



# Critical Agents Supporting Interactive Theorem Proving

Christoph Benz Müller

Joint work with Volker Sorge

`chris|sorge@ags.uni-sb.de`

The  $\Omega$ MEGA Group

`http://www.ags.uni-sb.de/~omega`

FB Informatik, Universität des Saarlandes, Saarbrücken, Germany

EPIA99, Evora, September 1999

# Overview

- Motivation
- A **layered blackboard mechanism** for suggesting promising proof steps
- Guiding the search:
  - A **resource adapted** approach
  - A **resource adaptive** approach
- Applications & Conclusions



# Tactical Theorem Proving in $\Omega$ MEGA

- Natural deduction (ND) rules
- **Tactics** are macro steps (expandable in a sequence of ND rule applications)
- **Proof methods** are tactics enriched by a declaratively specified pre- and post-conditions
- Other commands:
  - call **external systems** like OTTER or LEO
  - **verbalise** a proof, **verify** a proof, ...
- This talk addresses: **rules, tactics, methods, and external systems**

$$\frac{\forall x. A}{[t/x]A} \quad \forall_E(t) \quad \frac{\Phi[x]_p \quad x = y}{\Phi'[y]_p} =_{Subst}(p)$$



# State of the Art & Research Goals

- Drawback of respective suggestion mechanisms in state of the art theorem proving environments (e.g.  $\Omega$ MEGA, TPS, HOL):  
pure **passive character**
- A **good** suggestion mechanisms instead should
  - **autonomously** run in the background of the system
  - **dynamically** present the most promising suggestions to the user
  - **react** and **respond** to special queries by the user
  - **optimally** employ the available resources
  - ...
- Research goal: **active** and **intelligent** proof assistant



# Example ND-Proof

Problem:  $p_{o \rightarrow o} (a_o \wedge b_o) \Rightarrow (p (b \wedge a))$

ND-proof:

$C \quad () \quad \vdash \quad (p (a \wedge b)) \Rightarrow (p (b \wedge a)) \quad \text{OPEN}$



# Example ND-Proof

Problem:  $p_{o \rightarrow o} (a_o \wedge b_o) \Rightarrow (p (b \wedge a))$

ND-proof:

$L_1 (L_1) \vdash (p (a \wedge b))$  Hyp

$L_2 (L_1) \vdash (p (b \wedge a))$  OPEN

$C () \vdash (p (a \wedge b)) \Rightarrow (p (b \wedge a)) \Rightarrow_I: (L_2)$



# Example ND-Proof

Problem:  $p_{o \rightarrow o} (a_o \wedge b_o) \Rightarrow (p (b \wedge a))$

ND-proof:

$L_1 (L_1) \vdash (p (a \wedge b))$  Hyp

$L_3 (L_1) \vdash (b \wedge a) = (a \wedge b)$  OPEN

$L_2 (L_1) \vdash (p (b \wedge a))$   $=_{\text{subst}}: (\langle 1 \rangle)(L_1 L_3)$

$C () \vdash (p (a \wedge b)) \Rightarrow (p (b \wedge a))$   $\Rightarrow_I: (L_2)$



# Example ND-Proof

Problem:  $p_{o \rightarrow o} (a_o \wedge b_o) \Rightarrow (p (b \wedge a))$

ND-proof:

$L_1 (L_1) \vdash (p (a \wedge b))$

$L_4 (L_1) \vdash (b \wedge a) \Leftrightarrow (a \wedge b)$

$L_3 (L_1) \vdash (b \wedge a) = (a \wedge b)$

$L_2 (L_1) \vdash (p (b \wedge a))$

$C () \vdash (p (a \wedge b)) \Rightarrow (p (b \wedge a))$

Hyp

OPEN

$\Leftrightarrow 2 = : (L_4)$

$=_{\text{subst}} : (\langle 1 \rangle)(L_1 L_3)$

$\Rightarrow_I : (L_2)$



# Example ND-Proof

Problem:  $p_{o \rightarrow o} (a_o \wedge b_o) \Rightarrow (p (b \wedge a))$

ND-proof:

$L_1 (L_1) \vdash (p (a \wedge b))$

$L_4 (L_1) \vdash (b \wedge a) \Leftrightarrow (a \wedge b)$

$L_3 (L_1) \vdash (b \wedge a) = (a \wedge b)$

$L_2 (L_1) \vdash (p (b \wedge a))$

$C () \vdash (p (a \wedge b)) \Rightarrow (p (b \wedge a))$

Hyp

OTTER

$\Leftrightarrow 2 = : (L_4)$

$=_{\text{subst}} : (\langle 1 \rangle)(L_1 L_3)$

$\Rightarrow_I : (L_2)$



# Partial Argument Instantiations

rule, tactic, method

invoking command

$$\frac{\Phi[x]_p \quad x = y}{\Phi'[y]_p} =_{Subst}(p) \quad \longrightarrow \quad \frac{prem \quad eq}{conc} =_{Subst}(pos)$$

$$\frac{\forall x. A}{[t/x]A} \forall_E(t) \quad \longrightarrow \quad \frac{prem}{conc} \text{ForallE}(term)$$

- Representing suggestions by Partial Argument Instantiations (**PAI**)

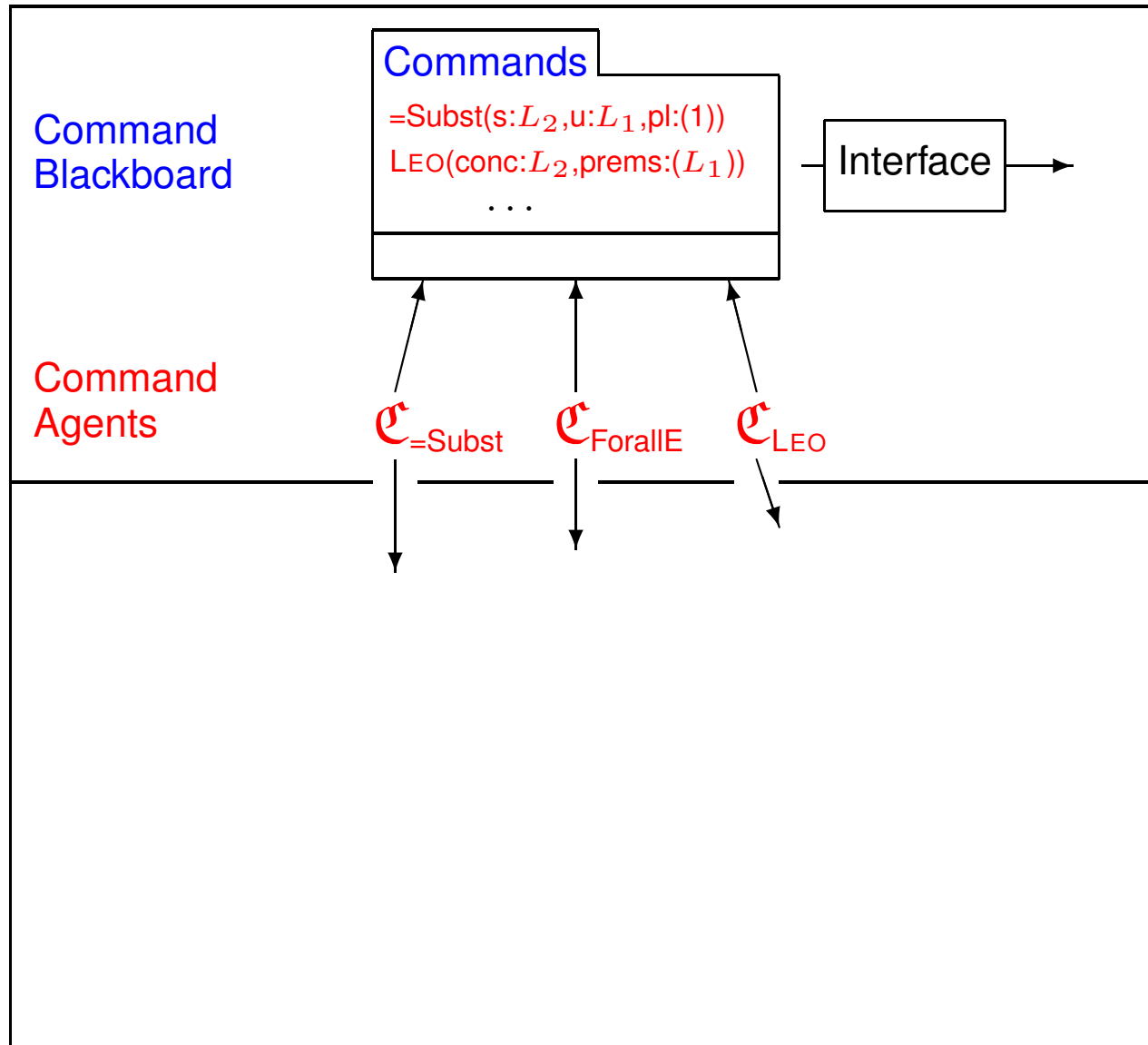
$$=_{Subst}(prem : L_1, conc : L_2, eq : L_3, pos : (1))$$

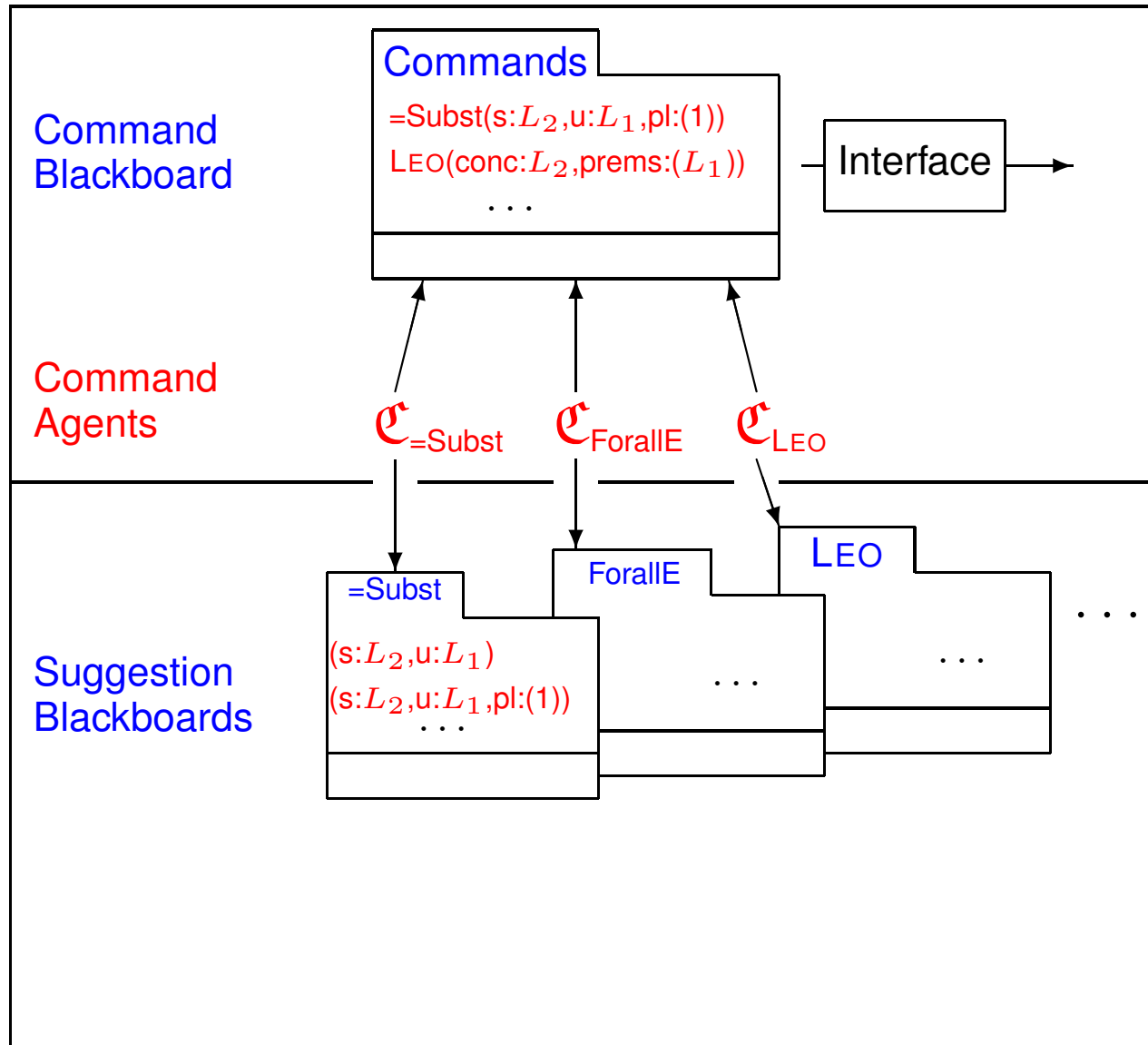
$$=_{Subst}(prem : L_1, eq : L_3, pos : (1))$$

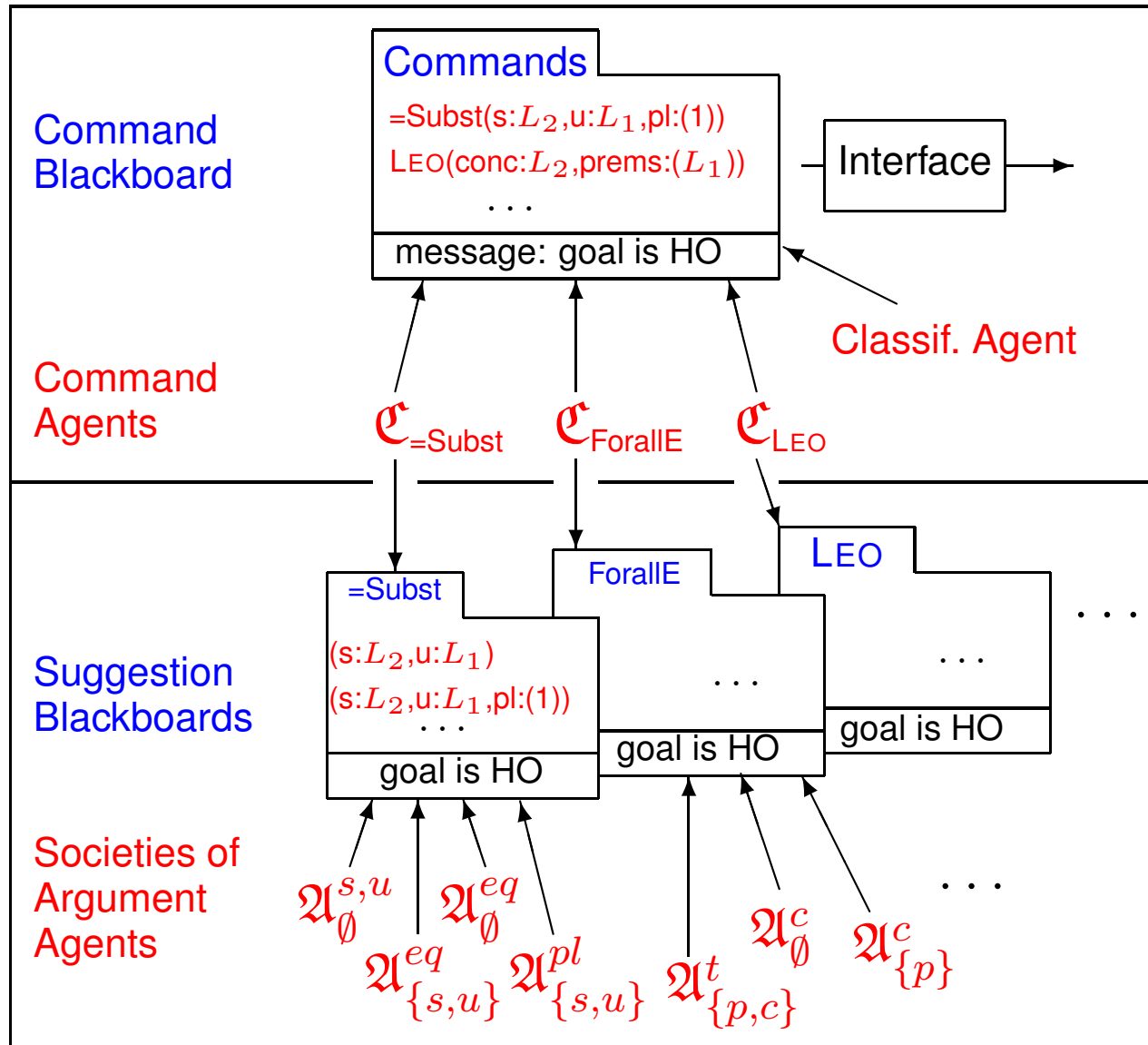
$$\text{ForallE}(prem : L_4)2$$

- Task of the suggestion mechanism: **compute PAI's**









# Argument Agents

ForallE-Agents:

$$\frac{\forall x. A}{[t/x]A} \forall_E(t) \longrightarrow \frac{prem}{conc} \text{ForallE}(term)$$

$$\mathcal{A}_{\emptyset}^{prem} = \{ \text{find universally quantified line } prem \}$$

$$\mathcal{A}_{\{prem\}}^{conc} = \{ \text{find 'matching' open line } conc \text{ for } prem \}$$

$$\mathcal{A}_{\{prem, conc\}}^{term} = \{ \text{compute } \text{mgm}(\text{scope of } prem, conc) \}$$

...



# Argument-Agents

$$\text{=Subst-Agents: } \frac{\Phi[x]_p \quad x = y}{\Phi'[y]_p} =_{\text{Subst}}(p) \longrightarrow \frac{\text{prem} \quad eq}{\text{conc}} =_{\text{Subst}}(\text{pos})$$

$$\mathcal{A}_{\emptyset}^{eq} = \{\text{find an equality line } eq\}$$

$$\mathcal{A}_{\emptyset}^{\text{conc}, \text{prem}} = \left\{ \begin{array}{l} \text{find open line } \text{conc} \text{ and a support line } \text{prem} \text{ that differ} \\ \text{only wrt. occurrences of a single proper subterm} \end{array} \right\}$$

$$\mathcal{A}_{\{\text{conc}, \text{prem}\}}^{eq} = \{\text{find equality line } eq \text{ suitable for rewriting } \text{prem} \text{ into } \text{conc}\}$$

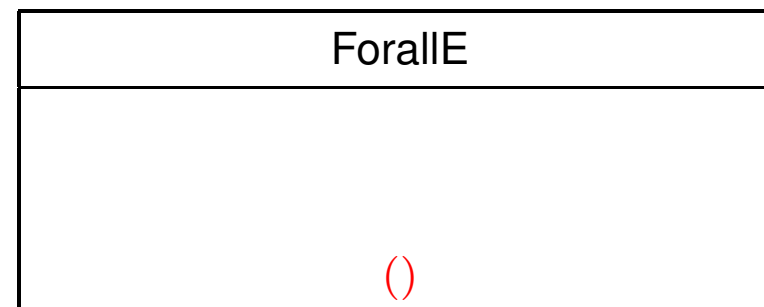
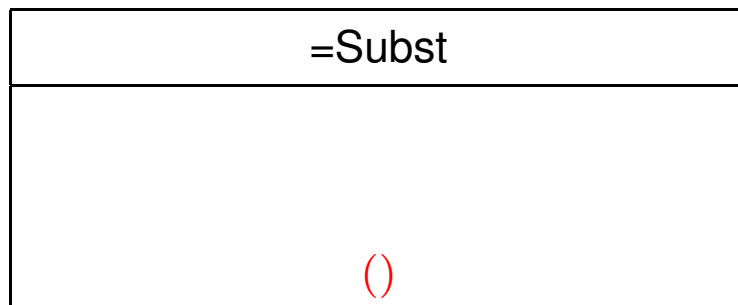
$$\mathcal{A}_{\{\text{prem}, \text{conc}\}}^{\text{pos}} = \{\text{compute positions where } \text{conc} \text{ and } \text{prem} \text{ differ}\}$$

... currently 25 agents specified for  $=_{\text{Subst}}$  ...



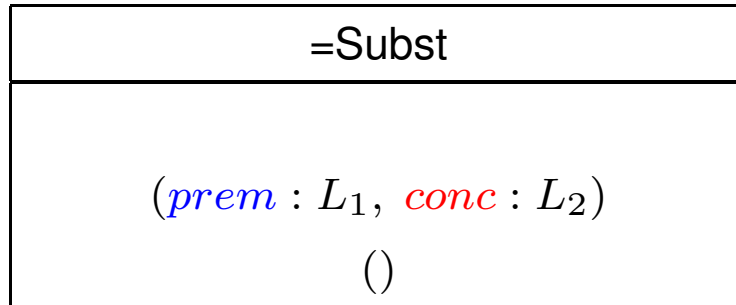
# Argument Agents at Work

$A_1$	$(A_1)$	$\vdash$	$\forall X_o \bullet p_{o \rightarrow o}(b_o \wedge X_o)$	Assumption
$A_2$	$(A_2)$	$\vdash$	$\forall Y_o \bullet p_{o \rightarrow o}(Y_o \wedge c_o)$	Assumption
$L_1$	$(L_1)$	$\vdash$	$p_{o \rightarrow o}(a_o \wedge b_o)$	Hyp
			...	
$L_2$	$(L_1)$	$\vdash$	$p_{o \rightarrow o}(b_o \wedge a_o)$	OPEN
$C$	$()$	$\vdash$	$p_{o \rightarrow o}(a_o \wedge b_o) \Rightarrow p_{o \rightarrow o}(b_o \wedge a_o)$	$\Rightarrow_I: (L_2)$

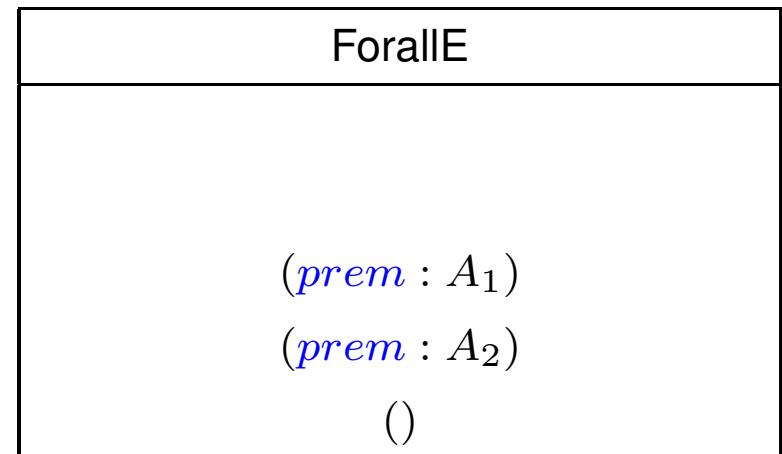


$A_1$	$(A_1)$	$\vdash$	$\forall X_o \cdot p_{o \rightarrow o}(b_o \wedge X_o)$	Assumption
$A_2$	$(A_2)$	$\vdash$	$\forall Y_o \cdot p_{o \rightarrow o}(Y_o \wedge c_o)$	Assumption
$L_1$	$(L_1)$	$\vdash$	$p_{o \rightarrow o}(a_o \wedge b_o)$	Hyp
			...	
$L_2$	$(L_1)$	$\vdash$	$p_{o \rightarrow o}(b_o \wedge a_o)$	OPEN
$C$	$()$	$\vdash$	$p_{o \rightarrow o}(a_o \wedge b_o) \Rightarrow p_{o \rightarrow o}(b_o \wedge a_o)$	$\Rightarrow_I: (L_2)$

$\mathcal{A}_\emptyset^{prem, conc}$



$\mathcal{A}_\emptyset^{prem}$



$A_1$	$(A_1)$	$\vdash$	$\forall X_o. p_{o \rightarrow o}(b_o \wedge X_o)$	Assumption
$A_2$	$(A_2)$	$\vdash$	$\forall Y_o. p_{o \rightarrow o}(Y_o \wedge c_o)$	Assumption
$L_1$	$(L_1)$	$\vdash$	$p_{o \rightarrow o}(a_o \wedge b_o)$	Hyp
			...	
$L_2$	$(L_1)$	$\vdash$	$p_{o \rightarrow o}(b_o \wedge a_o)$	OPEN
$C$	$()$	$\vdash$	$p_{o \rightarrow o}(a_o \wedge b_o) \Rightarrow p_{o \rightarrow o}(b_o \wedge a_o)$	$\Rightarrow_I: (L_2)$

 $\mathcal{A}_{\{prem, conc\}}^{pos}$ 

=Subst
$(prem : L_1, conc : L_2, pos : (1))$
$(prem : L_1, conc : L_2)$
$()$

 $\mathcal{A}_{\{prem\}}^{conc}$ 

ForallE
$(prem : A_1, conc : L_2)$
$(prem : A_1)$
$(prem : A_2)$
$()$



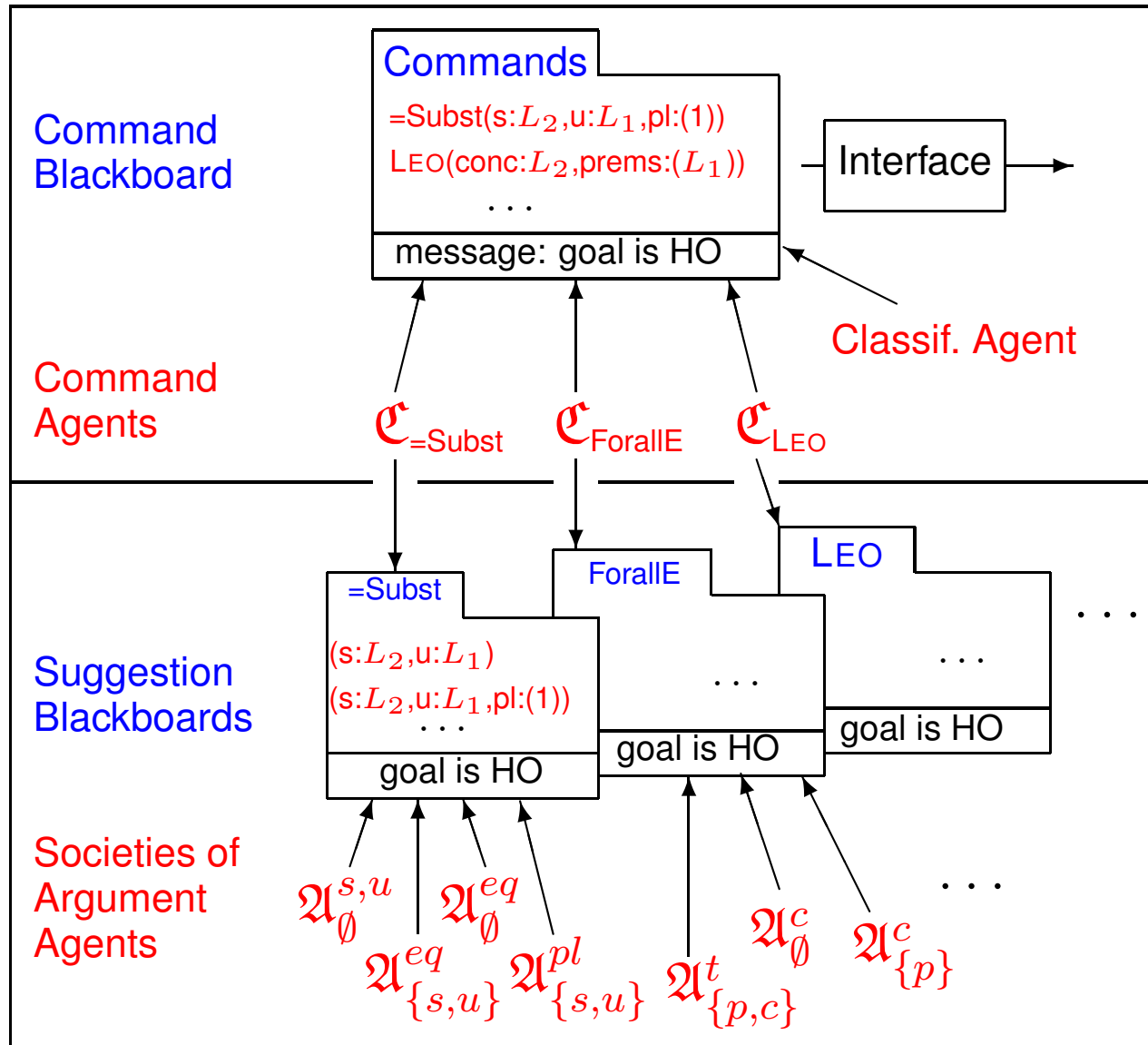
$A_1$	$(A_1)$	$\vdash$	$\forall X_o. p_{o \rightarrow o}(b_o \wedge X_o)$	Assumption
$A_2$	$(A_2)$	$\vdash$	$\forall Y_o. p_{o \rightarrow o}(Y_o \wedge c_o)$	Assumption
$L_1$	$(L_1)$	$\vdash$	$p_{o \rightarrow o}(a_o \wedge b_o)$	Hyp
			...	
$L_2$	$(L_1)$	$\vdash$	$p_{o \rightarrow o}(b_o \wedge a_o)$	<b>OPEN</b>
$C$	$()$	$\vdash$	$p_{o \rightarrow o}(a_o \wedge b_o) \Rightarrow p_{o \rightarrow o}(b_o \wedge a_o)$	$\Rightarrow_I: (L_2)$

<b>=Subst</b>
$(prem : L_1, conc : L_2, pos : (1))$
$(prem : L_1, conc : L_2)$
$()$

 $\mathcal{A}_{\{prem, conc\}}^{term}$ 

<b>ForallE</b>
$(prem : A_1, conc : L_2, term : a_o)$
$(prem : A_1, conc : L_2)$
$(prem : A_1)$
$(prem : A_2)$
$()$





# A Resource Concept is needed

- In large proof contexts **hundreds of agents may become active**
- They may employ **quite expensive algorithms**
  - higher-order unification/matching
  - comparing all subterm positions of terms
  - employ little deductive processes or external reasoning systems
  - ...
- The suggestion mechanism should **support different modi**
  - quick respond mode  $\longleftrightarrow$  lunch time mode



# A Resource Adapted Approach

- Complexity rating  $\gamma$  (mirrors computation costs): e.g.

$$\gamma(\mathcal{A}_{\emptyset}^{conc, prem}) = 10$$

and

$$\gamma(\mathcal{A}_{\emptyset}^{eq}) = 2$$

for command

$$\frac{pre\text{m} \quad eq}{conc} = \text{Subst}(pos)$$

- User specifies a global **deactivation threshold** in order to suppress computations of ‘expensive’ agents

Disadvantages:

**Tuning by hand** not reflects real costs, no dynamic adjustments, tuning of hundreds of single argument agents



# A Resource Adaptive Approach

## 1. Self adjustments of the argument agents

- Argument agents **monitor and adjust** their **own performance and contributions** in the past using as criterions:
  - cpu time the argument agents consume
  - penalty for failing several times in a row
- **Resource adjustments** are **communicated** to command agents **via blackboards**



# A Resource Adaptive Approach

## 2. Explicit reasoning on suitable resource distributions

- The command agents accumulate the ratings of their argument agents and pass them to a **special resource agent** via the command blackboard
- Resource agent reasons on **global adjustments** and communicates them to the lower layer agents via the blackboards
- Aspects of the resource reasoning
  - avoid complete retirement
  - user specific preference for certain commands
  - switch of proof context
  - ... and ...



# A Resource Adaptive Approach

## 3. Informed activation & deactivation

- Rules, tactics, methods are associated with a specific logic or theory  
( $\Omega$ MEGA provides special proof methods for limit-theorems)
- Problems (in mathematics) are associated with a specific logic or theory  
(Extensionality rules belong to HO, ForallE belongs to FO, AndE belongs to PL)

⇒ Use knowledge to activate appropriate and deactivate inappropriate agents

- A classification agent tries to classify each new subproblem and passes this information via the blackboards to the argument agents
- Argument agents can themselves decide whether they belong to this class and get active or inactive



# Applications

- Suggestion mechanism for **interactive systems**
  - suggest next interactive step
  - complete partial command instantiations suggested by the user
- Support in **proof planning**: compute applicable proof methods in parallel
- Suggestion mechanism enriched with simple backtracking over the suggestions = an **automated theorem prover (proof planner)**



# Conclusion

Advantages of the proposed suggestion mechanism to other systems:

- Flexibility
- Anytime character
- Robustness
- Expandability, User adaptability, Manageability

What interactive theorem provers gain:

- Better **exploitation of available resources** as in traditional systems  
(steadily works background, autonomous, concurrent, resource adaptive)

