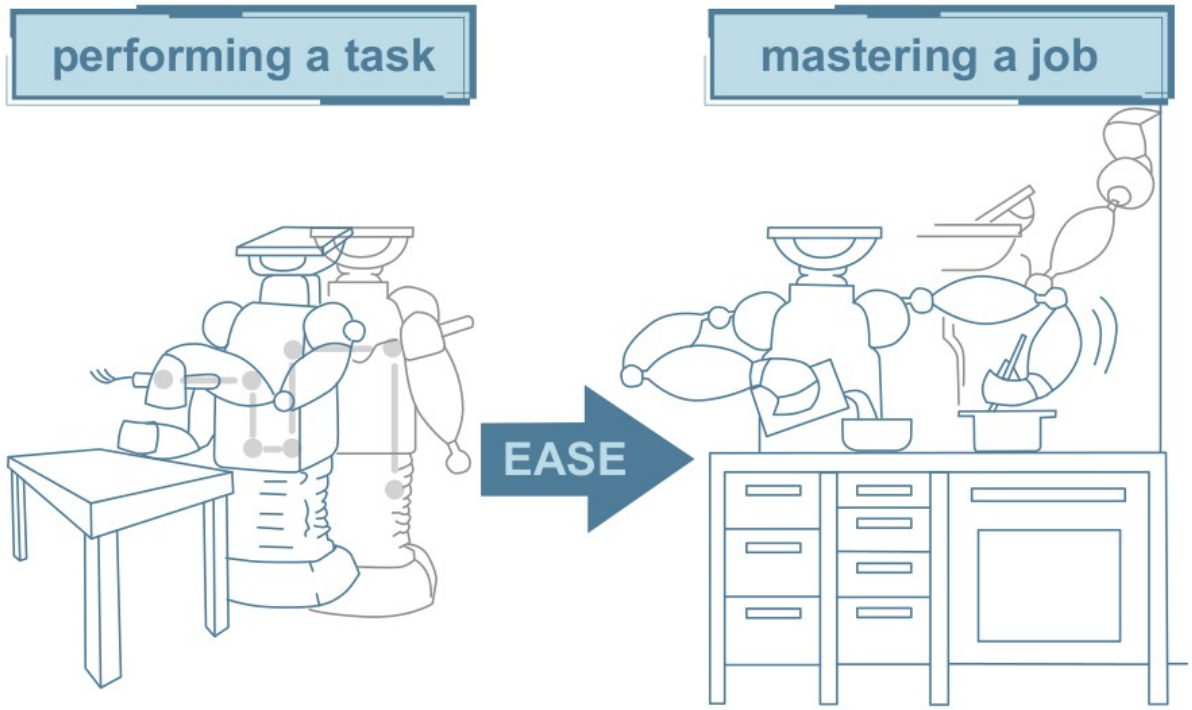
Embodied semantics for the language of action and change: Combining analysis, reasoning and simulation

Mihai Pomarlan, John Bateman

The big challenge: robots in everyday settings

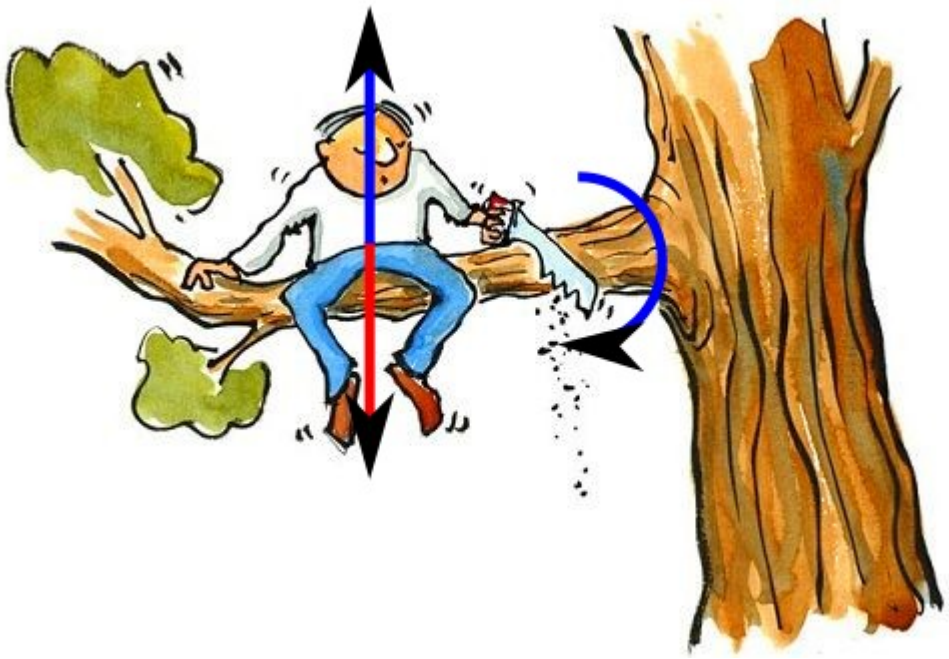


- EASE: “from performing specific tasks with specific objects in specific contexts to mastering human-scale everyday manipulation tasks”
- such competence would also involve communication
- interesting linguistically to study how simulation may help understanding

(One) big obstacle: common sense

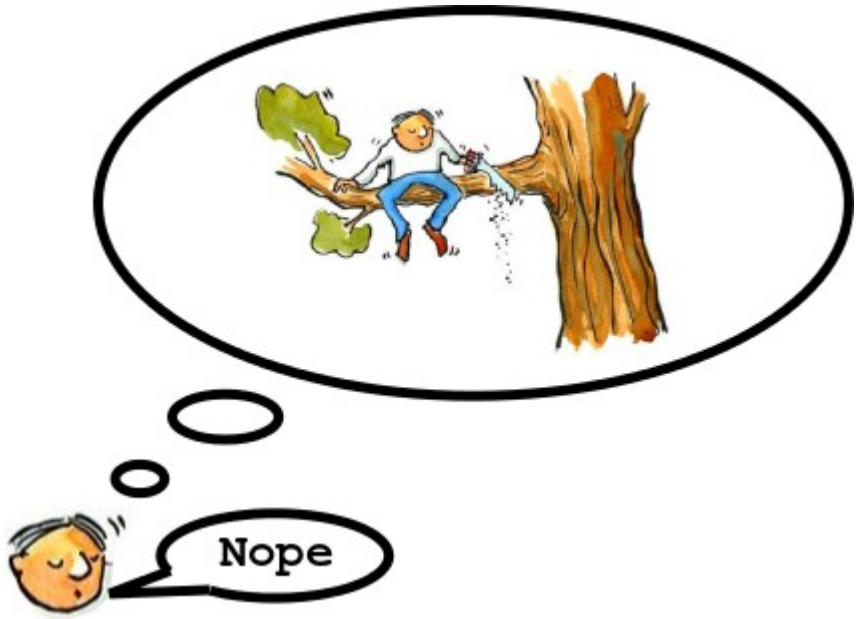
- “easy” to do-- for humans-- and therefore rarely articulated
- unclear how to formally capture it
- essential to handling the variability of semi-/unstructured physical environments

One potential approach: simulation



- has its own problems (cf. Davis' papers)
- appears used in the brain in some capacity
- a very straightforward technique to incorporate physical knowledge and reasoning about experience

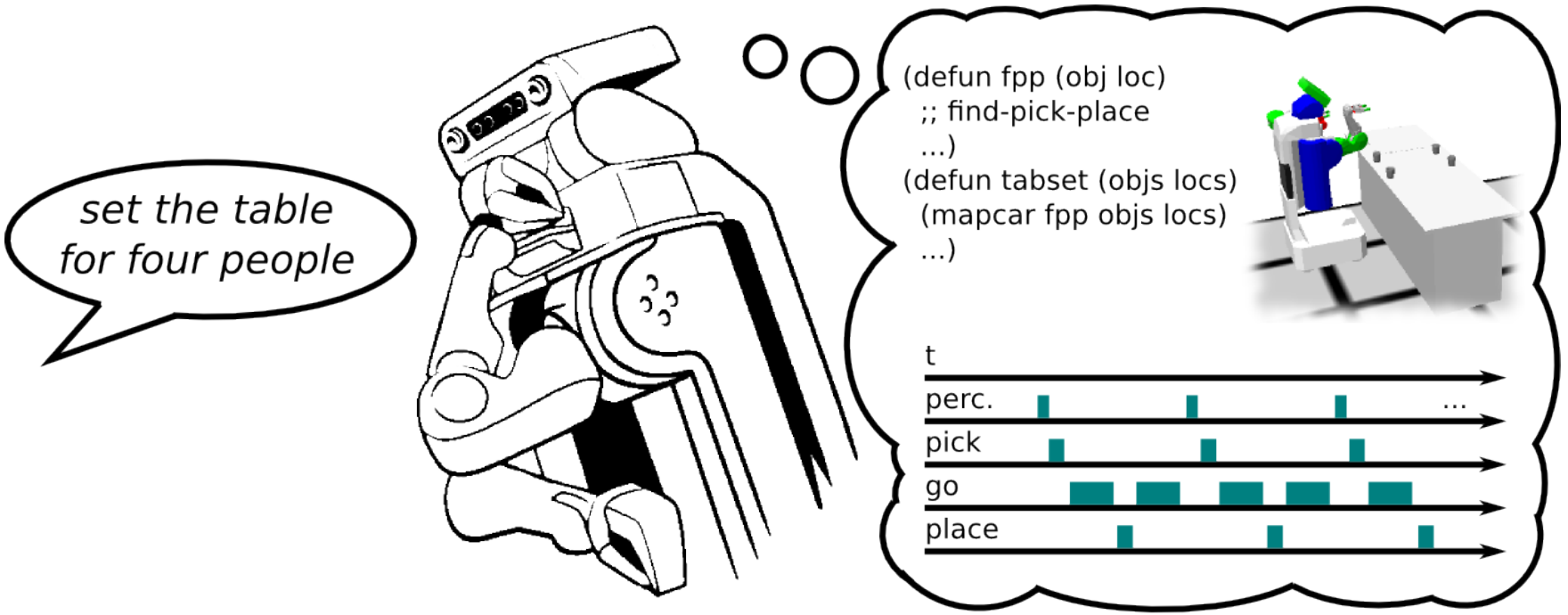
Hence: embodied cognition through simulation



- simulation as cheaper trial and error
  - analyzing events to improve future performance
- simulation as a reasoning technique to incorporate constraints resulting from embodiment
  - from embodiment constraints to interpretation constraints

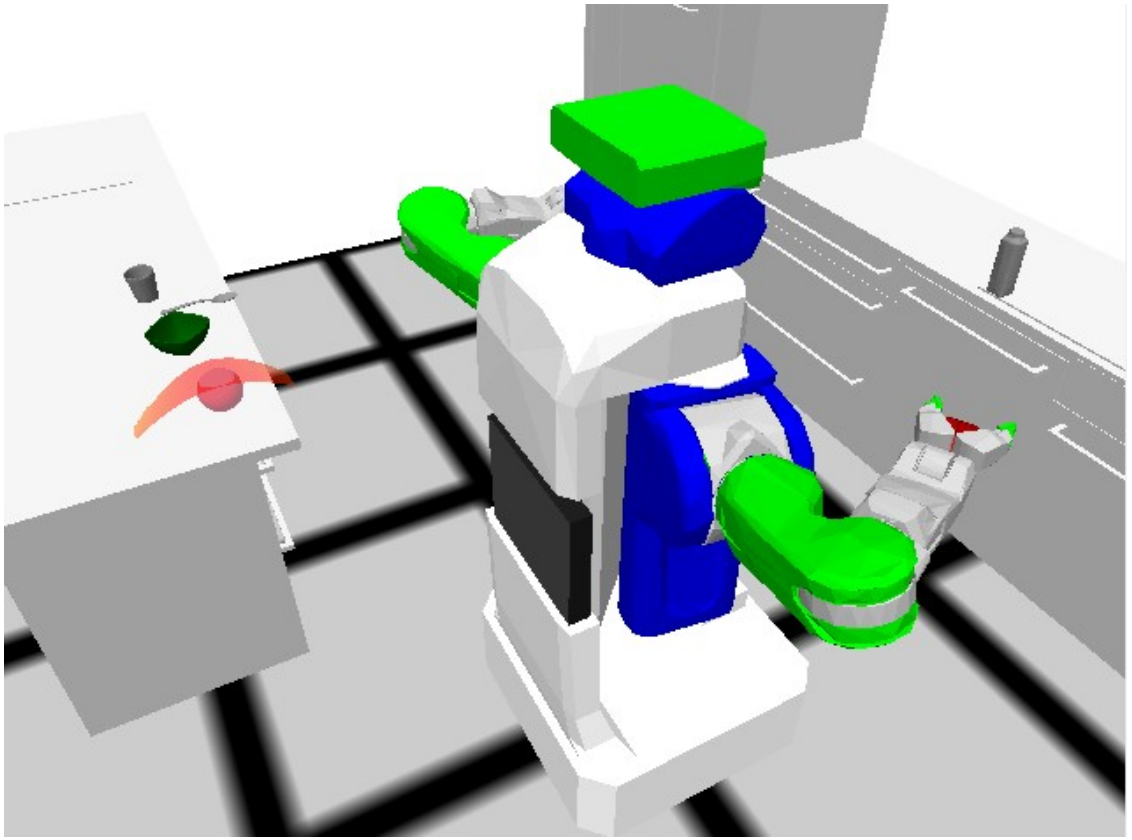
I: Constructing programs

From semantic specification to program



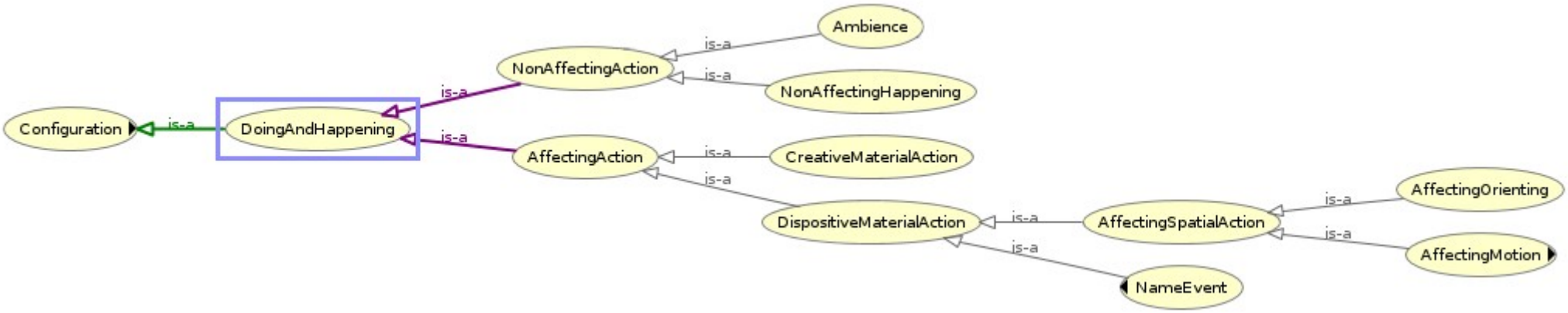
- assume a semantic specification (“semspec”) given by a parser
- our job: create a robot program from the semspec
  - programs constructed from basic building blocks (a “plan library”)
- use simulation (“projection”) to obtain better parametrizations

Background: CRAM



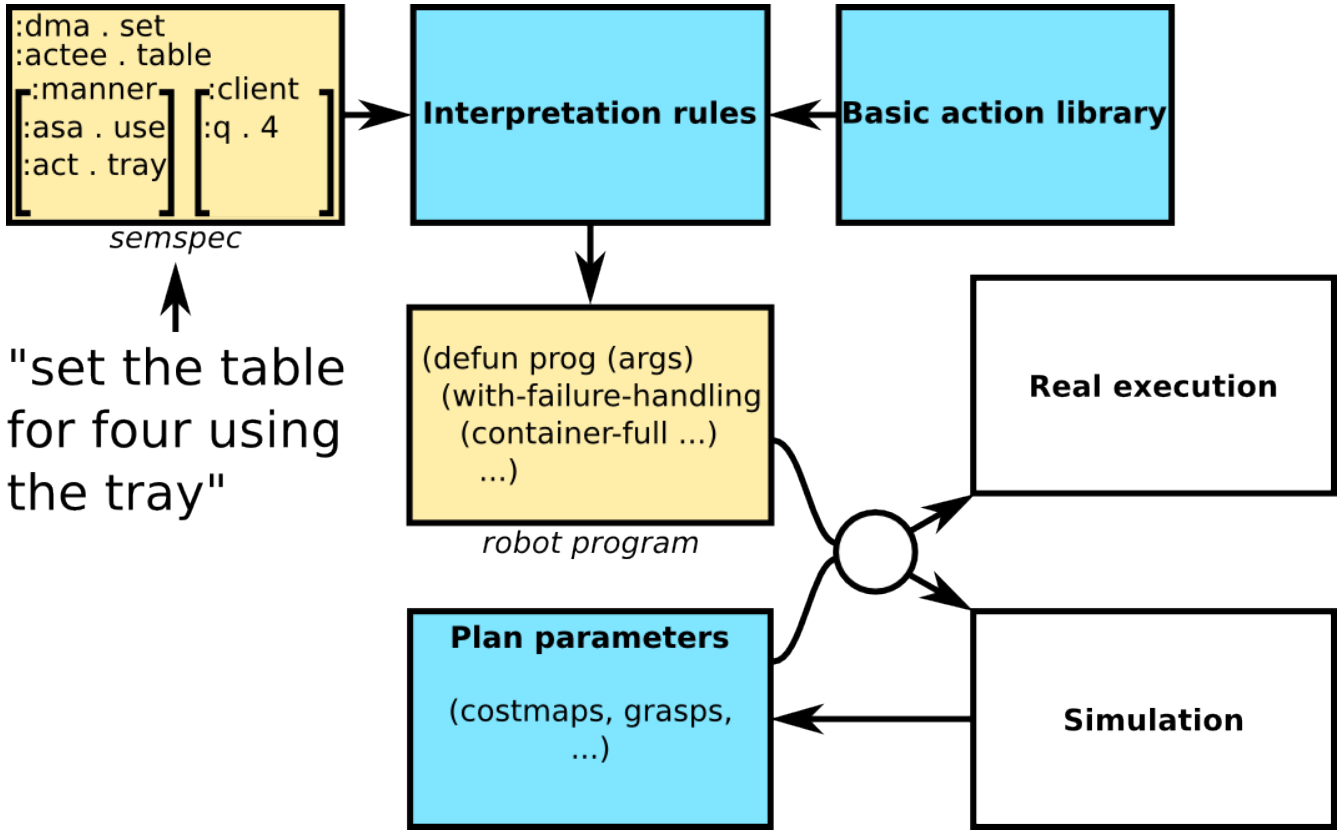
- Cognitive Robot Abstract Machine: a language and software framework to develop reactive robots
- designators: semantically refer to objects, locations, actions
- projection: light-weight, fast simulation

Background: GUM[Space]



- GUM: linguistically motivated upper ontology
- GUM-Space: extension including spatio-temporal concepts

Conversion overview



- semspec keys are concepts from the GUM ontology
- rule-based generation of plans from semspecs

Rule patterns

```
((:-
    [(:a . x) (:old-key . ?y) ...]
    [(:new-key . ?y) ...])
  (:@ :c :d))
```

- antecedent: something to look for in a semspec
- consequent: what to replace it with
- scope restriction: where to look for the antecedent
- unify variables in the antecedent pattern with the values in the semspec being processed by the rule

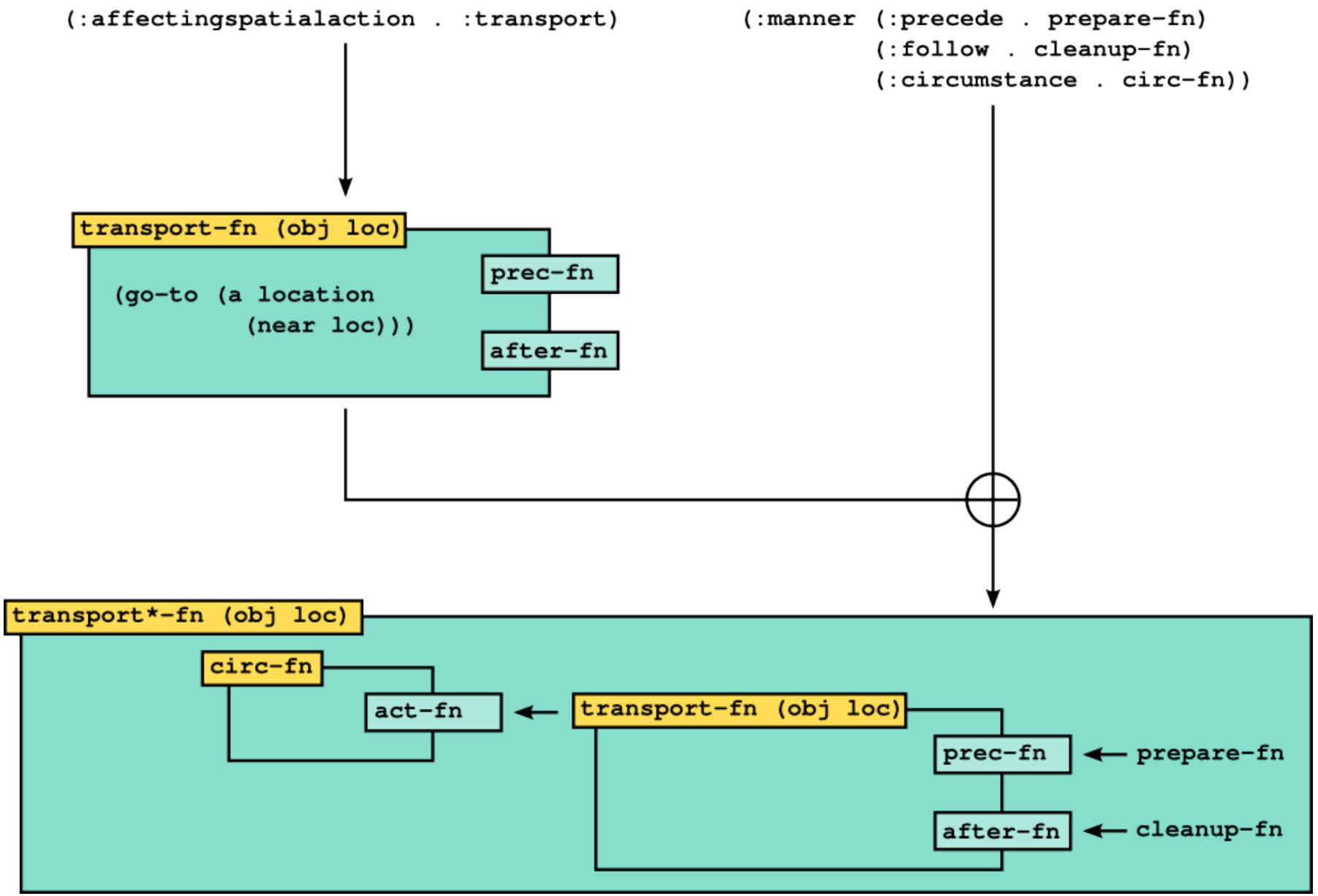
Rule types

- idiom: short phrase to longer description
- fusion: construct function objects and designators from subclauses
- execution preparation: create the top level plan

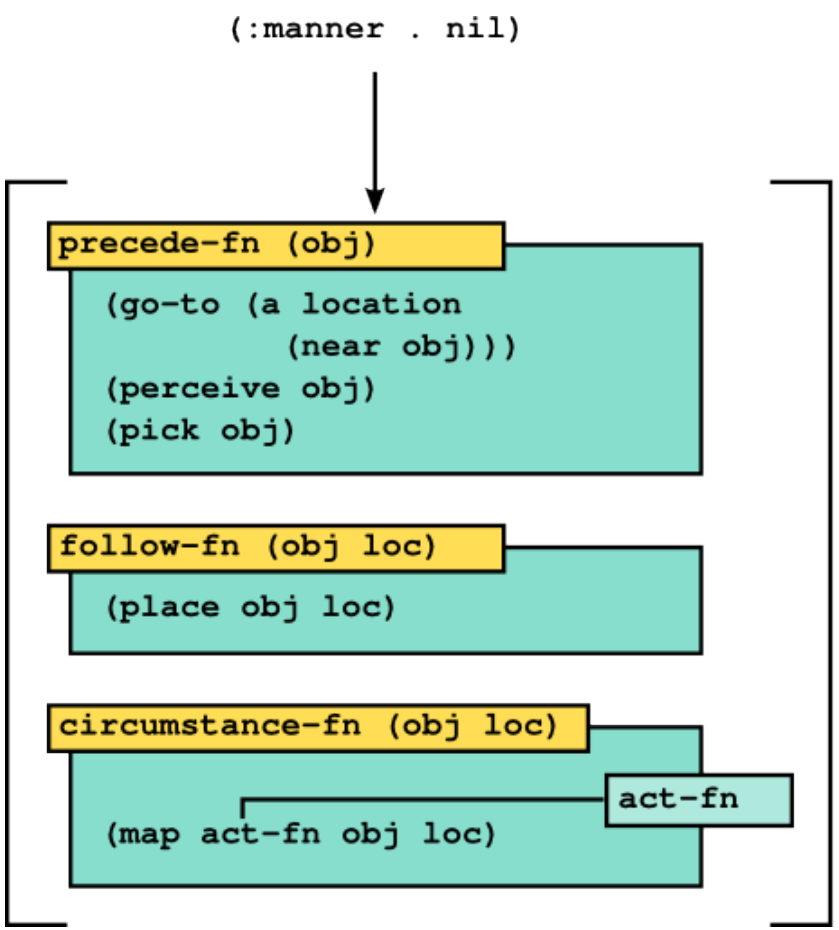
Rule ordering

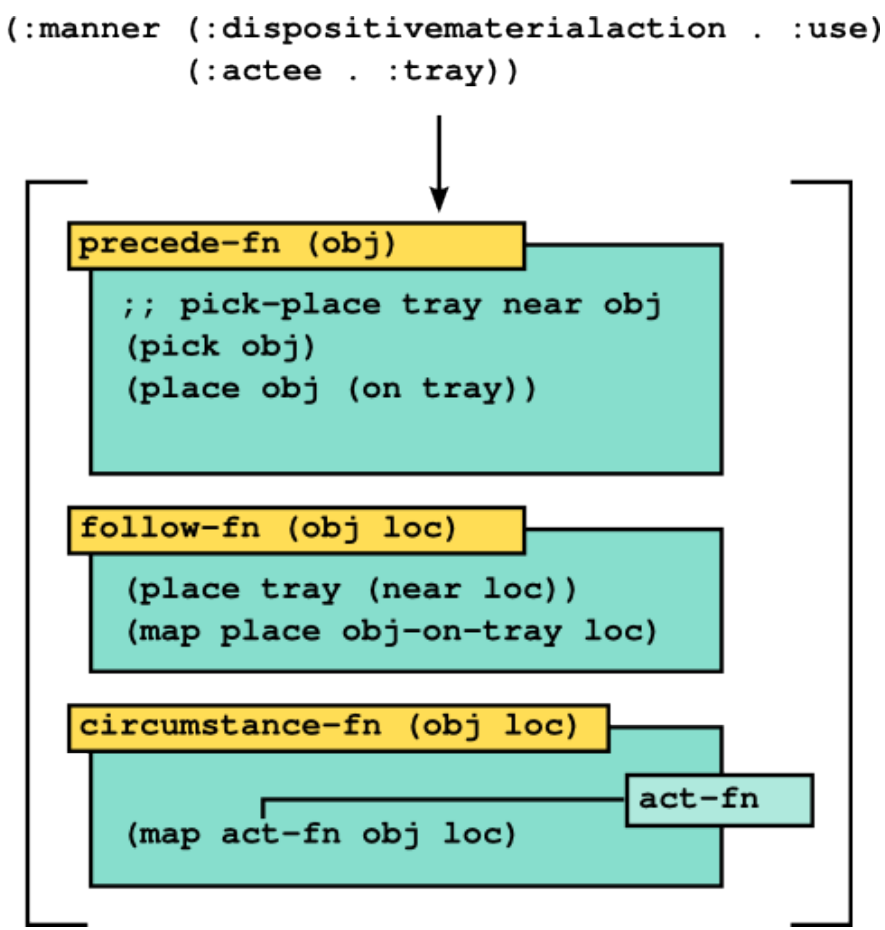
- in general rule application is not commutative
  - hence application order matters
- order by type (idiom, fusion, execution preparation)
- order by specificity
  - of scope restriction
  - of antecedent (subsumption induced by unification)
- order by user-provided score

Example: the transport action

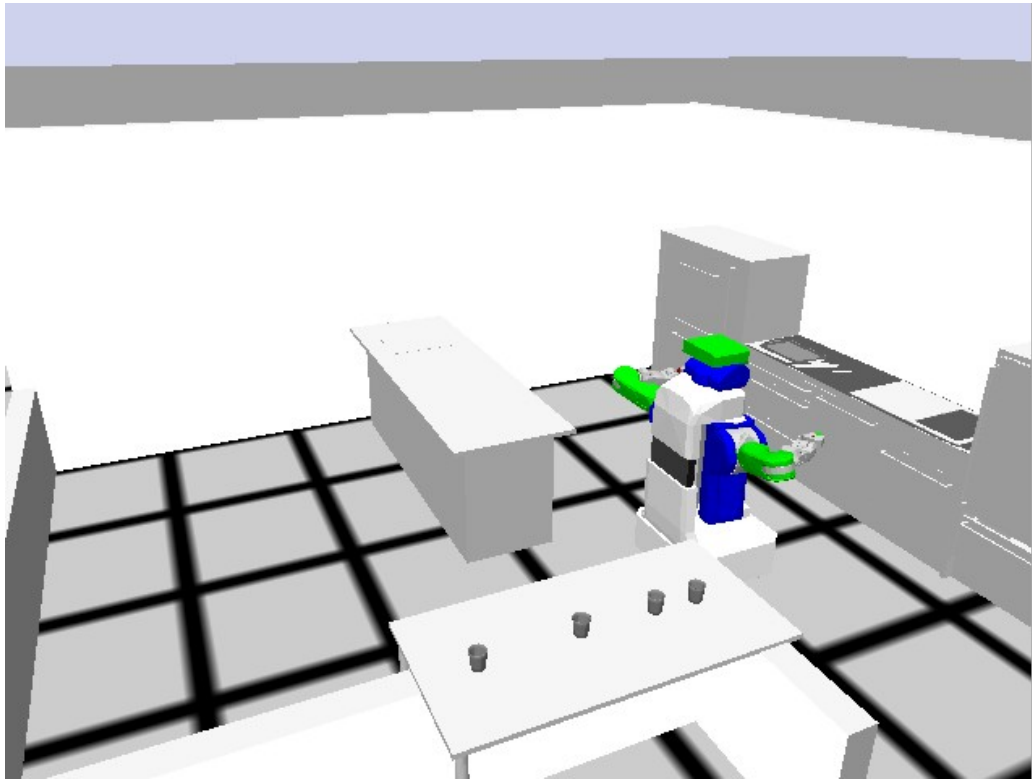


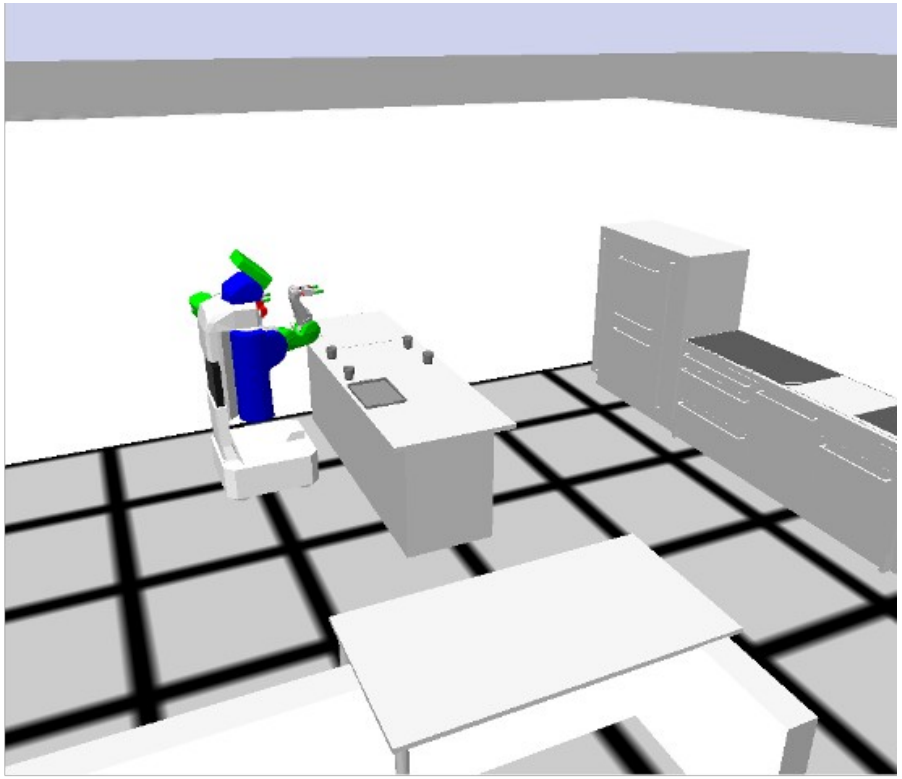
Default vs. mannered transport





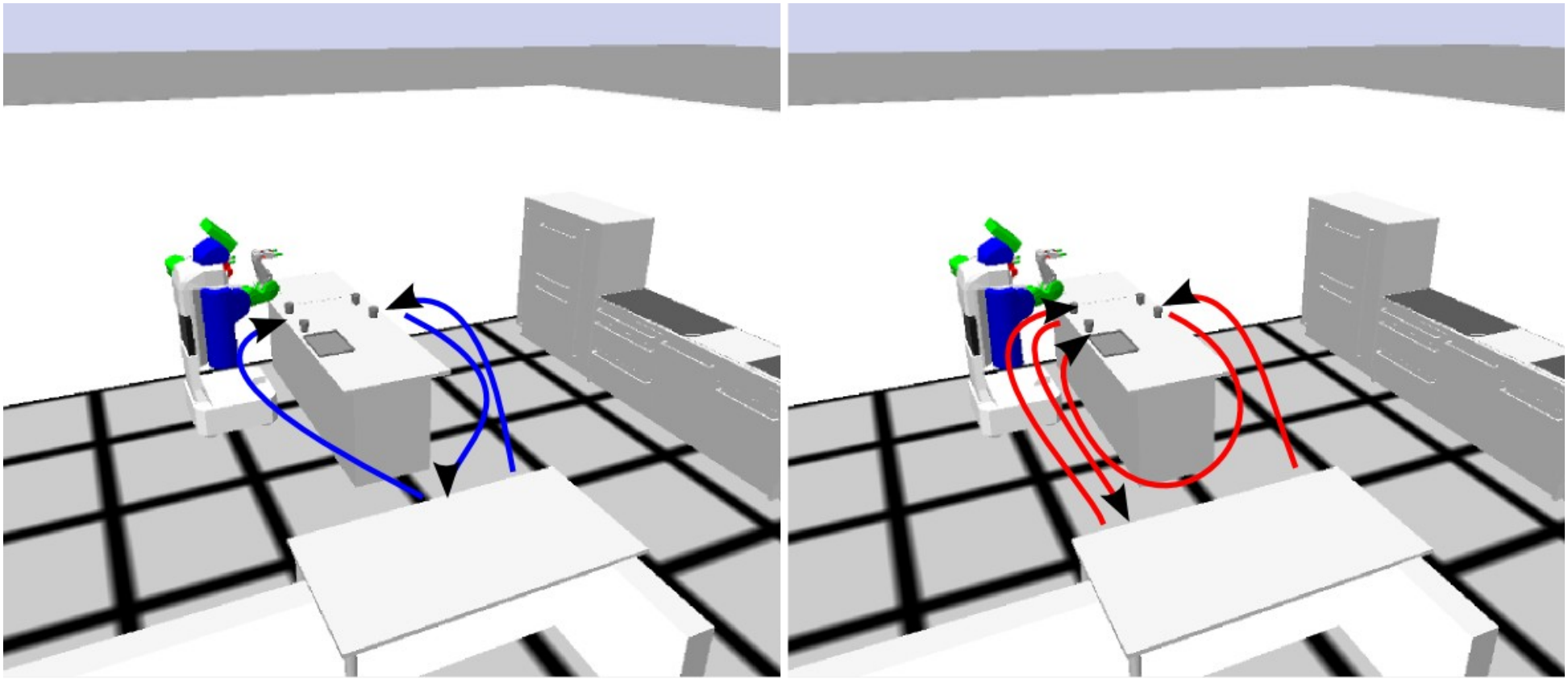
Simulation in projection





- transport 2..4 randomly placed cups to a destination [using the tray]
- projection doesn't report time; used distance travelled by base

Simulation in projection



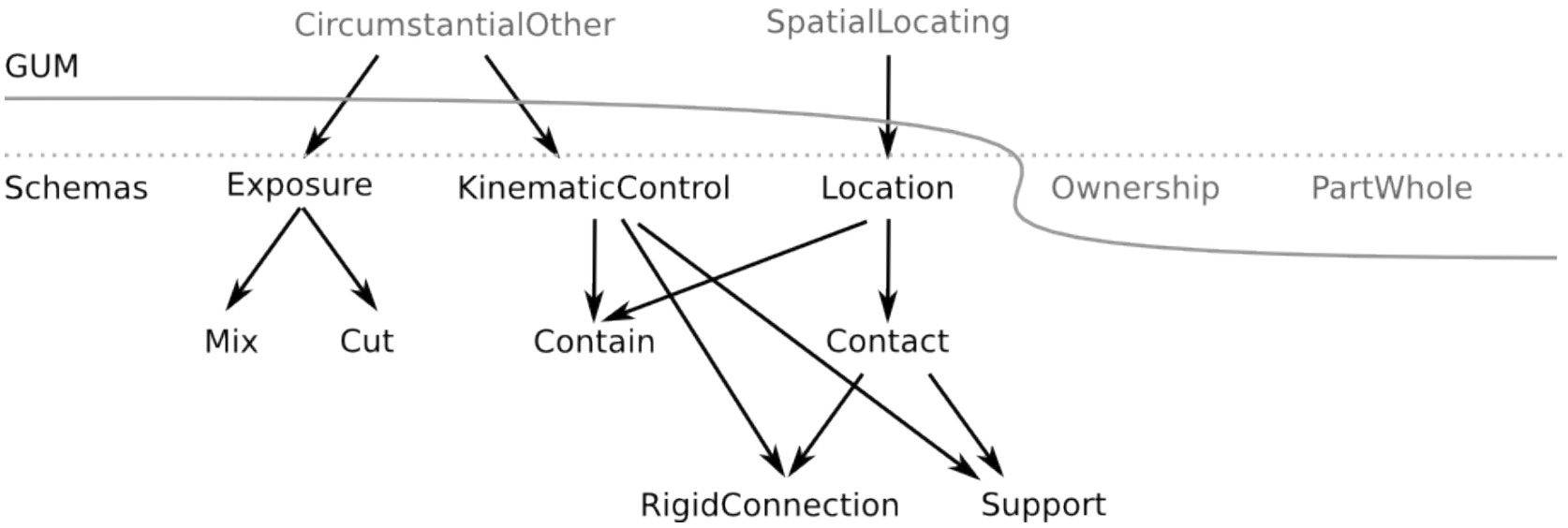
- the robot can put at most 3 cups on the tray
- need to carry 4 -> better to carry 2 then 2, rather than 3 then 1

II: Program building blocks

Schemas

- inspired by image schemas from cognitive linguistics
- schemas (as used here): “a pattern that abstracts from details for which we have efficient mechanisms of specification once given an embodiment and environment”
  - a belief that a functional relationship holds between entities in the world
  - at least one program to establish it
  - may have perceptual expectations which, upon failure, invalidate the schema

Schema Taxonomy

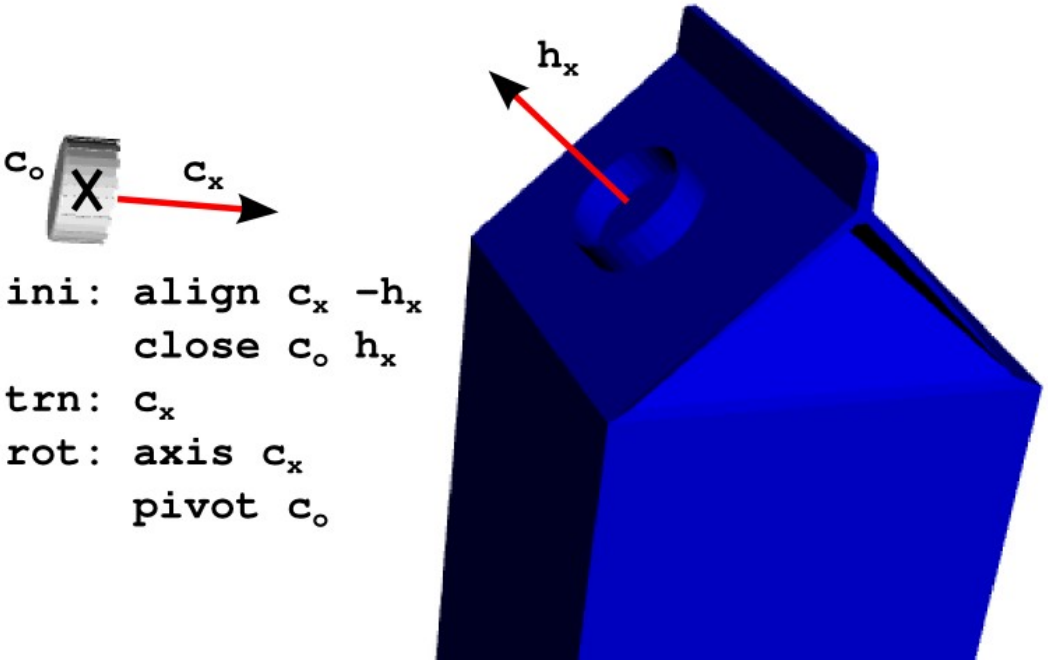


- use the GUM as an upper ontology
- schema roles are (descended from) GUM object properties

Schema definitions

- Ownership: a :possessor owns a :possessed item (and may use it)
- Location: a :locatum is at :placement (e.g. :on) relative to :relatum
- KinematicControl: an :actee will follow moves of an :instrumental
- Exposure: an :ingredient is exposed to :phenomenon
  - by using an :instrumental (e.g. frying pan)
  - driven by an :enablement (e.g. stove)
  - exposure may have a :manner
  - subcategories of exposure (Mix, Cut) define :result(s)

Schematic action primitives

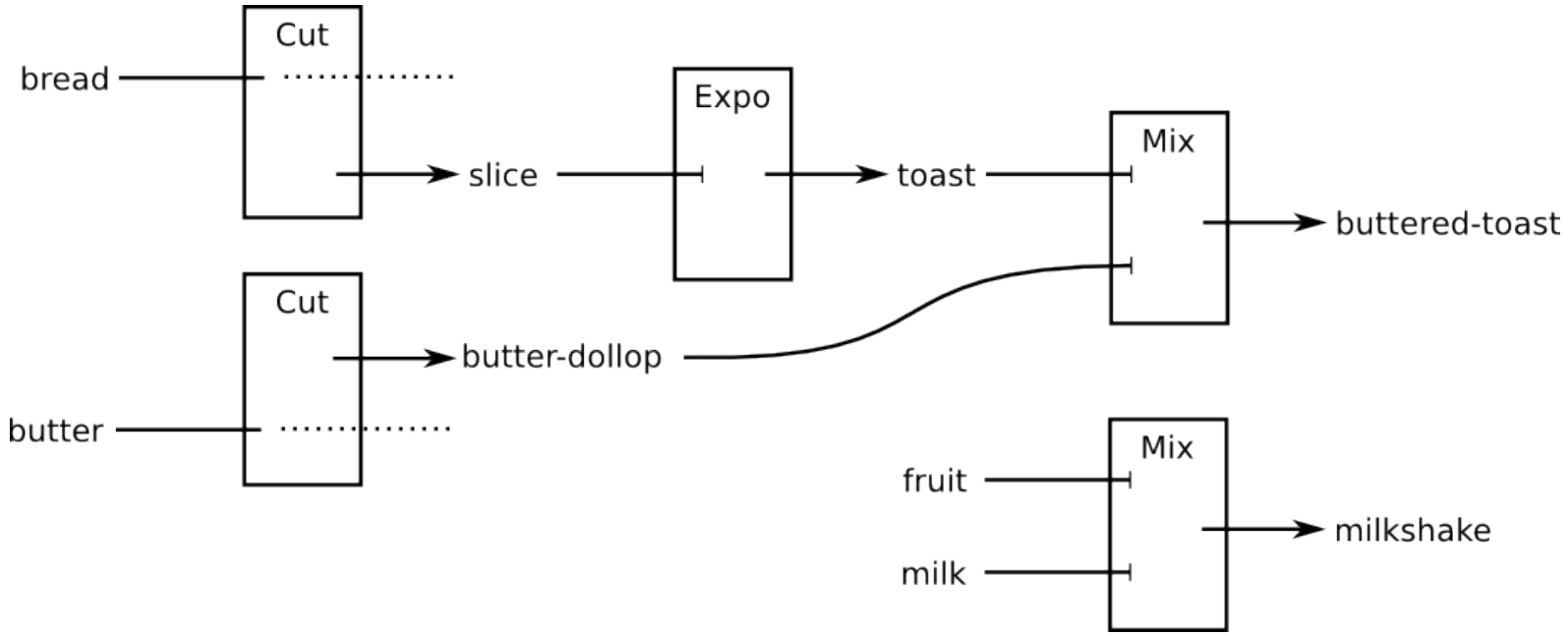


- Wait
- ApplyMovement: ensure an :actee follows a semantically defined trajectory
- Transport: move a kinematically controlled object to a location
- Transfer: change the kinematic controller of an :actee

Establishing schema: Exposure

```
def establish-exposure
      (actee, phenomenon, instrumental, enablement, manner):
    establish-ownership(robot,
                        (actee instrumental enablement))
    if valid-instrument(phenomenon, instrumental, enablement):
        unless enablement:
            enablement=instrumental
        establish-kincon(instrumental, actee)
        establish-contact(instrumental, enablement,
                          purpose=phenomenon)
        if enablement==robot:
            applymovement((a motion (to phenomenon)
                                    (who enablement)
                                    (with instrumental)
                                    (how manner)))
        else:
            applymovement((a motion (to activate)
                                    (who enablement)))
            expected-perception((a perception (who enablement)
                                              (state active)))
        assert(exposure, actee, phenomenon, instrumental,
                         enablement, manner)
    else:
        fail((wrong-instrument phenomenon instrumental
                               enablement))
```

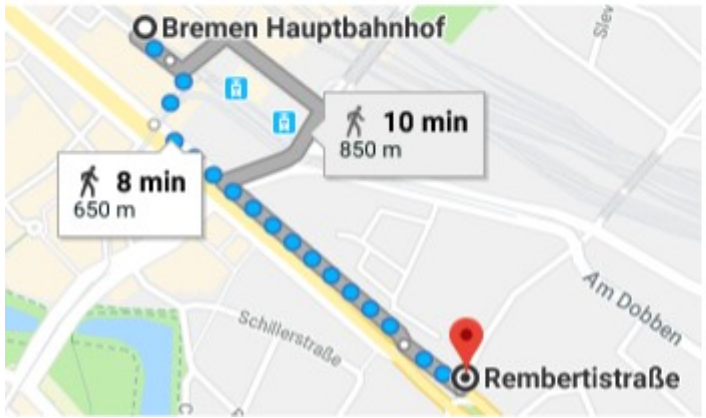
Building programs from schemas



- program as a sequence of schemas to establish/undo
- object arguments may be specific individuals or classes
- use similarity metrics to find replacements for primary items

III: Embodied contextualization

“Near” is relative ...



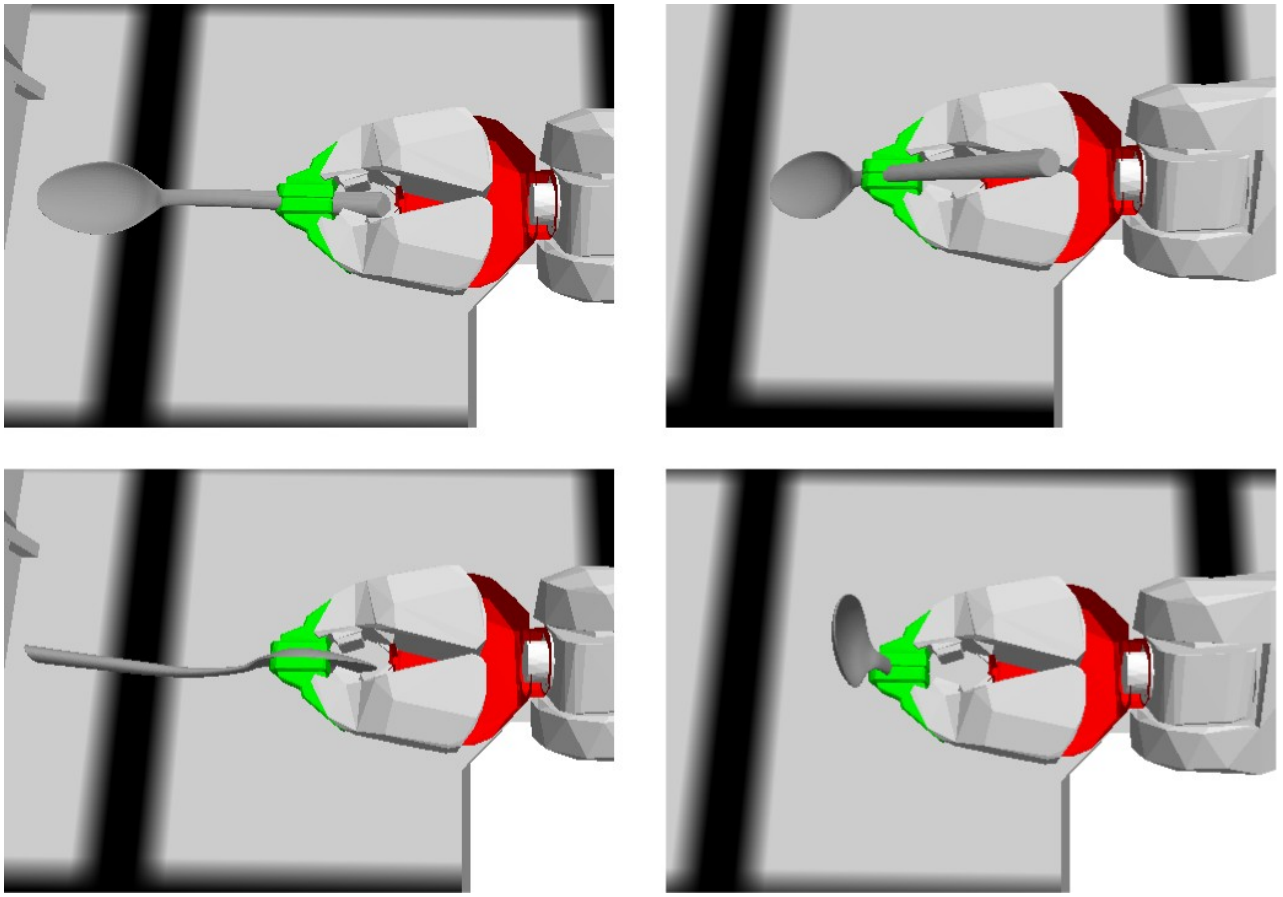
- ... but to what?

“Put the plates on the table”



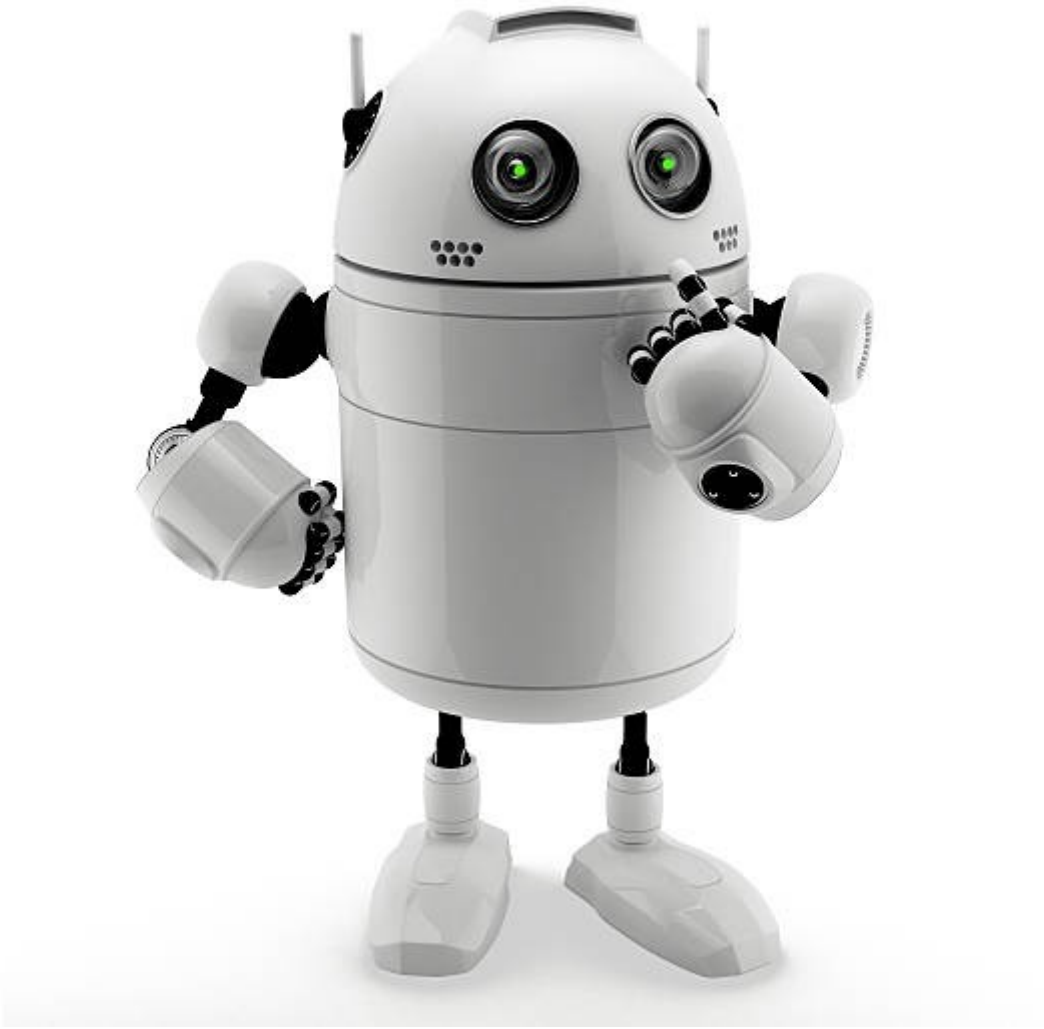

- only some interpretations make sense, and not always

“Pick up the spoon”



- several ways to do so; which is appropriate?

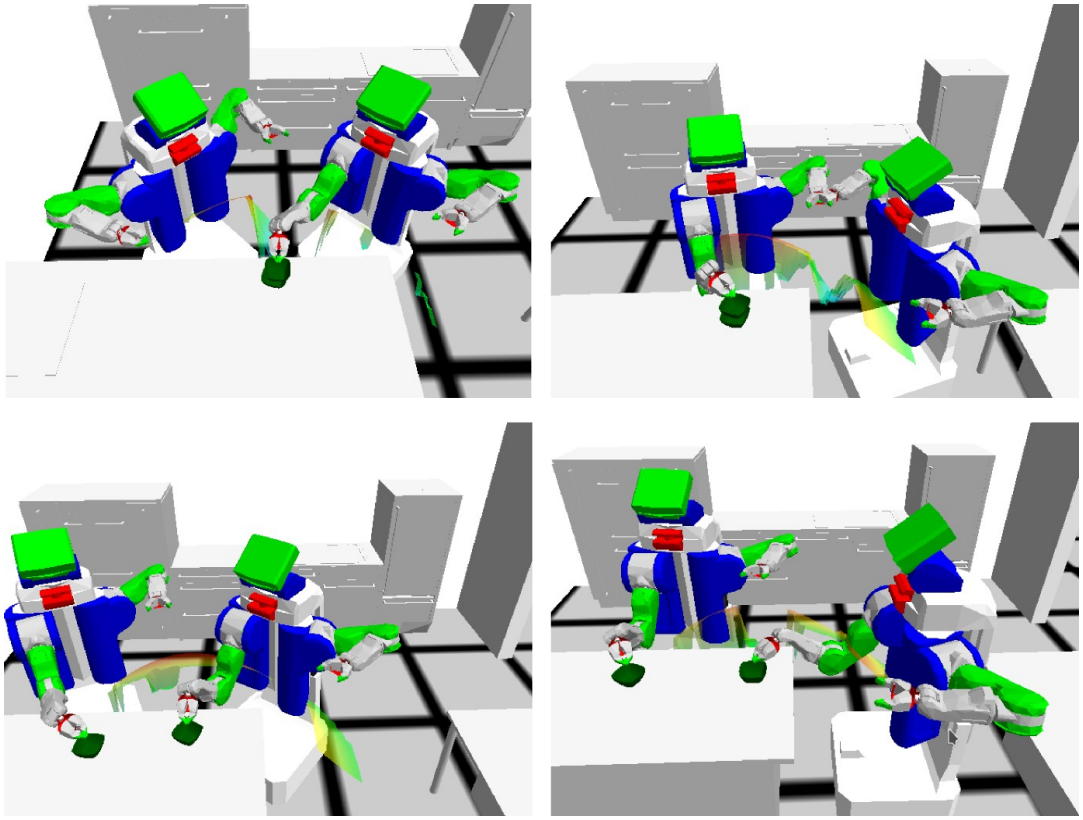
What to do?



Meaning ~ semantics + pragmatics

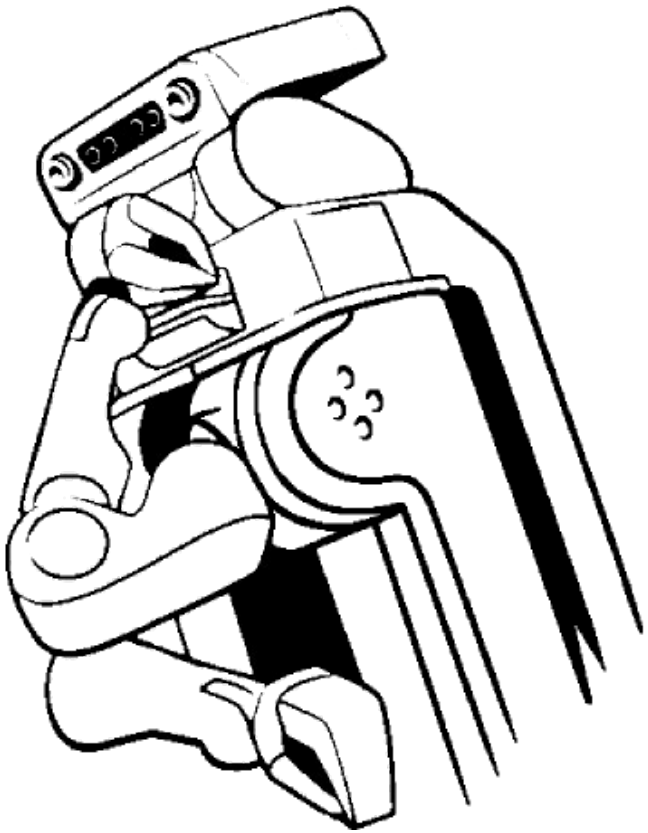
- context helps to pick out interpretations
- in particular, we look at task context: some behaviors are less appropriate for particular tasks
- possible to implement a rule-based disambiguation based on context
  - ... but probably a bad idea to hard-code though
  - doesn't scale
  - doesn't understand
- reflexive rules should come after understanding grounded in embodiment
  - simulation helps here

“On the table”: for eating vs. washing up



- simulate a “main” task for the robot, followed by an “enabled” task some other agent(s) must perform
- the two tasks should “work well” together
- simulate table setting (with leaving plates stacked vs. leaving them separate)
- simulation shows why leaving plates stacked to eat from them is a bad idea

Thank you!



References

- B. K. Bergen et al., Embodied Construction Grammar in Simulation-based Language Understanding, tech. report, 2003.
- J. Bos et al., A spoken language interface with a mobile robot, Artificial Life and Robotics 11, vol. 1, 2007.
- M. Eppe et al., Exploiting deep semantics and compositionality of natural language for Human-Robot Interaction, IROS, 2016.
- C. Matuszek et al., Learning to parse natural language commands to a robot system, Springer, 2013.
- D. K. Misra et al., Tell me Dave: context-sensitive grounding of natural language to manipulation instructions, RSS, 2014.
- D. Nyga et al., Everything robots wanted to know about housework (but were afraid to ask), IROS, 2012.