_{i=1}^k \text{vars}(t_i) = \emptyset \end{array}$$

- ▶ Any such  $\mathcal{G}'$  determines a solution  $[x_1 = t_1, \dots, x_k = t_k]$ .

# Algorithm: Notation



Suppose  $\mathcal{G} = \{ l_1 \stackrel{?}{=} r_1, \dots, l_n \stackrel{?}{=} r_n \}$ . We define:

$$\text{vars}(\mathcal{G}) \triangleq \bigcup_{i=1}^n (\text{vars}(l_i) \cup \text{vars}(r_i))$$

$$\mathcal{G}\rho \triangleq \{ l_1\rho \stackrel{?}{=} r_1\rho, \dots, l_n\rho \stackrel{?}{=} r_n\rho \}$$

# Unification Algorithm



## delete

$$\mathcal{G} \cup \{t \stackrel{?}{=} t\} \\ \text{becomes} \\ \mathcal{G}$$

## eliminate

$$\mathcal{G} \cup \{x \stackrel{?}{=} t\} \\ \text{becomes} \\ \mathcal{G}[x = t] \cup \{x \stackrel{?}{=} t\} \quad \text{if } x \in \text{vars}(\mathcal{G}) \setminus \text{vars}(t)$$

## swap

$$\mathcal{G} \cup \{f(t_1, \dots, t_m) \stackrel{?}{=} x\} \\ \text{becomes} \\ \mathcal{G} \cup \{x \stackrel{?}{=} f(t_1, \dots, t_m)\}$$

## decompose

$$\mathcal{G} \cup \{f(t_1, \dots, t_m) \stackrel{?}{=} f(u_1, \dots, u_m)\} \\ \text{becomes} \\ \mathcal{G} \cup \{t_1 \stackrel{?}{=} u_1, \dots, t_m \stackrel{?}{=} u_m\}$$

## occur-check

$$\mathcal{G} \cup \{x \stackrel{?}{=} f(t_1, \dots, t_m)\} \\ \text{fails if } x \in \text{vars}(f(t_1, \dots, t_m))$$

## conflict

$$\mathcal{G} \cup \{f(t_1, \dots, t_m) \stackrel{?}{=} g(u_1, \dots, u_h)\} \\ \text{fails if } f \neq g \text{ or } m \neq h$$

# Unification Algorithm: Example



$$\{\text{plus}(\text{succ}(x), x) \stackrel{?}{=} \text{plus}(y, 0)\}$$



# Unification Algorithm: Example



$$\{\text{plus}(\text{succ}(x), x) \stackrel{?}{=} \text{plus}(y, 0)\}$$

**decompose**

$$\{\text{succ}(x) \stackrel{?}{=} y, x \stackrel{?}{=} 0\}$$

# Unification Algorithm: Example



$$\{\text{plus}(\text{succ}(x), x) \stackrel{?}{=} \text{plus}(y, 0)\}$$

**decompose**

$$\{\text{succ}(x) \stackrel{?}{=} y, x \stackrel{?}{=} 0\}$$

**eliminate**

$$\{\text{succ}(0) \stackrel{?}{=} y, x \stackrel{?}{=} 0\}$$

# Unification Algorithm: Example



$$\{\text{plus}(\text{succ}(x), x) \stackrel{?}{=} \text{plus}(y, 0)\}$$

**decompose**

$$\{\text{succ}(x) \stackrel{?}{=} y, x \stackrel{?}{=} 0\}$$

**eliminate**

$$\{\text{succ}(0) \stackrel{?}{=} y, x \stackrel{?}{=} 0\}$$

**swap**

$$\{y \stackrel{?}{=} \text{succ}(0), x \stackrel{?}{=} 0\}$$

# Unification Algorithm: Example



$$\{\text{plus}(\text{succ}(x), x) \stackrel{?}{=} \text{plus}(y, 0)\}$$

**decompose**

$$\{\text{succ}(x) \stackrel{?}{=} y, x \stackrel{?}{=} 0\}$$

**eliminate**

$$\{\text{succ}(0) \stackrel{?}{=} y, x \stackrel{?}{=} 0\}$$

**swap**

$$\{y \stackrel{?}{=} \text{succ}(0), x \stackrel{?}{=} 0\}$$

✓ **success:**  $\rho = [y = \text{succ}(0), x = 0]$

# Unification Algorithm: Example



$$\{\text{plus}(0, x) \stackrel{?}{=} \text{succ}(y)\}$$

# Unification Algorithm: Example



$$\{\text{plus}(0, x) \stackrel{?}{=} \text{succ}(y)\}$$

**conflict:**  $\text{plus} \neq \text{succ}$

# Unification Algorithm: Example



$$\{\text{plus}(0, x) \stackrel{?}{=} \text{succ}(y)\}$$

**conflict:**  $\text{plus} \neq \text{succ}$

**✗ fail**

# Unification: Another Example



$$\{\text{succ}(x) \stackrel{?}{=} y, \text{succ}(y) \stackrel{?}{=} x\}$$



# Unification: Another Example



$$\{\text{succ}(x) \stackrel{?}{=} y, \text{succ}(y) \stackrel{?}{=} x\}$$

**swap**

$$\{\text{succ}(x) \stackrel{?}{=} y, x \stackrel{?}{=} \text{succ}(y)\}$$

# Unification: Another Example



$$\{\text{succ}(x) \stackrel{?}{=} y, \text{succ}(y) \stackrel{?}{=} x\}$$

**swap**

$$\{\text{succ}(x) \stackrel{?}{=} y, x \stackrel{?}{=} \text{succ}(y)\}$$

**eliminate**

$$\{\text{succ}(\text{succ}(y)) \stackrel{?}{=} y, x \stackrel{?}{=} \text{succ}(y)\}$$

# Unification: Another Example



$$\{\text{succ}(x) \stackrel{?}{=} y, \text{succ}(y) \stackrel{?}{=} x\}$$

**swap**

$$\{\text{succ}(x) \stackrel{?}{=} y, x \stackrel{?}{=} \text{succ}(y)\}$$

**eliminate**

$$\{\text{succ}(\text{succ}(y)) \stackrel{?}{=} y, x \stackrel{?}{=} \text{succ}(y)\}$$

**swap**

$$\{y \stackrel{?}{=} \text{succ}(\text{succ}(y)), x \stackrel{?}{=} \text{succ}(y)\}$$

# Unification: Another Example



$$\{\text{succ}(x) \stackrel{?}{=} y, \text{succ}(y) \stackrel{?}{=} x\}$$

**swap**

$$\{\text{succ}(x) \stackrel{?}{=} y, x \stackrel{?}{=} \text{succ}(y)\}$$

**eliminate**

$$\{\text{succ}(\text{succ}(y)) \stackrel{?}{=} y, x \stackrel{?}{=} \text{succ}(y)\}$$

**swap**

$$\{y \stackrel{?}{=} \text{succ}(\text{succ}(y)), x \stackrel{?}{=} \text{succ}(y)\}$$

**occur-check:**  $y \in \text{vars}(\text{succ}(\text{succ}(y)))$

**✗ fail**

# Exercise



$$\{\text{plus}(x, \text{succ}(x)) \stackrel{?}{=} \text{plus}(0, y), \text{plus}(y, z) \stackrel{?}{=} \text{plus}(z, w)\}$$

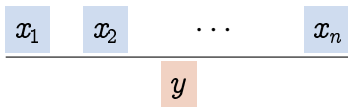


# Inference Rules

# Inference Rules



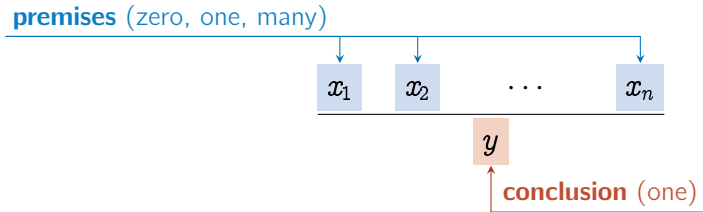
An inference rule:



# Inference Rules



An inference rule:

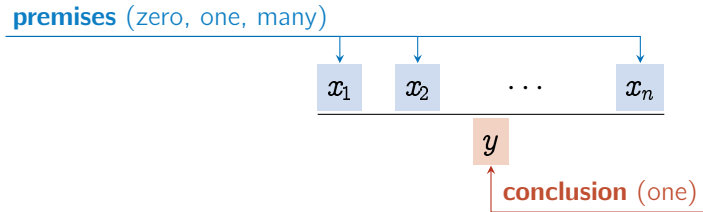




# Inference Rules



An inference rule:

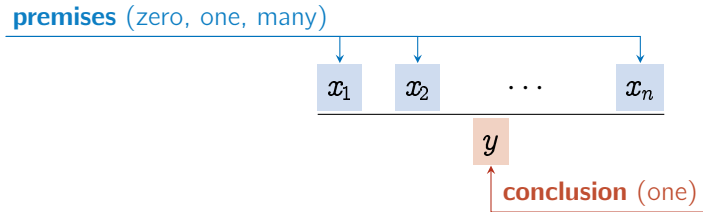


- **Intuition:** If the premises are valid, then the conclusion is also valid.
- In an **axiom** there are no premises: the conclusion is always valid (it is a fact)
- Above,  $x_1, x_2, \dots, y$  are formulas. Any variable in them is universally quantified (implicitly).

# Inference Rules



An inference rule:



- **Intuition**: If the premises are valid, then the conclusion is also valid.
- In an **axiom** there are no premises: the conclusion is always valid (it is a fact)
- Above,  $x_1, x_2, \dots, y$  are formulas. Any variable in them is universally quantified (implicitly).
- A **rule instance** is obtained by applying some substitution  $\rho$  to  $x_1, x_2, \dots, y$ .

# Rule Instances



$$\begin{array}{c} (prod) \\ \frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \otimes E_1 \longrightarrow n} \quad n = n_0 \cdot n_1 \end{array}$$

Two **instances** of *(prod)*:

$$\begin{array}{c} (prod) \\ \frac{1 \longrightarrow 1 \quad 1 \oplus 2 \longrightarrow 3}{1 \otimes (1 \oplus 2) \longrightarrow 3} \quad 3 = 1 \cdot 3 \end{array}$$

$$\begin{array}{c} (prod) \\ \frac{1 \longrightarrow 3 \quad 1 \oplus 2 \longrightarrow 5}{1 \otimes (1 \oplus 2) \longrightarrow 3} \quad 15 = 3 \cdot 5 \end{array}$$

Not all instances are valid!

# Rule Instances



$$\begin{array}{c} (prod) \\ \frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \otimes E_1 \longrightarrow n} \quad n = n_0 \cdot n_1 \end{array}$$

Another instance:

$$\begin{array}{c} (prod) \\ \frac{E \otimes 2 \longrightarrow k \quad E \oplus 1 \longrightarrow 3}{(E \otimes 2) \otimes (E \oplus 1) \longrightarrow 3k} \end{array}$$

# Rule Instances



$$\begin{array}{c} (prod) \\ \frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \otimes E_1 \longrightarrow n} \quad n = n_0 \cdot n_1 \end{array}$$

Another instance:

variables can be shared

$$\begin{array}{c} (prod) \\ \frac{E \otimes 2 \longrightarrow k \quad E \oplus 1 \longrightarrow 3}{(E \otimes 2) \otimes (E \oplus 1) \longrightarrow 3k} \end{array}$$

variables can be shared



A **logical system** is a set of axioms and inference rules:

$$R = \left\{ \frac{}{z}, \frac{x_1 \quad \cdots \quad x_n}{y}, \dots \right\}$$

If an inference rule contains some variables, we assume all its instances are in  $R$

# Derivations



Given a logical system  $R$ , a **derivation in  $R$** , is written

$$d \Vdash_R y$$

where

- ▶ either  $d = \left( \frac{\quad}{y} \right)$  is an axiom of  $R$ ;
- ▶ or  $d = \left( \frac{d_1 \ \cdots \ d_n}{y} \right)$  for some derivations  $d_1 \Vdash_R x_1, \dots, d_n \Vdash_R x_n$   
such that  $\left( \frac{x_1 \ \cdots \ x_n}{y} \right)$  is an inference rule of  $R$ .

Put differently: a derivation is a **proof tree** whose leaves are axioms.

# Example



$$R = \left\{ \frac{}{N \longrightarrow n}, \frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \oplus E_1 \longrightarrow n_0 + n_1}, \frac{E_0 \longrightarrow n_0 \quad E_1 \longrightarrow n_1}{E_0 \otimes E_1 \longrightarrow n_0 \cdot n_1} \right\}$$

$$d = \frac{\frac{\overline{1 \longrightarrow 1} \quad \overline{2 \longrightarrow 2}}{(1 \oplus 2) \longrightarrow 3} \quad \frac{\overline{3 \longrightarrow 3} \quad \overline{4 \longrightarrow 4}}{(3 \oplus 4) \longrightarrow 7}}{(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21}$$

Hence

$$d \Vdash_R (1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21$$





- Given a logical system  $R$ , a **theorem of  $R$**  is written

$$\Vdash_R y$$

That is,  $y$  is a formula for which we can find a derivation  $d$  in  $R$ .

- The set of all theorems of  $R$  is denoted by  $I_R$ :

$$I_R \triangleq \{y \mid \Vdash_R y\}$$

# An Inline Notation for Derivations



$$d = \frac{\frac{\frac{1 \longrightarrow 1}{\quad} \quad \frac{2 \longrightarrow 2}{\quad}}{(1 \oplus 2) \longrightarrow 3} \quad \frac{\frac{3 \longrightarrow 3}{\quad} \quad \frac{4 \longrightarrow 4}{\quad}}{(3 \oplus 4) \longrightarrow 7}}{(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21}$$

$$\begin{aligned} (1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21 &\nwarrow (1 \oplus 2) \longrightarrow 3, (3 \oplus 4) \longrightarrow 7 \\ &\nwarrow 1 \longrightarrow 1, 2 \longrightarrow 2, (3 \oplus 4) \longrightarrow 7 \\ &\nwarrow 2 \longrightarrow 2, (3 \oplus 4) \longrightarrow 7 \\ &\nwarrow (3 \oplus 4) \longrightarrow 7 \\ &\nwarrow 3 \longrightarrow 3, 4 \longrightarrow 4 \\ &\nwarrow 4 \longrightarrow 4 \\ &\nwarrow \square \quad [\text{the empty goal: nothing left to prove}] \end{aligned}$$

# Backtracking



A goal oriented derivation (depth-first):

$$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21$$

$$\swarrow (1 \oplus 2) \longrightarrow 7, (3 \oplus 4) \longrightarrow 3$$

$$\swarrow 1 \longrightarrow 1, 2 \longrightarrow 6, (3 \oplus 4) \longrightarrow 3$$

$$\swarrow 2 \longrightarrow 6, (3 \oplus 4) \longrightarrow 3$$

fail! need to backtrack to the last choice and retry

$$\swarrow 1 \longrightarrow 2, 2 \longrightarrow 5, (3 \oplus 4) \longrightarrow 3$$

fail! need to backtrack to the last choice and retry

...



# Logic Programming



- ▶ PROLOG ('PROgrammation en LOGique') is a simple, yet powerful declarative programming language, based on first-order predicate logic
- ▶ Key ideas:

Algorithm = Logic + Control

What (problem description)	How (steps to reach a solution)
-------------------------------	------------------------------------

Horn clauses	Resolution
--------------	------------

Database	Interpreter
----------	-------------



► Base sets:

$$X = \{x, y, \dots\}$$

a set of **variables**

$$\Sigma = \{\Sigma_n\}_{n \in \mathbb{N}}$$

a signature of **function symbols**  $c, f, g, \dots$

$$\Pi = \{\Pi_n\}_{n \in \mathbb{N}}$$

a signature of **predicate symbols**  $p, q, \dots$

► Atomic formula

$$a = p(t_1, \dots, t_n)$$

where  $p \in \Pi_n$  and  $t_1, \dots, t_n \in T_{\Sigma, X}$

► A formula

$$a_1, \dots, a_n$$

is a possibly empty conjunction of atomic formulas

# Logic Programs



A logic program serves to answer the question: given a formula  $g$  that we want to prove (a **goal**), what are the **valid instances** of  $g$ ?

# Logic Programs



A logic program serves to answer the question: given a formula  $g$  that we want to prove (a **goal**), what are the **valid instances** of  $g$ ? Example:

```
% Rules
grandparent(X, Y) :- parent(X, Z), parent(Z, Y).

% Facts
parent(jan, merel).
parent(merel, sandra).

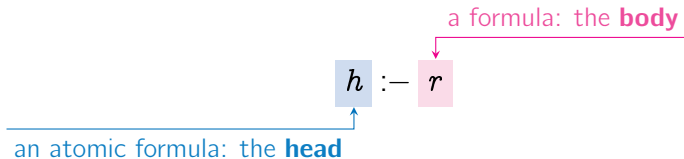
:- initialization(main).
main :-
    grandparent(X, sandra),
    write('Sandras grandparent: '), write(X), nl, halt.
```



# Logic Programs, Formally



- A Horn clause:



- Having  $a \text{ :- } a_1, \dots, a_n$  is analogous to  $\frac{a_1 \quad \dots \quad a_n}{a}$ .
- A **logic program** is a set / list of Horn clauses:

$$L = \left\{ \begin{array}{c} \dots \\ h \text{ :- } r \\ \dots \end{array} \right\}$$

# SLD Resolution



**Idea** Iteratively reduce the initial goal  $g$  by applying one of the Horn clauses in  $L$  to one of the atomic formulas in the goal

Each application

- ▶ computes a most general unifier (mgu)
- ▶ replaces the selected formula with the body of the selected clause and
- ▶ applies the mgu to the new goal

$$\begin{array}{l} ?-g \quad \swarrow_{\sigma_1} \quad g_1 \\ \quad \quad \swarrow_{\sigma_2} \quad g_2 \\ \quad \quad \swarrow_{\sigma_3} \quad \dots \\ \quad \quad \swarrow_{\sigma_m} \quad \square \end{array}$$

Then  $g\sigma_1\sigma_2\cdots\sigma_m$  is a theorem.



Assume given:

$$?-a_1, \dots, a_i, \dots, a_k \qquad L = \{\dots, h :- b_1, \dots, b_n, \dots\}$$

Repeat until no goal is left:

1. Select a clause of the goal  $a_i$  (e.g. the first)
2. Select a Horn clause  $h :- b_1, \dots, b_n$  from  $L$  whose heads unifies with  $a_i$
3. Let  $\sigma$  be a most general unifier ( $a_i\sigma = h\sigma$ )
4. Replace  $a_i$  with  $b_1, \dots, b_n$
5. Apply the substitution  $\sigma$  to the revised goal  $(a_1, \dots, b_1, \dots, b_n, \dots, a_k)\sigma$ .  
This ensures that  $\sigma$  is propagated uniformly, as goals may share variables.

# SLD Resolution: Variables Matter



**Note** The same clause can be reused many times: each time its variables must be renamed (before unification) with **fresh identifiers**, to avoid clashes.

# SLD Resolution: Variables Matter



**Note** The same clause can be reused many times: each time its variables must be renamed (before unification) with **fresh identifiers**, to avoid clashes.

Repeat until no goal is left:

1. Select a clause of the goal  $a_i$  (e.g. the first)
2. Select a Horn clause  $h :- b_1, \dots, b_n$  from  $M$  whose heads unifies with  $a_i$
3. Let  $\rho : X \rightarrow X$  be a renaming of  $\text{vars}(h :- b_1, \dots, b_n)$  to fresh variables

The renamed clause  $(h :- b_1, \dots, b_n)\rho$  is a **variant** of the original one

4. Let  $\sigma$  be a most general unifier ( $a_i\sigma = (h\rho)\sigma$ )
5. Replace  $a_i$  with  $(b_1, \dots, b_n)\rho$
6. Apply  $\sigma$  to the revised goal  $(a_1, \dots, (b_1, \dots, b_n)\rho, \dots, a_k)\sigma$ .  
This ensures that  $\sigma$  is propagated uniformly, as goals may share variables.

# SLD Resolution: Substitutions



In the computed substitution, we are only interested in the variables that appear in the goal. We therefore define a 'partial substitution':

Given  $\sigma : X \rightarrow T_{\Sigma, X}$  and  $Y \subseteq X$ , we define:

$$\sigma|_Y \triangleq \begin{cases} \sigma(x) & \text{if } x \in Y \\ x & \text{otherwise} \end{cases}$$

In resolution we then use  $\sigma$  and  $\hat{\sigma}$ :

$$a_1, \dots, a_i, \dots, a_k \xleftarrow{\hat{\sigma}} (a_1, \dots, b_1, \dots, a_n, \dots, a_k) \sigma$$

where  $\hat{\sigma} \triangleq \sigma|_Y$  with  $Y = \text{vars}(a_1, \dots, a_k)$ .

## Example: Summation (1/4)



- Define:

$$\begin{aligned}\Sigma_0 &= \{0, \dots\} & \Pi_3 &= \{\text{sum}, \dots\} \\ \Sigma_1 &= \{s, \dots\}\end{aligned}$$

- Sum as a predicate: `sum( $x, y, z$ )` means ' $x + y = z$ '.
- The set  $L$ :

```
sum(0, y, y).  
sum(s(x), y, s(z)) :- sum(x, y, z).
```

- Let's target the goal: `?- sum(s(s(0)), s(s(0)), n)`. (That is: ' $2 + 2 = ?$ ') )

## Example: Summation (2/4)



Given our goal  $\text{sum}(s(s(0)), s(s(0)), n)$  and the set  $L$ :

$$\blacktriangleright \{ \text{sum}(s(s(0)), s(s(0)), n) \stackrel{?}{=} \text{sum}(0, y', y') \}$$

**fails!**



## Example: Summation (2/4)



Given our goal  $\text{sum}(s(s(0)), s(s(0)), n)$  and the set  $L$ :

- ▶  $\{\text{sum}(s(s(0)), s(s(0)), n) \stackrel{?}{=} \text{sum}(0, y', y')\}$
- ▶  $\{\text{sum}(s(s(0)), s(s(0)), n) \stackrel{?}{=} \text{sum}(s(x_1), y_1, s(z_1))\}$

**fails!**

**succeeds!**

We have

$$\sigma_1 = [x_1 = s(0), y_1 = s(s(0)), n = s(z_1)]$$

$$\widehat{\sigma}_1 = [n = s(z_1)]$$

## Example: Summation (2/4)



Given our goal  $\text{sum}(s(s(0)), s(s(0)), n)$  and the set  $L$ :

- ▶  $\{\text{sum}(s(s(0)), s(s(0)), n) \stackrel{?}{=} \text{sum}(0, y', y')\}$
- ▶  $\{\text{sum}(s(s(0)), s(s(0)), n) \stackrel{?}{=} \text{sum}(s(x_1), y_1, s(z_1))\}$

**fails!**

**succeeds!**

We have

$$\sigma_1 = [x_1 = s(0), y_1 = s(s(0)), n = s(z_1)]$$

$$\widehat{\sigma}_1 = [n = s(z_1)]$$

Therefore:

$$\begin{aligned} \text{sum}(s(s(0)), s(s(0)), n) &\prec_{\widehat{\sigma}_1} (\text{sum}(x_1, y_1, z_1))\sigma_1 \\ &= \text{sum}(s(0), s(s(0)), z_1) \end{aligned}$$

## Example: Summation (2/4)



Given our goal  $\text{sum}(s(s(0)), s(s(0)), n)$  and the set  $L$ :

$$\blacktriangleright \{\text{sum}(s(s(0)), s(s(0)), n) \stackrel{?}{=} \text{sum}(0, y', y')\}$$

**fails!**

$$\blacktriangleright \{\text{sum}(s(s(0)), s(s(0)), n) \stackrel{?}{=} \text{sum}(s(x_1), y_1, s(z_1))\}$$

**succeeds!**

We have

$$\sigma_1 = [x_1 = s(0), y_1 = s(s(0)), n = s(z_1)]$$

$$\widehat{\sigma}_1 = [n = s(z_1)]$$

Therefore:

$$\begin{aligned} \text{sum}(s(s(0)), s(s(0)), n) &\searrow_{\widehat{\sigma}_1} (\text{sum}(x_1, y_1, z_1))\sigma_1 \\ &= \text{sum}(s(0), s(s(0)), z_1) \end{aligned}$$

Note: we write  $\searrow_{\widehat{\sigma}_1}$  because  $\widehat{\sigma}_1$  gives the least condition for the derivation.

## Example: Summation (3/4)



Given our new goal  $\text{sum}(s(0), s(s(0)), z_1)$  and the set  $L$ :

$$\blacktriangleright \{\text{sum}(s(0), s(s(0)), z_1) \stackrel{?}{=} \text{sum}(0, y', y')\}$$

**fails!**

## Example: Summation (3/4)



Given our new goal  $\text{sum}(s(0), s(s(0)), z_1)$  and the set  $L$ :

►  $\{\text{sum}(s(0), s(s(0)), z_1) \stackrel{?}{=} \text{sum}(0, y', y')\}$

**fails!**

►  $\{\text{sum}(s(0), s(s(0)), z_1) \stackrel{?}{=} \text{sum}(s(x_2), y_2, s(z_2))\}$

**succeeds!**

We have

$$\sigma_2 = [x_2 = 0, y_2 = s(s(0)), n = s(z_2)]$$

$$\widehat{\sigma}_2 = [z_1 = s(z_2)]$$

## Example: Summation (3/4)



Given our new goal  $\text{sum}(s(0), s(s(0)), z_1)$  and the set  $L$ :

- ▶  $\{\text{sum}(s(0), s(s(0)), z_1) \stackrel{?}{=} \text{sum}(0, y', y')\}$
- ▶  $\{\text{sum}(s(0), s(s(0)), z_1) \stackrel{?}{=} \text{sum}(s(x_2), y_2, s(z_2))\}$

**fails!**

**succeeds!**

We have

$$\sigma_2 = [x_2 = 0, y_2 = s(s(0)), n = s(z_2)]$$

$$\widehat{\sigma}_2 = [z_1 = s(z_2)]$$

Therefore:

$$\begin{aligned} \text{sum}(s(s(0)), s(s(0)), n) &\nwarrow_{\widehat{\sigma}_1} \text{sum}(s(0), s(s(0)), z_1) \\ &\nwarrow_{\widehat{\sigma}_2} (\text{sum}(x_2, y_2, z_2))\sigma_2 \\ &= \text{sum}(0, s(s(0)), z_2) \end{aligned}$$

## Example: Summation (4/4)



Given our new goal `sum(0, s(s(0)), z2)` and the set  $L$ :

$$\blacktriangleright \{\text{sum}(0, s(s(0)), z_2) \stackrel{?}{=} \text{sum}(0, y_3, y_3)\}$$

**succeeds!**

We have

$$\sigma_3 = [y_3 = s(s(0)), z_2 = s(s(0))]$$

$$\widehat{\sigma}_3 = [z_2 = s(s(0))]$$

## Example: Summation (4/4)



Given our new goal  $\text{sum}(0, s(s(0)), z_2)$  and the set  $L$ :

►  $\{\text{sum}(0, s(s(0)), z_2) \stackrel{?}{=} \text{sum}(0, y_3, y_3)\}$

succeeds!

We have

$$\sigma_3 = [y_3 = s(s(0)), z_2 = s(s(0))]$$

$$\widehat{\sigma}_3 = [z_2 = s(s(0))]$$

Therefore:

$$\begin{aligned} \text{sum}(s(s(0)), s(s(0)), n) &\nwarrow_{\widehat{\sigma}_1} \text{sum}(s(0), s(s(0)), z_1) \\ &\nwarrow_{\widehat{\sigma}_2} \text{sum}(0, s(s(0)), z_2) \\ &\nwarrow_{\widehat{\sigma}_3} \square \end{aligned}$$



## Example: Summation (4/4)



Given our new goal  $\text{sum}(0, s(s(0)), z_2)$  and the set  $L$ :

- $\{\text{sum}(0, s(s(0)), z_2) \stackrel{?}{=} \text{sum}(0, y_3, y_3)\}$

succeeds!

We have

$$\sigma_3 = [y_3 = s(s(0)), z_2 = s(s(0))]$$

$$\widehat{\sigma}_3 = [z_2 = s(s(0))]$$

Therefore:

$$\begin{aligned} \text{sum}(s(s(0)), s(s(0)), n) &\nwarrow_{\widehat{\sigma}_1} \text{sum}(s(0), s(s(0)), z_1) \\ &\nwarrow_{\widehat{\sigma}_2} \text{sum}(0, s(s(0)), z_2) \\ &\nwarrow_{\widehat{\sigma}_3} \square \end{aligned}$$

- Recall:  $\widehat{\sigma}_1 = [n = s(z_1)]$ ,  $\widehat{\sigma}_2 = [z_1 = s(z_2)]$ ,  $\widehat{\sigma}_3 = [z_2 = s(s(0))]$ .  
We conclude:  $\widehat{\sigma}_1 \cdot \widehat{\sigma}_2 \cdot \widehat{\sigma}_3 = [n = s(s(s(s(0))))]$ , as desired.



The End