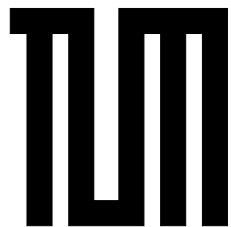
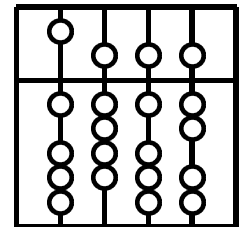


# RNS – A Schema for Specifying Computational Autonomy

Matthias Nickles, Michael Rovatsos, Gerhard Weiß



Computer Science Department  
Technical University of Munich  
{nickles,rovatsos,weissg}@in.tum.de



- ▶ Motivation
- ▶ Basic Specification Constructs of **RNS**
- ▶ Specifying Norms and Sanctions with **RNS**
- ▶ Specifying Activities with **RNS**
- ▶ Discussion
- ▶ Conclusion

- ▶ Autonomy is a key property of computational agency

- ▶ Autonomy is a key property of computational agency
- ▶ Potential to be enabling technology for broad range of important applications
  - telecommunications, logistics, e/m-commerce, pervasive and ubiquitous computing
  - open, dynamic, networked, decentralized, unpredictable applications

- ▶ Specification of kind and level of autonomy is most critical engineering challenges
- ▶ *Autonomy specification dilemma:*
  - too rigid → suppression of necessary action choice
  - too generous → admission of unnecessary action choice

- ▶ Specification of kind and level of autonomy is most critical engineering challenges
- ▶ *Autonomy specification dilemma:*
  - too rigid → suppression of necessary action choice
  - too generous → admission of unnecessary action choice
- ▶ Strong need for techniques (methods, formalisms, tools, etc.) to specify computational autonomy
- ▶ **RNS** (“Roles, Norms, Sanctions”) developed in response to this need

- ▶ Agents are embedded in a social frame which regulates (but does not fully constrain) their behavior
- ▶ The social frame, called role space, is composed of roles

```
ROLE SPACE role_space_id { role_id_list }
```

where

*role\_space\_id* = unique role space identifier

*role\_id\_list* = list of unique role identifiers

- ▶ Agents must act as role owners, they can try to achieve their goals through playing roles
- ▶ Conceptually, roles
  - serve as as a means for specifying desired behavior
  - are *not* viewed as a means for fully constraining behavior (they leave room for individuality!)



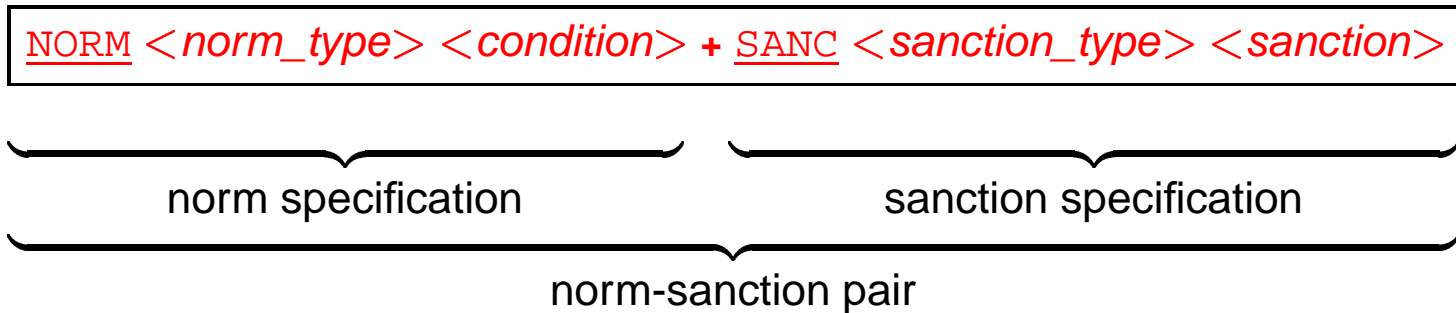
- ▶ Agents must act as role owners, they can try to achieve their goals through playing roles
- ▶ Conceptually, roles
  - serve as as a means for specifying desired behavior
  - are *not* viewed as a means for fully constraining behavior (they leave room for individuality!)
- ▶ Specifying a role requires to specify the activities that are of relevance to an agent playing this role

```
ROLE role_id { activity_list }
```

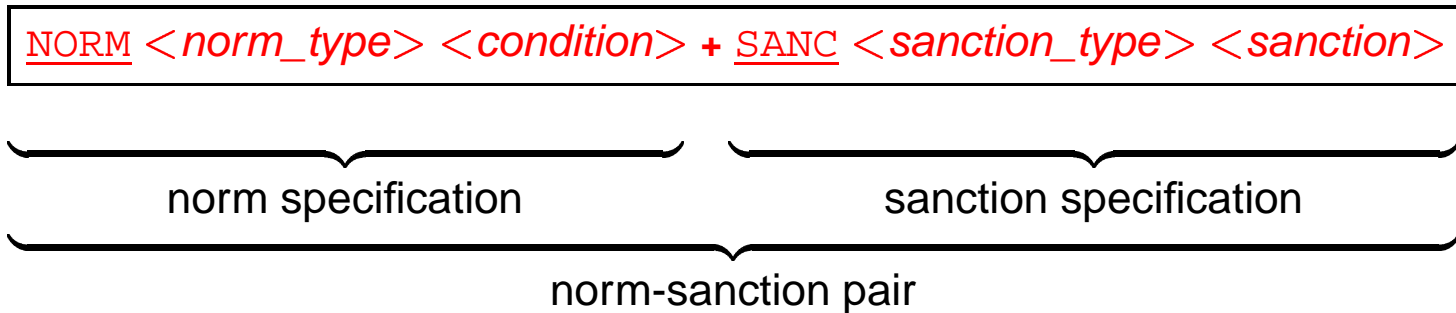
- ▶ Agents must act as role owners, they can try to achieve their goals through playing roles
- ▶ Conceptually, roles
  - serve as as a means for specifying desired behavior
  - are *not* viewed as a means for fully constraining behavior (they leave room for individuality!)
- ▶ Specifying a role requires to specify the activities that are of relevance to an agent playing this role

```
ROLE role_id { activity_list }
```
- ▶ Specifying an activity requires to specify the norms and sanctions the activity is subject to

- ▶ Pairwise specification of norms and sanctions:



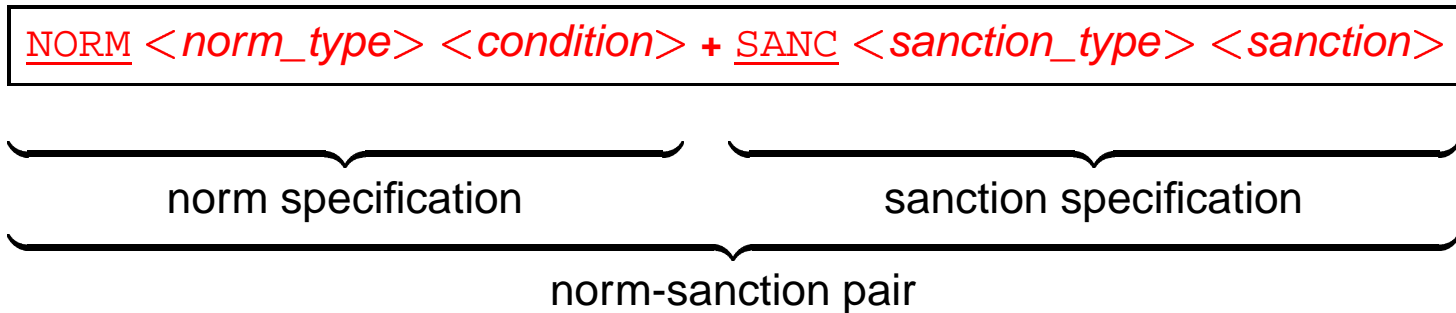
- ▶ Pairwise specification of norms and sanctions:



- ▶ Three types of norms:

- permissions (**P**), “agent *may* do x”
- obligations (**O**), “agent *must* do x”
- interdictions (**I**), “agent *must not* do x”

- ▶ Pairwise specification of norms and sanctions:



- ▶ Three types of norms:

- permissions (**P**), “agent *may* do x”
- obligations (**O**), “agent *must* do x”
- interdictions (**I**), “agent *must not* do x”

- ▶ Two types of sanctions:

- reward (**RE**) of norm-conforming behavior
- punishment (**PU**) of norm-deviating behavior

- ▶ Two types of norm-sanction pairs

- ▶ Two types of norm-sanction pairs
  - “*DEPENDENT*”: become relevant after explicit request for (not) executing the activity

`<DEP role_id> : norm-sanction_pair`

  
status type

- ▶ Two types of norm-sanction pairs
  - “*DEPENDENT*”: become relevant after explicit request for (not) executing the activity

`<DEP role_id> : norm-sanction_pair`

⏟  
status type

- “*INDEPENDENT*”: valid regardless of requests

`<IND> : norm-sanction_pair`

⏟  
status type



- ▶ General syntactic form of dependent and independent norm-sanction pairs:

`<status_type> : norm-sanction_pair`

status statement

- ▶ Specifications of this form are called *status statements*

- ▶ General syntactic form of dependent and independent norm-sanction pairs:

`<status_type> : norm-sanction_pair`

status statement

- ▶ Specifications of this form are called *status statements*
- ▶ The list of all status statements attached to an activity is called the activity's *status range*

`STATUS_RANGE status_statement_list`

- ▶ General syntactic form of dependent and independent norm-sanction pairs:

`<status_type> : norm-sanction_pair`

status statement

- ▶ Specifications of this form are called *status statements*
- ▶ The list of all status statements attached to an activity is called the activity's *status range*

`STATUS_RANGE status_statement_list`

- ▶ Example:

STATUS\_RANGE

`<IND> : NORM <P> <NO> + SANC <NO> <NO>`

`<DEP EACH> : NORM <O> <quantity ≤ 100> + SANC <PU> <withdraw_role>`

`<DEP AssemblyMg> : NORM <I> <material = steel> + SANC <PU> <pay_fine>`

- ▶ Four types of activities
  - *basic activities*, i.e. resource and event handling
  - *execution request activities*, i.e. requests for executing activities
  - *sanctioning activities*, i.e. activities that result in punishment (reward) of norm-deviating (norm-conforming) behavior
  - *change activities*, i.e. activities that result in changes of norms and/or sanctions

- ▶ Four types of activities
  - *basic activities*, i.e. resource and event handling
  - *execution request activities*, i.e. requests for executing activities
  - *sanctioning activities*, i.e. activities that result in punishment (reward) of norm-deviating (norm-conforming) behavior
  - *change activities*, i.e. activities that result in changes of norms and/or sanctions
- ▶ Complex activity constructs are possible, e.g. requests for sanction, requests for requests (for requests for ...), requests for changes of norms, ...

- ▶ *Basic activities* have the general syntax

```
ACT activity_id ( activity_variable_list )  
    { STATUS_RANGE status_statement_list }
```

where

STATUS\_RANGE = as explained above

- ▶ *Basic activities* have the general syntax

```
ACT activity_id ( activity_variable_list )  
    { STATUS RANGE status_statement_list }
```

where

STATUS RANGE = as explained above

- ▶ Example of a *basic activity*

```
ACT deliver ( material,quantity )  
    { STATUS RANGE  
      <IND> : NORM <P> <NO> + SANC <NO> <NO>  
      <DEP EACH> : NORM <O> <quantity ≤ 100> + SANC <PU> <withdraw_role>  
      <DEP AssemblyMg> : NORM <I> <material = steel> + SANC <PU> <pay_fine>  
    }
```

- ▶ *Execution request activities* have the general syntax

```
ACT REQUEST ( agent_id_list ; role_id_list ; [NOT] activity_id ( activity_variable_list ) )  
  { STATUS RANGE status_statement_list  
    NORMATIVE IMPACT norm_specification_list }
```

where

- ▶ NORMATIVE IMPACT = normative impact of the request on the requested agent(s)
- ▶ *norm\_specification\_list* = list of norm specifications of the form  
NORM <*norm\_type*> <*condition*>



- ▶ *Execution request activities* have the general syntax

```
ACT REQUEST ( agent_id_list ; role_id_list ; [NOT] activity_id ( activity_variable_list ) )  
  { STATUS RANGE status_statement_list  
    NORMATIVE IMPACT norm_specification_list }
```

where

- ▶ NORMATIVE IMPACT = normative impact of the request on the requested agent(s)
- ▶ norm\_specification\_list = list of norm specifications of the form  
NORM <norm\_type> <condition>

- ▶ Example of an *execution request activity*

```
ACT REQUEST ( EACH ; USsupplier, EUROsupplier ; NOT deliver ( material, quantity ) )  
  { STATUS RANGE  
    <IND> : NORM <P> < (material = steel) AND (rating(material) = poor)> +  
      SANC <NO> <NO>  
    NORMATIVE IMPACT  
      NORM <I> <material = steel>  
  }
```

- ▶ *Sanctioning activities* have the general syntax

```
ACT SANCTION ( agent_id_list ; role_id_list ; activity_id ; norm_spec )  
  { STATUS RANGE status_statement_list  
    SANCTIONING IMPACT sanction_specification_list }
```

where

SANCTIONING IMPACT = list of sanction specifications of the form

SANC <*sanction\_type*> <*sanction*>

- ▶ *Sanctioning activities* have the general syntax

```
ACT SANCTION ( agent_id_list ; role_id_list ; activity_id ; norm_spec )  
  { STATUS RANGE status_statement_list  
    SANCTIONING IMPACT sanction_specification_list }
```

where

SANCTIONING IMPACT = list of sanction specifications of the form

SANC <*sanction\_type*> <*sanction*>

- ▶ Example of a *sanctioning activity*

```
ACT SANCTION ( EACH ; EACH ; deliver ; NORM <O> <quantity ≤ 100> )  
  { STATUS RANGE  
    <IND> : NORM <P> <NO> + SANC <RE> <earn_bonus>  
    <DEP RoleSpaceMg> : NORM <I> <NO> + SANC <PU> <withdraw_role>  
    SANCTIONING IMPACT  
    SANC <PU> <withdraw_role>  
  }
```

- ▶ *Change activities* result in changes of norms and/or sanctions by modifying the status range of activities
- ▶ Three types of changes:
  - DEL (delete): a status statement is removed from the status range of an activity
  - REP (replace): a new status statement replaces an existing one
  - ADD (add): a new status statement is added
- ▶ Formal syntax and examples of DEL, REP, and ADD straightforward (cf. paper)

- ▶ **RNS** offers several useful features
  - domain- and application-independent
  - neutral w.r.t. autonomy (respects autonomy specification dilemma)
  - grounded in sociological role theory
  - strongly expressive
    - explicit specification of sanctions
    - explicit specification of changes
    - allows for different normative impacts of the same activity (“context sensitivity”)
    - supports “composed activities” (e.g. requests for changes)

- ▶ Key differences from other approaches:
  - expressiveness
  - no assumptions on agent-level (cognitive) processes
  - caters for norm-violating behavior

- ▶ Key differences from other approaches:
  - expressiveness
  - no assumptions on agent-level (cognitive) processes
  - caters for norm-violating behavior
- ▶ **RNS** leaves room for improvement
- ▶ Major deficiencies:
  - relationships among roles (e.g., generalization, aggregation, inheritance) cannot be expressed
  - no support for conflict identification and resolution (e.g., permission and interdiction of the same activity)
- ▶ Currently: work on the above + development of specification tool