

# Dealing with incomplete normative states







**JUAN MANUEL SERRANO & SERGIO SAUGAR**  
COMPUTING DEPARTMENT  
UNIVERSITY REY JUAN CARLOS  
MADRID, SPAIN

**COST ACTION AT  
WORKSHOP ON NORMS (WG2)**

**AYIA NAPA, CYPRUS**  
**15/12/2009**

# Normative incompleteness

2

<b>SOURCES</b> <b>APPROACHES</b>	<b>Underspecification</b> (i.e. general rules exist but they are not programmed)	<b>Inherent incompleteness</b> (i.e. lack of general rules)
<b>Default assumptions</b> (e.g. “if we don’t know that something is permitted, then it is prohibited”)		
<b>Incomplete normative states</b> (i.e. <i>unknown</i> normative relations are explicitly represented)		

# Management of university courses

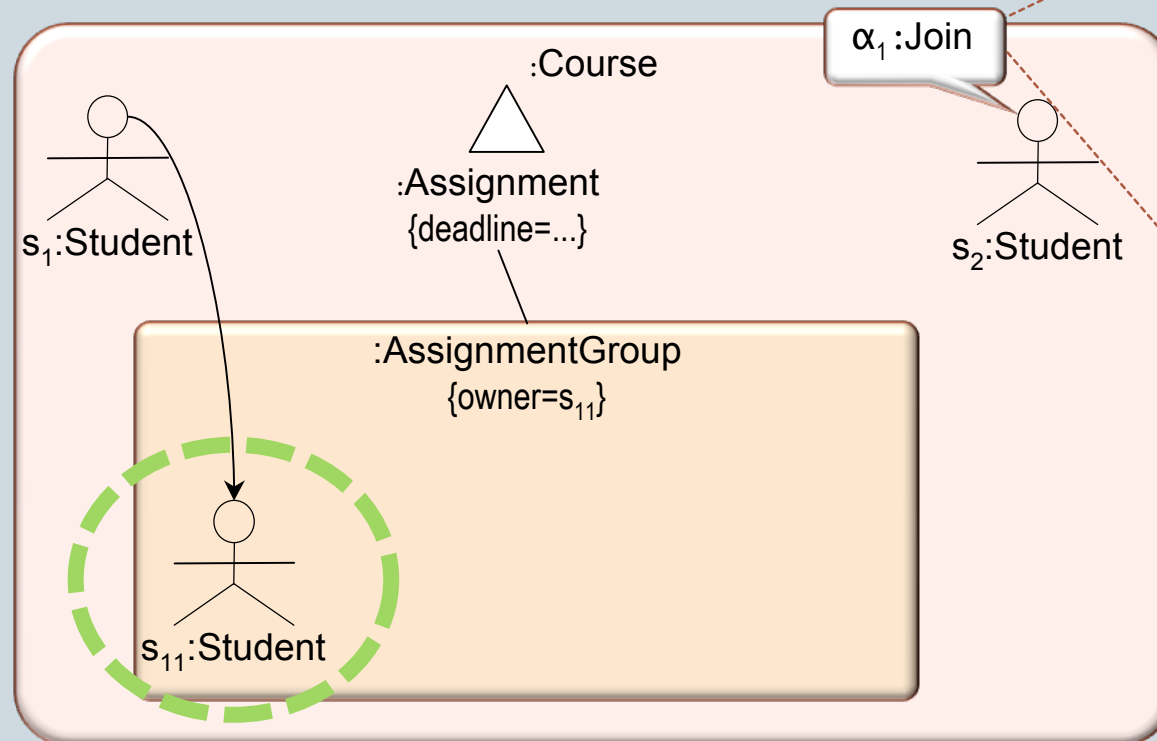
3

## EMPOWERMENT RULES

Students are empowered to *Join* assignment groups **if, and only if**, they have not passed that assignment, and do not already participate in another group for the same assignment

## PERMISSION RULES

Students are **only** permitted to *Join* an assignment group **if** the deadline for submitting the assignment has not yet passed



# Goals & Solution

4

## Goals

- Provide a normative framework which is able to represent and manage incomplete states for empowerment and permissions relations
- Provide a mechanism that allows agents to resolve incompleteness at run-time

## Solution

- Use of ASP and action language K to represent and reason about incomplete states
- Speech acts as *social entities*
  - *Empowerment* rules govern their creation
  - *Permission* rules, their execution
  - Possible states: executed, prohibited and *pending* for execution
- Tailor-made speech acts *allow* and *forbid* to resolve incompleteness

# K-specification of social actions

5

social\_action.plan

## fluents:

```
state_sa(Act,S)    requires social_action(Act), social_action_state(S).
empowered(Act)    requires social_action(Act).
permitted(Act)    requires social_action(Act).
```

...

## actions:

```
attempt(Act) requires social_action(Act).
perform(Act) requires social_action(Act).
```

## always:

...

```
executed perform(Act) if
  attempt(Act), empowered(Act), permitted(Act).
```

...

```
caused state_sa(Act,pending) after
  attempt(Act), empowered(Act),
  not permitted(Act), not -permitted(Act).
```

...

```
executed perform(Act) if
  state_sa(Act,pending), permitted(Act).
```

...

# K-specification of allow/forbid

6

allow.plan

## fluents:

action(Allow,Act) **requires** allow(Allow), social\_action(Act).

...

## always:

...

**caused** permitted(Act) **if** state\_sa(Act,pending) **after**  
allow(Allow), perform(Allow), action(Allow,Act).

...

forbid.plan

## fluents:

action\_f(Allow,Act) **requires** forbif(Allow), social\_action(Act).

...

## always:

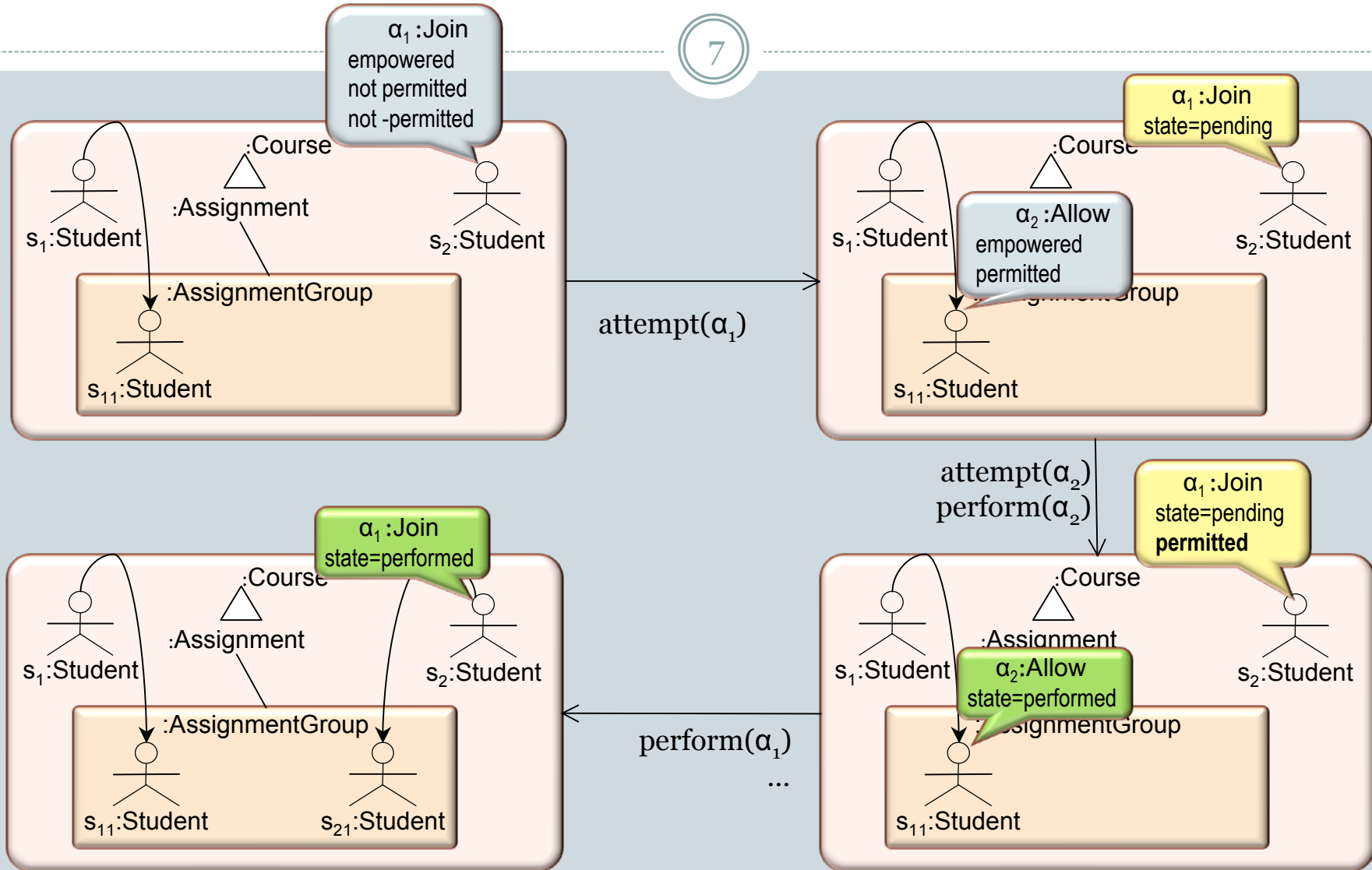
...

**caused** -permitted(Act) **after**  
forbid(Forbid), perform(Forbid), action\_f(Forbid,Act).

...

# Course management revisited

7



# Conclusion

8

- **Authorization hierarchies**
  - ad-hoc hierarchies vs. general hierarchies
- **Allow/Forbid vs. Permit/Prohibit**
  - Ad-hoc permissions vs. general rules (normative dynamics)
- **Weak/strong negation useful for representing ...**
  - Necessary conditions for normative relations
  - Semantics of normative speech acts
- **This work is part of SPEECH**
  - A language for programming social processes
  - Allow & forbid are part of the standard library of speech acts
  - Current & future work: commitments, full-fledged speech acts, ...

**<http://www.speechlang.org/k>**

# Dealing with incomplete normative states



JUAN MANUEL SERRANO & SERGIO SAUGAR  
COMPUTING DEPARTMENT  
UNIVERSITY REY JUAN CARLOS  
MADRID, SPAIN

**THANK YOU FOR YOUR ATTENTION!**

COST ACTION AT  
WORKSHOP ON NORMS (WG2)

AYIA NAPA, CYPRUS  
15/12/2009