Plan-based Action and Activity Control for Everyday Manipulation

Michael Beetz and the IAS Group

Intelligent Autonomous Systems
Technische Universität München

ICRA Workshop
Motion Planning for Physical Robots
Shanghai, May 2011

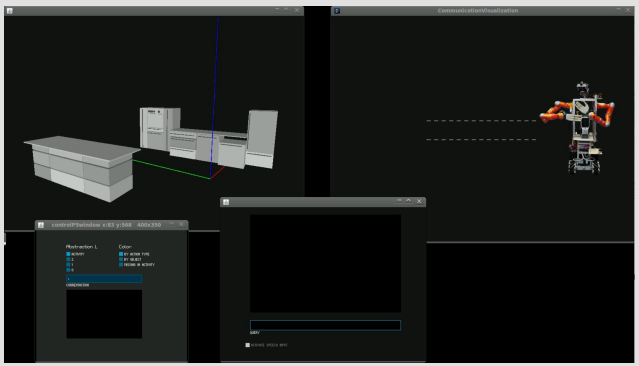
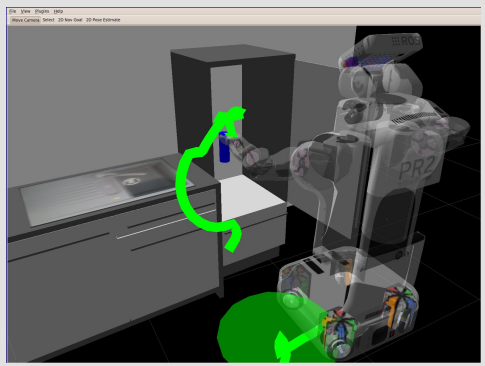
Intelligent Autonomous Systems

Robotic Roommates Making Pancakes

Our Vision:

Cognitive robots that

- autonomously perform human-scale everyday manipulation tasks and
- extend their repertoire of such by acquiring new skills using information resources intended for human use.

Plan Generation	Plan Execution	Plan Explanation
		

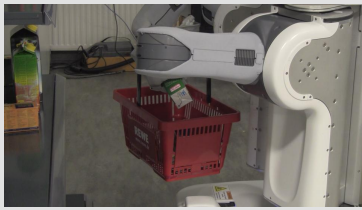
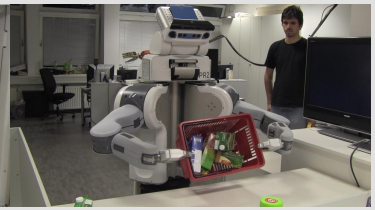
Robotic roommates making “Weisswürste”

Understanding by building

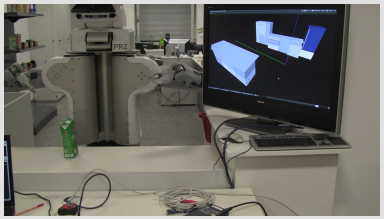
Shopping & cleaning up

1. shopping with basket



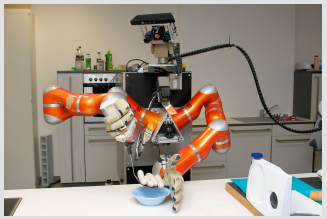



2. clean up according to organizational principles

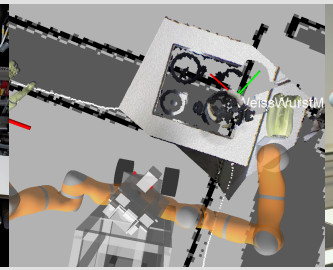
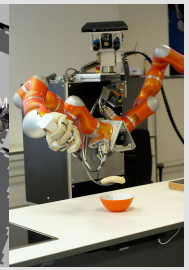



Making “Weisswürste”

1. putting “Weisswürste” into pot




2. fishing “Weisswürste”

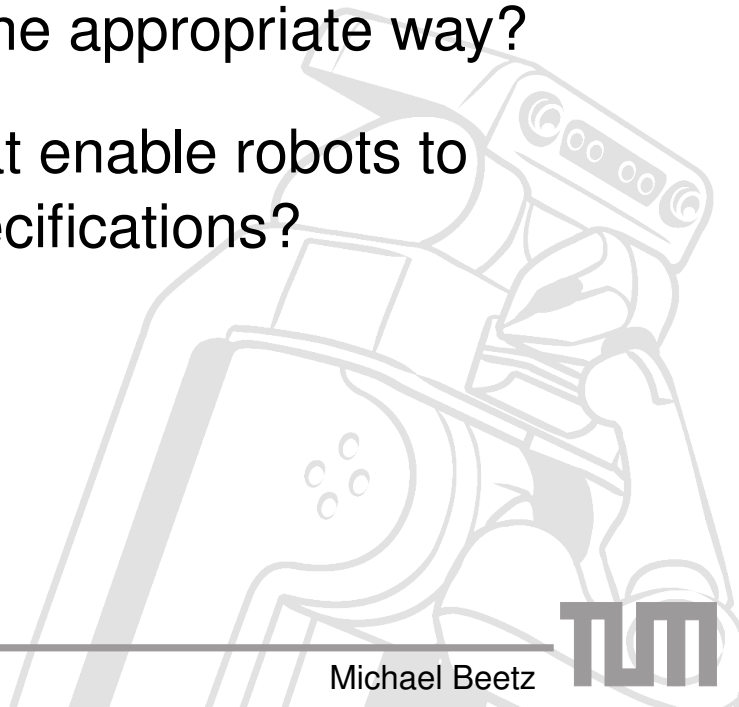




3. cutting bread

Research Questions

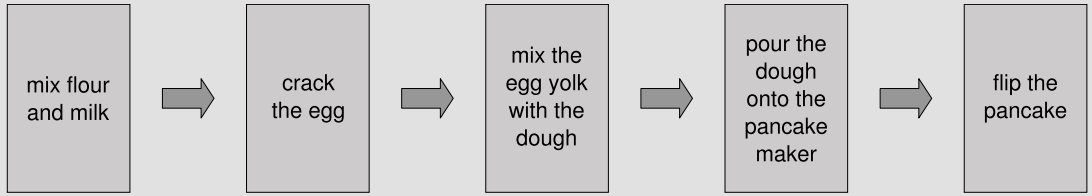
- how is it possible that we as humans get instructions such as
  - make pancake using a pancake mix
  - flip the pancake
  - push the spatula under the pancake

  and perform the intended tasks in the appropriate way?
- What are computational models that enable robots to perform such naturalistic action specifications?

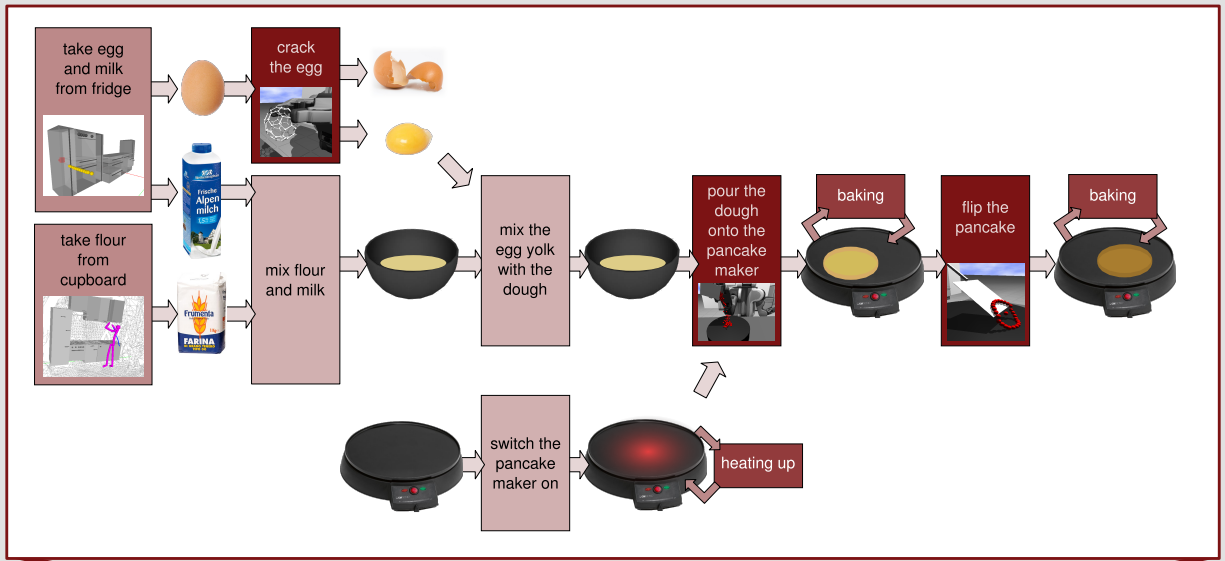


Everyday Manipulation Activities

From:

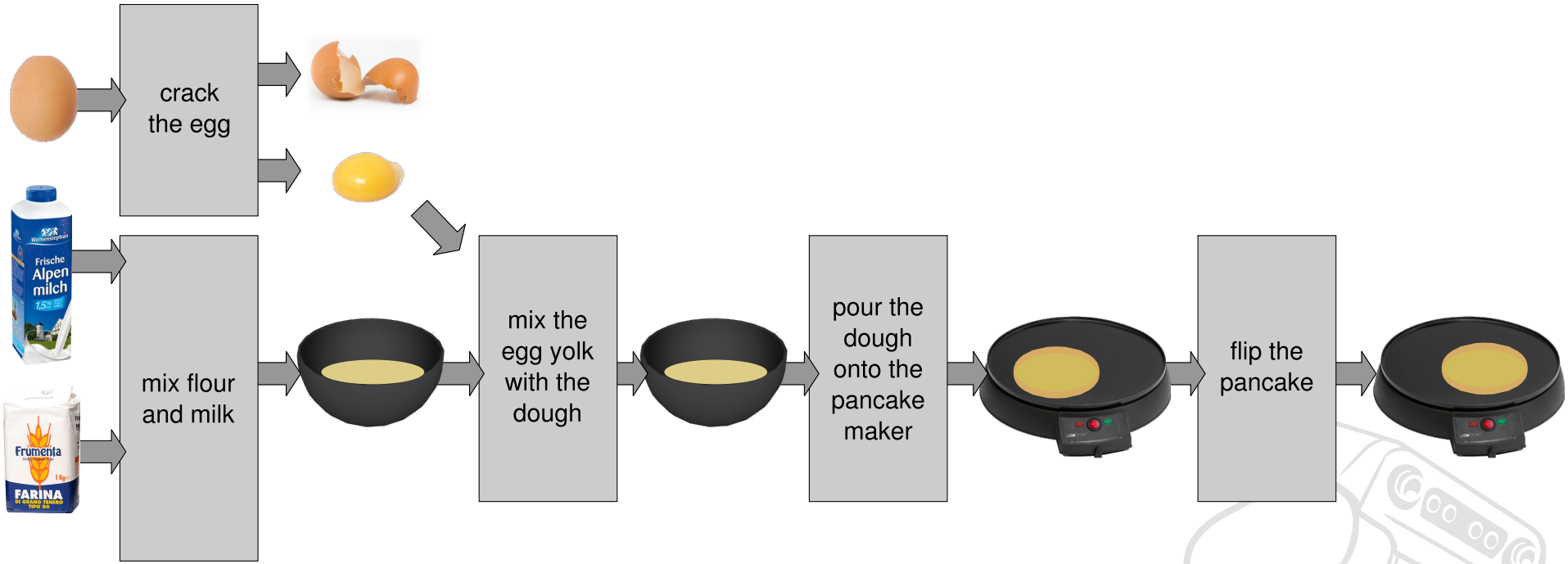


To:



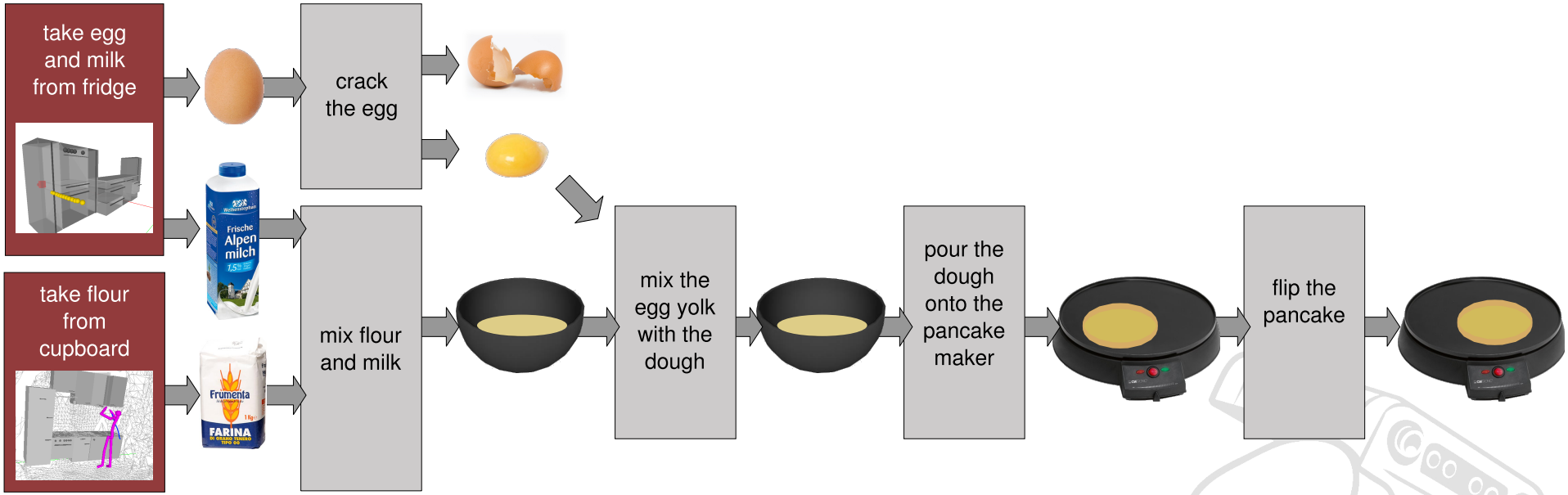
Robot Plan Generation

Inferring required objects



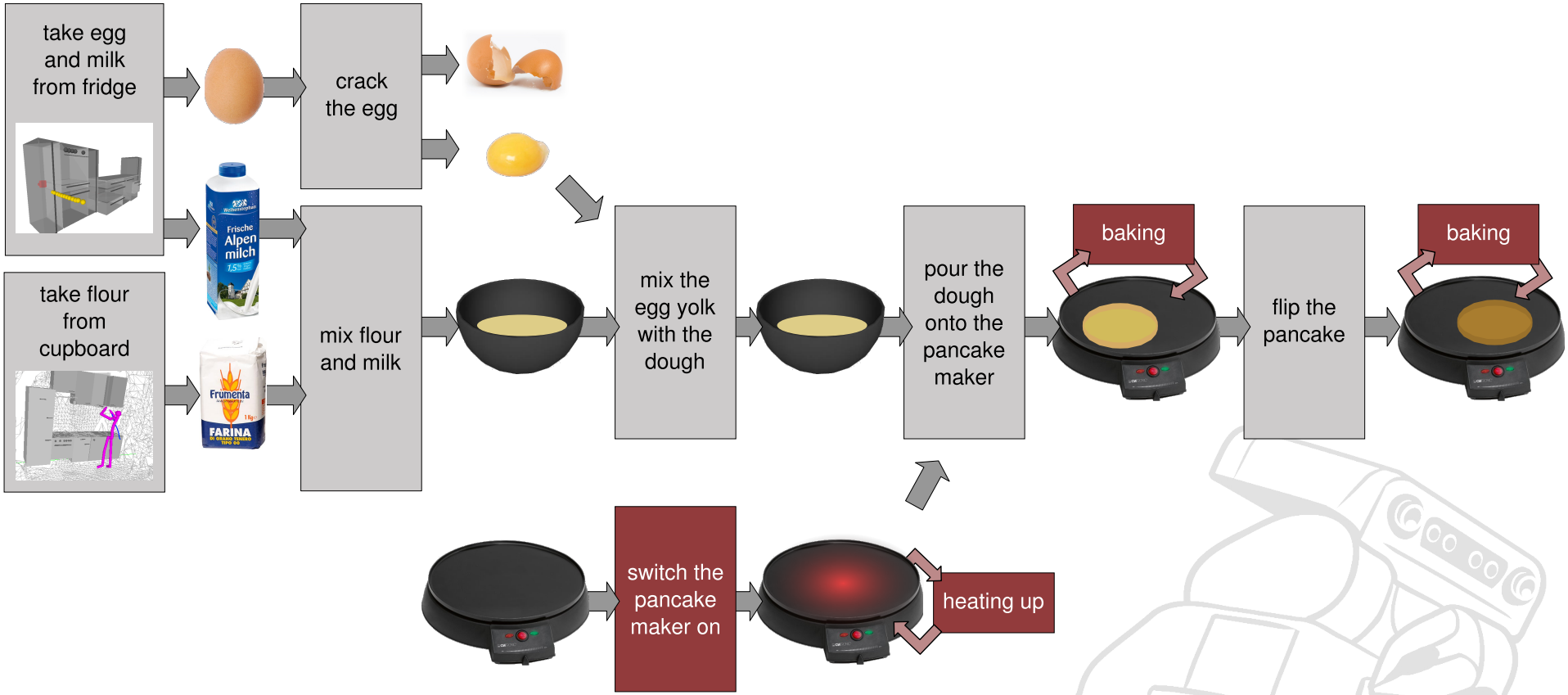
- Predict intermediate objects: Projection of action effects
- Make sure they can be recognized: Download object models

Inferring where objects can be found



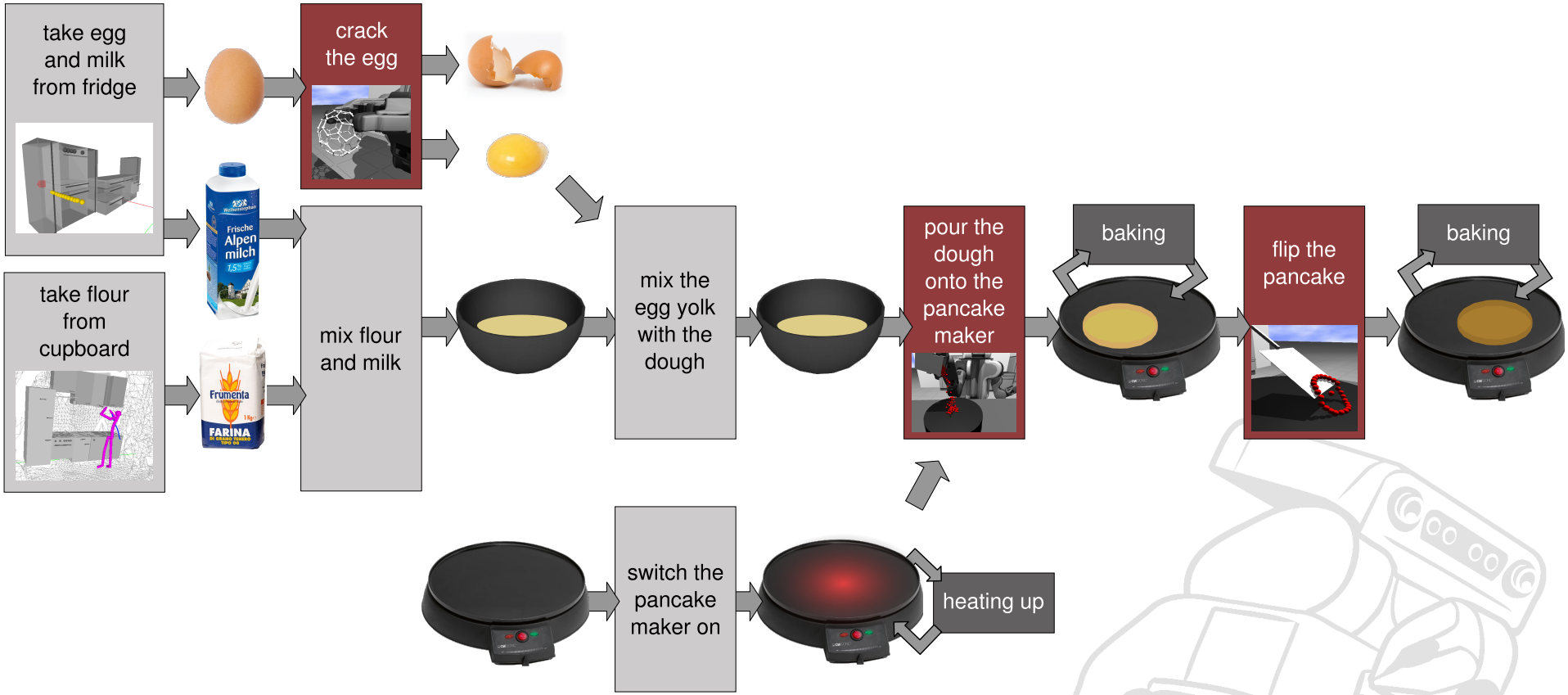
- Infer where to look for objects: Semantic environment models
- Add actions to fetch them: Articulation models and observations of humans

Planning with processes



- Check if results are correct: Action effect axioms
- Add required actions: Planning with actions and processes

Simulation for parameterizing actions



- Choose action parameters: Physically simulate action effects

Everyday Manipulation Actions

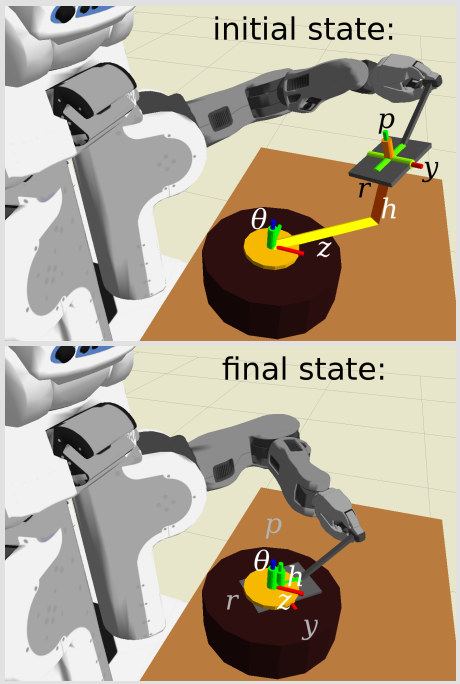
From: "push the spatula under the pancake"

To:

Task Specification

Define cylinder+rpy coordinates $\langle\langle\theta, x, z\rangle, \langle r, p, y\rangle\rangle$ between spatula and pancake.

- move in horizontal direction towards pancake ($x = 0$)
- keep spatula horizontal ($r = 0, p < 10^o$)
- maintain contact force with pancake baker ($force(z) > 1N$)



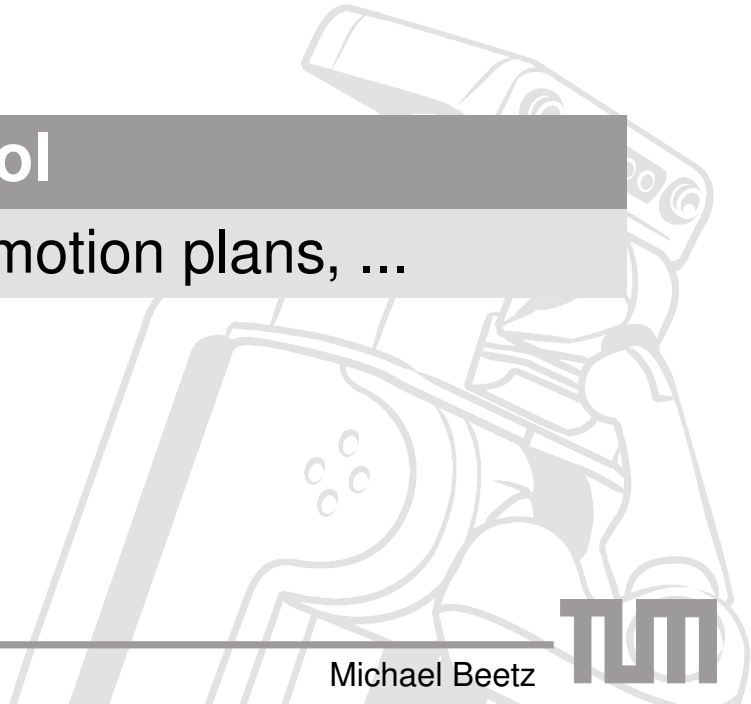
3 Layer Architectures

Abstract (symbolic) layer

“push the spatula under the pancake"

Low-level control

controllers, dynamic motion primitives, motion plans, ...



3 Layer Architectures

Abstract (symbolic) layer

“push the spatula under the pancake"

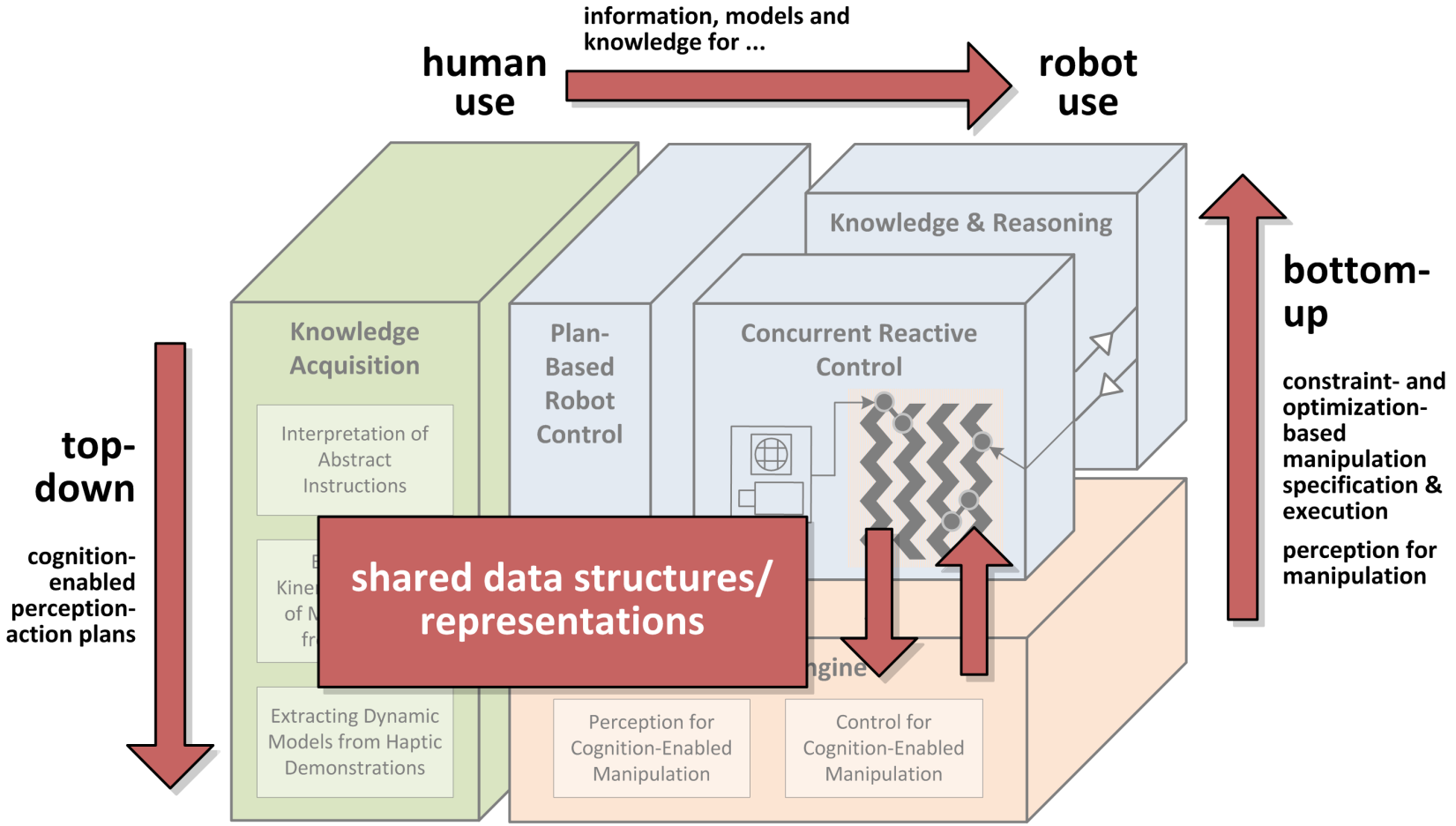
Plan execution layer

bridges between high- and lowlevel control

Low-level control

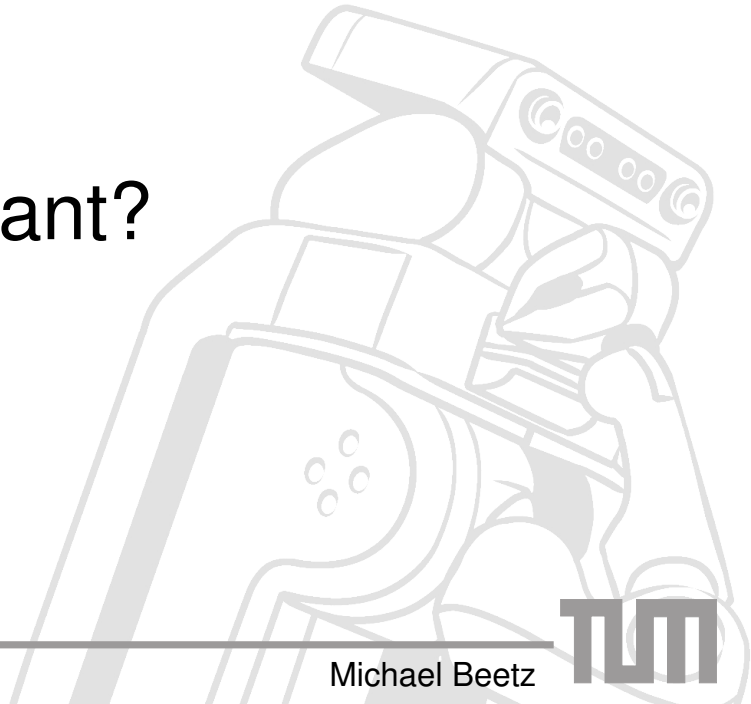
controllers, dynamic motion primitives, motion plans, ...

Shared Representation



Three fundamental research questions:

- How to specify actions?
- How to get what is meant from what is specified?
- How to execute what is meant?



Constraint-based Specifications

(perform (an action
(attribute/constraint$_1$ value$_1$)
...
(attribute/constraint$_n$ value$_n$)

vague specification	**effective specification**
push the spatula under the pancake	push the spatula under the pancake such that ◦ you can lift the pancake safely, ◦ don't damage the pancake, and ◦ don't push the pancake off the oven

Naturalistic Action Specifications

What is meant

what is specified	
push the spatula under the pancake	
flip the pancake carefully	
put the pancake mix down	
take the spatula	

Naturalistic Action Specifications

What is meant

what is specified	what is meant
push the spatula under the pancake	push the spatula under the pancake such that ◦ you can lift the pancake safely, ◦ don't damage the pancake, and ◦ don't push the pancake off the oven
flip the pancake carefully	flip the pancake such that ◦ undesired side effects are avoided and ◦ the robot stops if they might happen
put the pancake mix down	put the pancake mix down where ◦ it is visible and reachable when needed and ◦ it does not hinder the overall activity
take the spatula	take the ◦ handle of the spatula ◦ such that you can perform precision control of the blade of the spatula

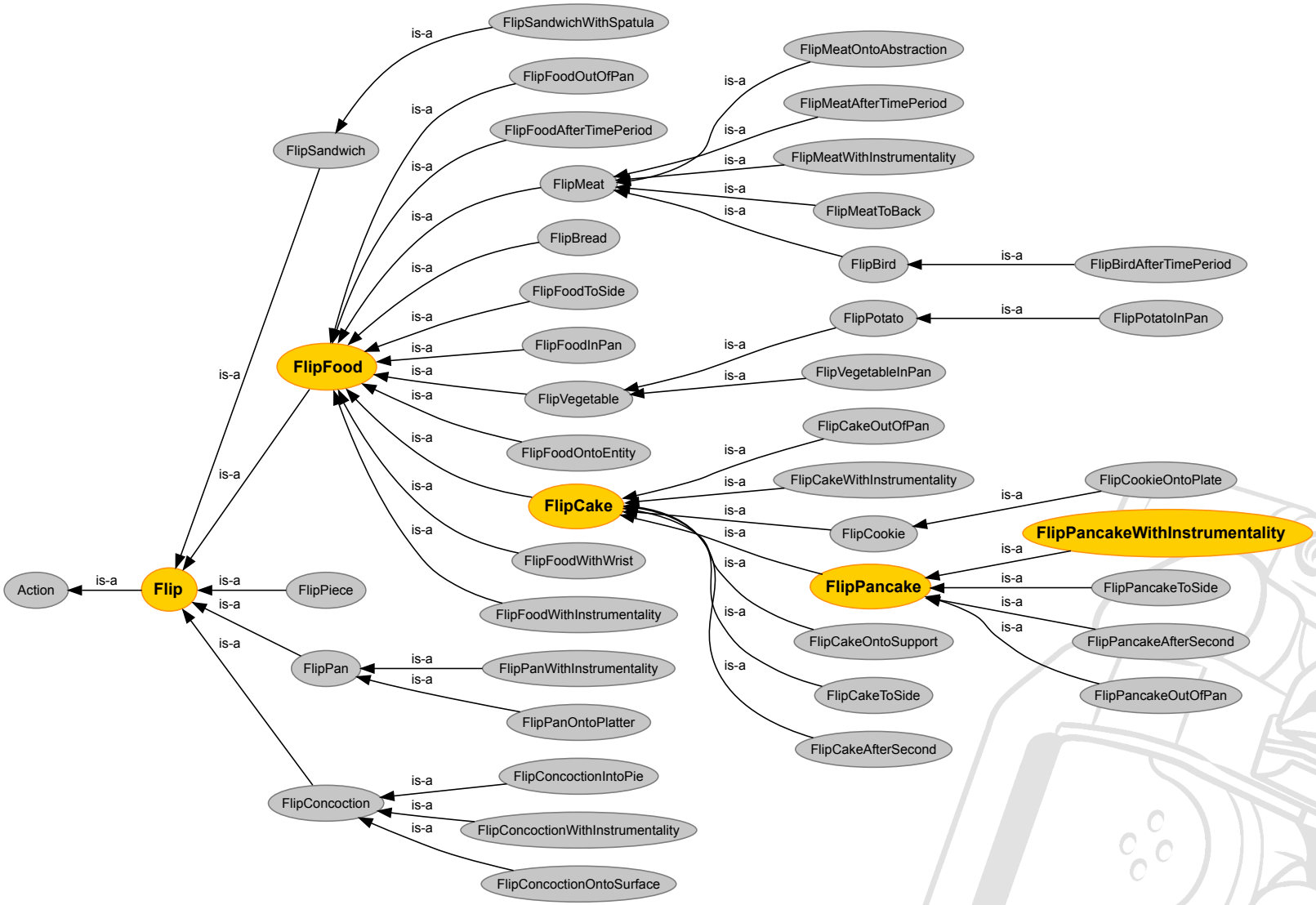
Building an action library

Mining instructions from wikiHow.com

- 273 Categories
- 8786 NL-Plans
- >130,000 sentences
- ≈53,000 relevant instructions
- ≈100 relevant action verbs
- most important: adding sth (> 7,900), Picking/Placing (> 4,900)
- top 15 action verbs more than 50% of actions

Flipping

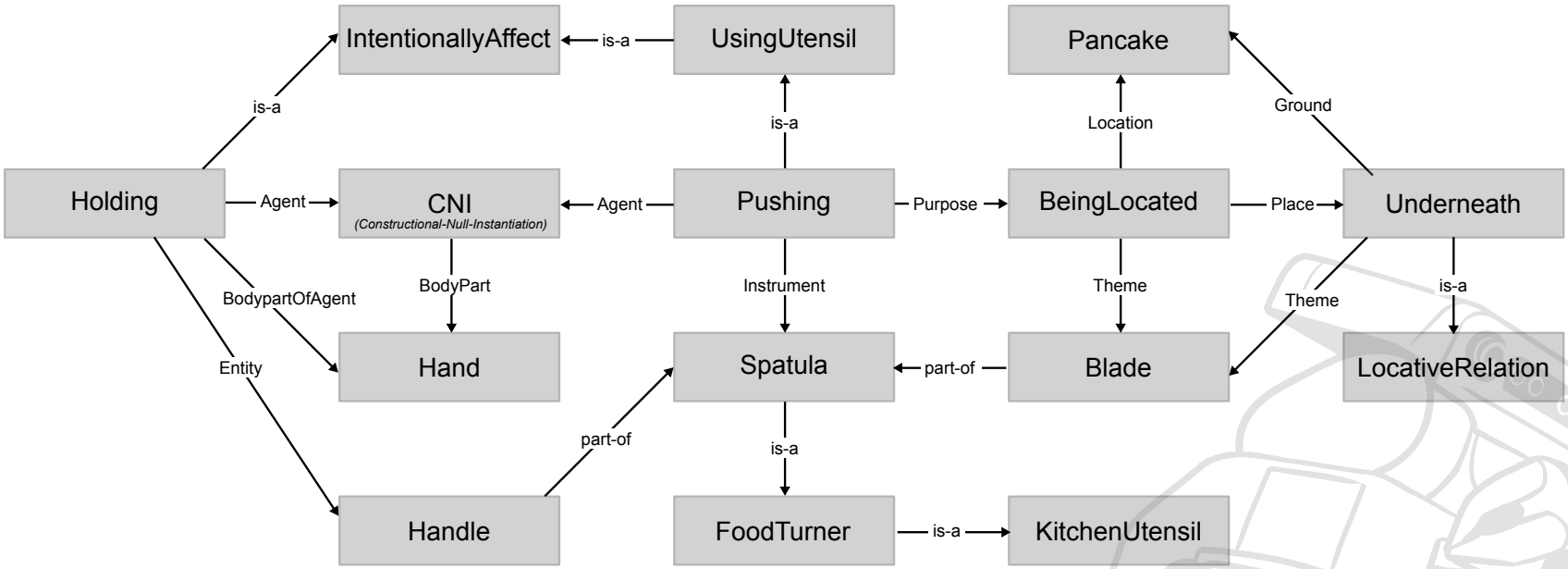
Building an action ontology



The semantic core of action verbs

Action Specification

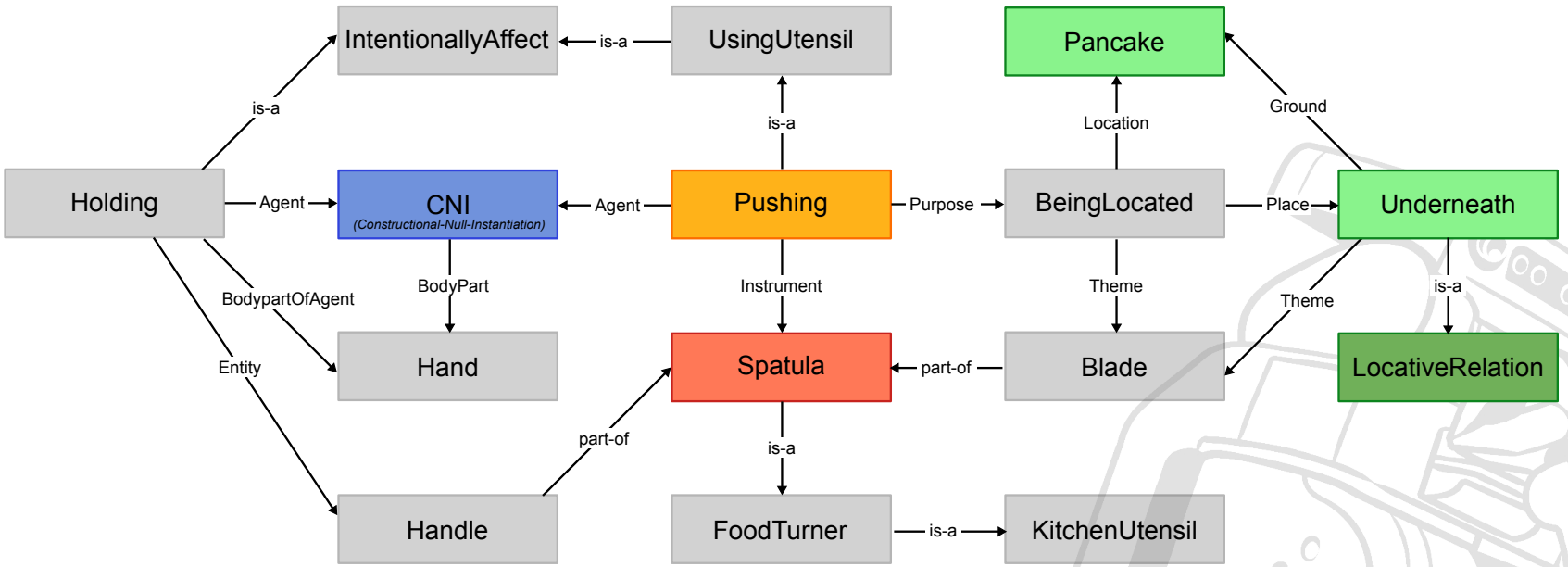
Semantic Core: Set of inter- and intraconceptual relations that constitute an abstract event type, assigning a semantic role to each entity that is affected by an action verb.



(thanks to Torsten Schubert)

The Semantic Core of Action Verbs

Action Specification



Take the spatula

Automatic completion of action verbs

```
(perform (an action
          (type grasp)
          (object (an object-part
                   (part-of spatula)
                   (type handle)))
          (desired-effect (and (grasp ?grasp-spec)
                               (succeeds (an action
                                          (type precision-control)
                                          (object (tip-of blade))
                                          (grasp ?grasp-spec)))))))
```

probabilistic completion

P(occurs(*ev*),
  type(*ev*,*type*),
  bodyPartOfAgent(*ev*,*Part*),
  entity(*ev*,*what*)
  | “push the spatula under the pancake”)

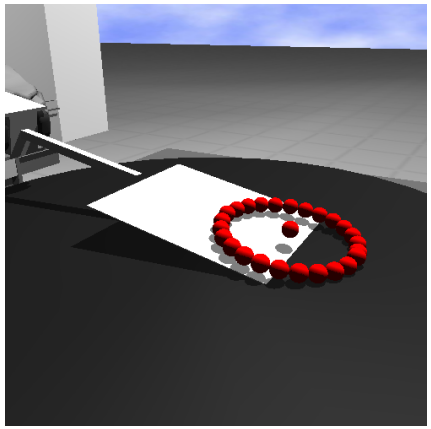
tuple with the highest probability:

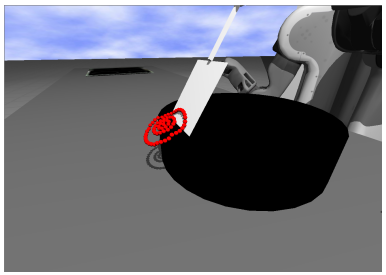
- *type* = holding
- *bodyPartOfAgent* = hand
- *entity* = handle

Push the spatula under the pancake

Consequence-based action parameterization

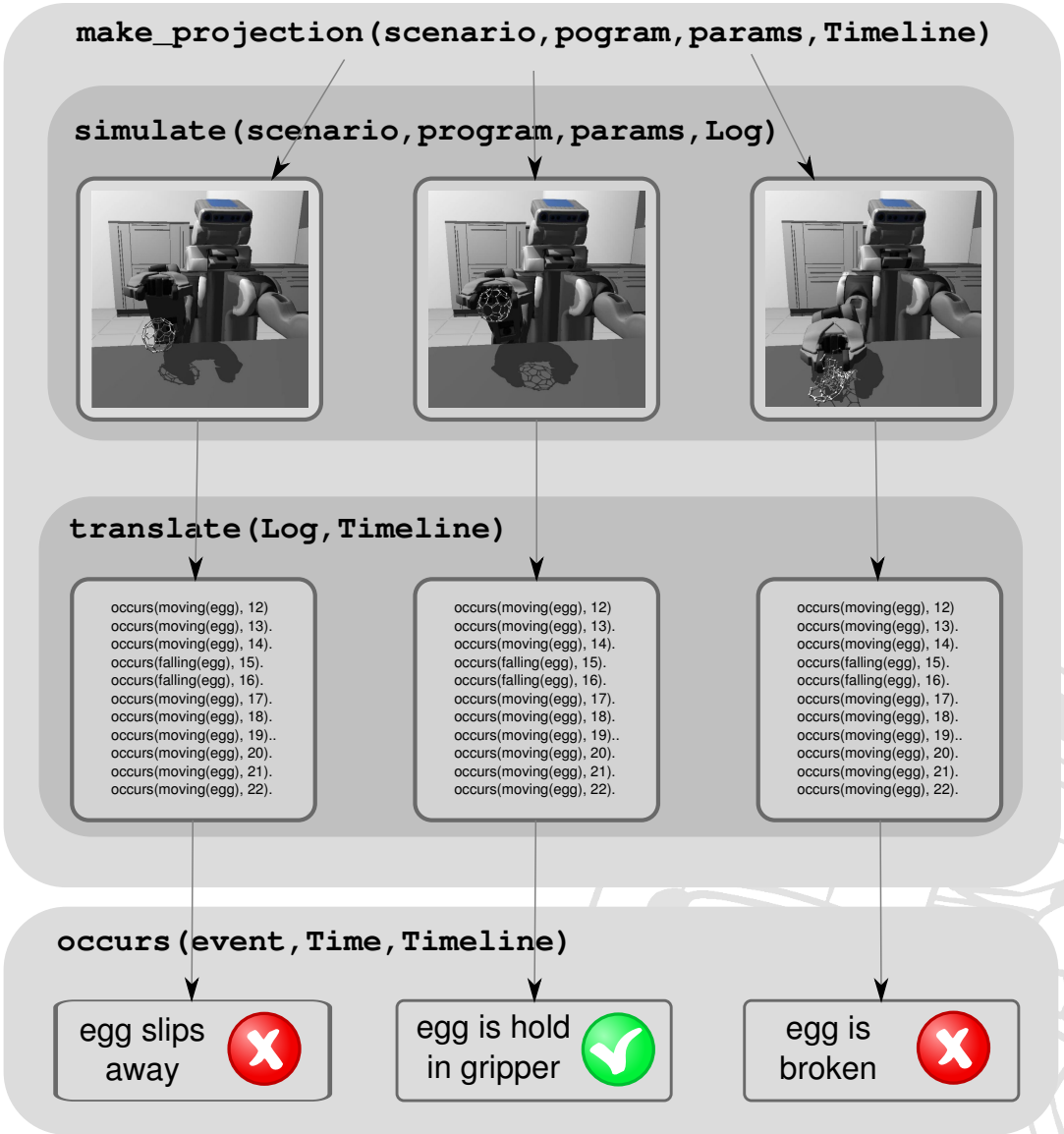
```
(perform (an action
           (type push)
           (object (an object-part
                     (part-of spatula)
                     (type blade)))
           (destination ?loc = (a location
                                 (under pancake)))
           (desired-effect (and (pose spatula ?loc)
                                (succeeds (an action
                                            (type lift)
                                            (object pancake)
                                            (starting-pose ?loc)))))
           (undesired-effects (and (damaged pancake)
                                   (on pancake counter)))))
```





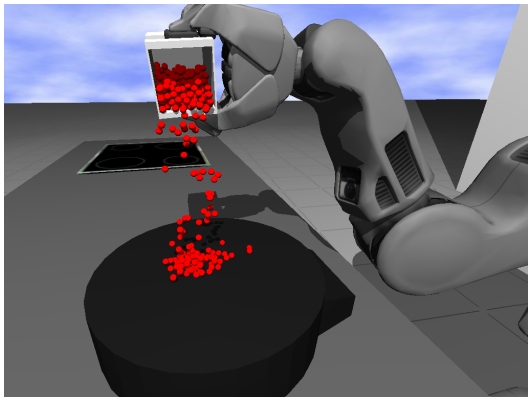
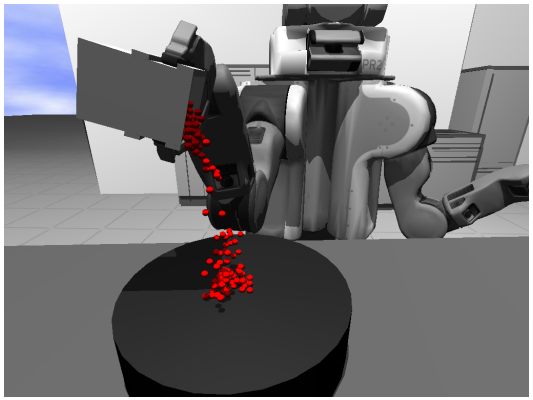
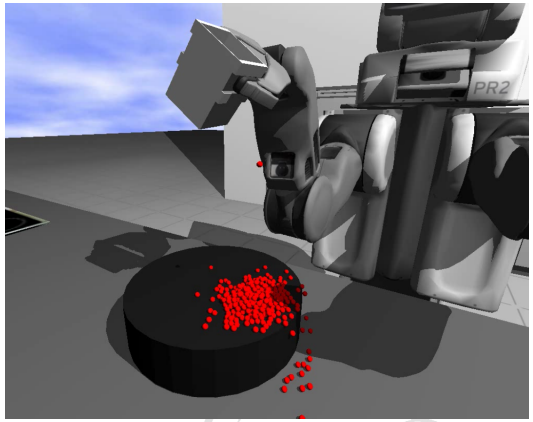
Temporal Projection Process

- **make_projection:**
  sample parameters
  - **simulate:**
    setup simulator
    run simulation
  - **translate:**
    ground predicates
    in logged
    simulations
- **evaluate:**
  events/fluents
  specialized predicates

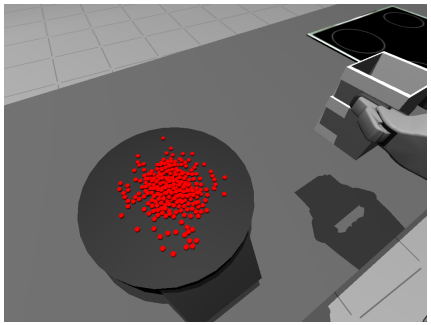
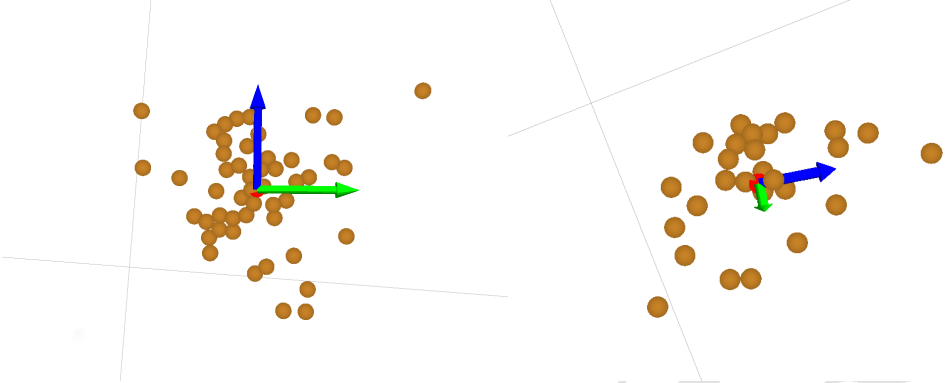


Example: Pouring pancake mix

- Parameters: position, time, angle
- Outcomes: number of particles on pan (spilled on table)

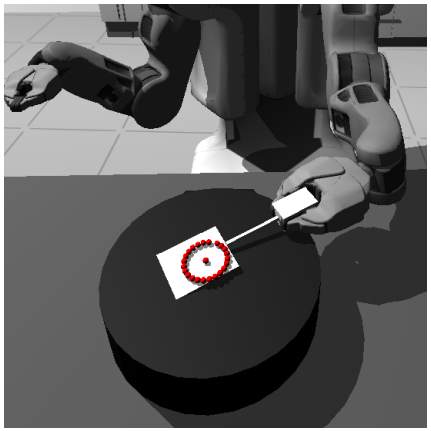
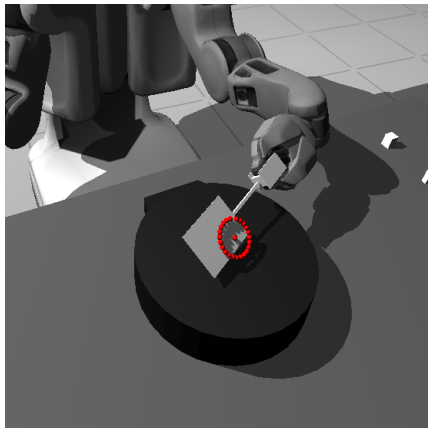
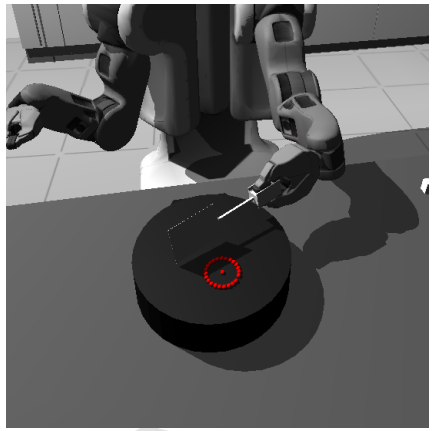




- Specialized predicates on particle sets: round/centered

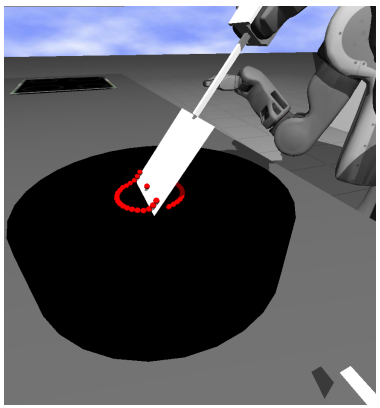



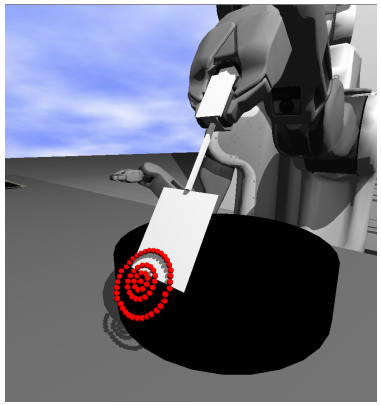
Example: Flipping a pancake

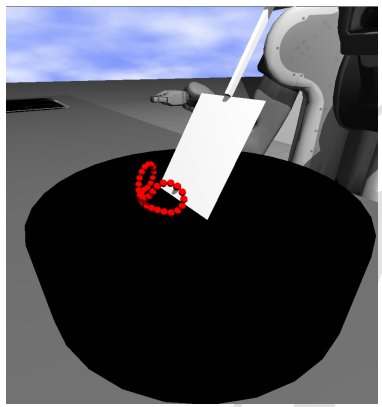
- Parameters: angle of spatula
- Outcomes: turned, not turned

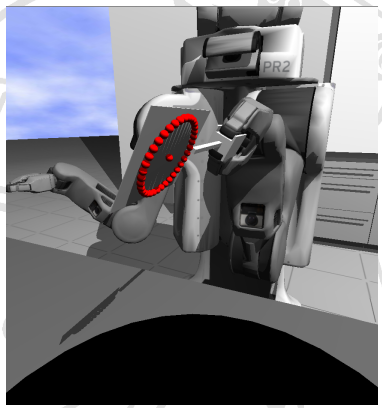




- Common failures:


break

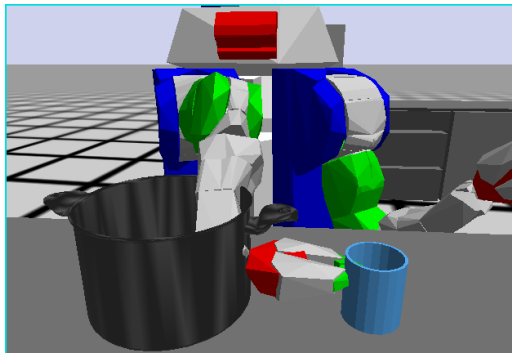

push off

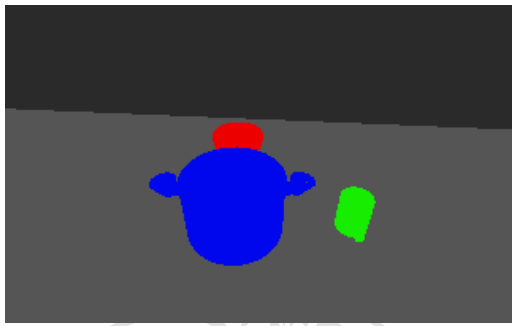

fold


stick on

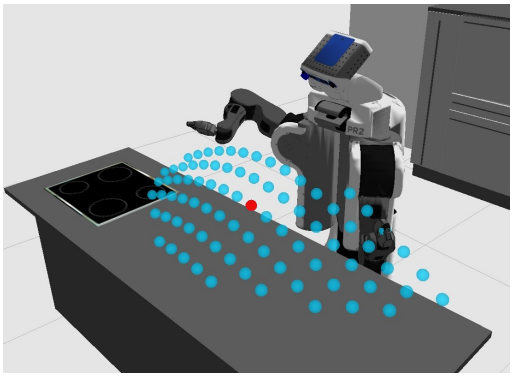
Put the pancake mix away

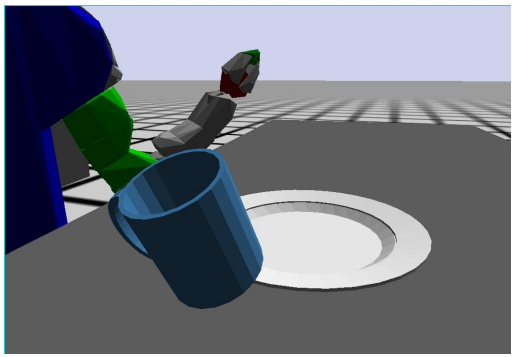
Exploiting underspecification





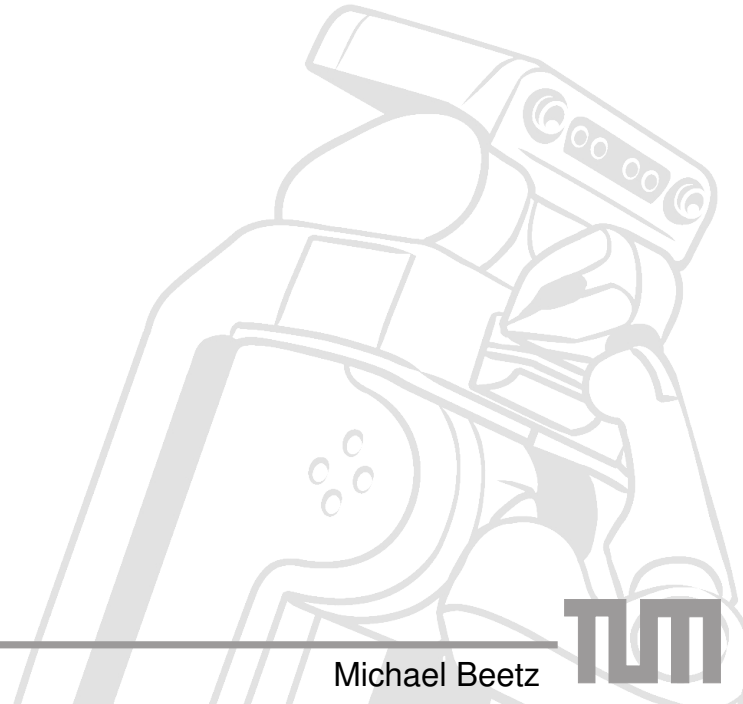
```
(perform (an action
                (type put-away)
                (object ?obj = (the object
                                    (type pancake-mix)))
                (destination ?loc = (a location
                                        (on counter)
                                        (stable ?obj)
                                        (reachable t)
                                        (visible-for James)
                                        (not (hindering (the activity
                                                            (type pancake-making))))))))
```





Inference algorithm

setof ?Pose On(Counter, ?Pose) ?Poses $\wedge$ member(?P, ?Poses)
$\wedge$ Pose(Cup, ?P) $\wedge$ stable(Cup)

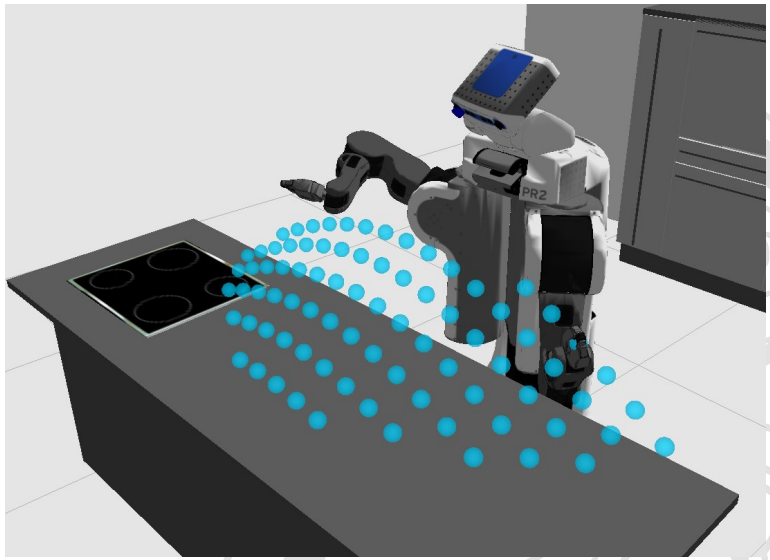


Inference algorithm

setof ?Pose On(Counter, ?Pose) ?Poses $\wedge$ member(?P, ?Poses) $\wedge$ Pose(Cup, ?P) $\wedge$ stable(Cup)

1. setof ?Pose On(Counter, ?Pose) ?Poses
2. member(?P, ?Poses)
3. Pose(Cup, ?P)
4. stable(Cup)

Create distribution for sampling poses

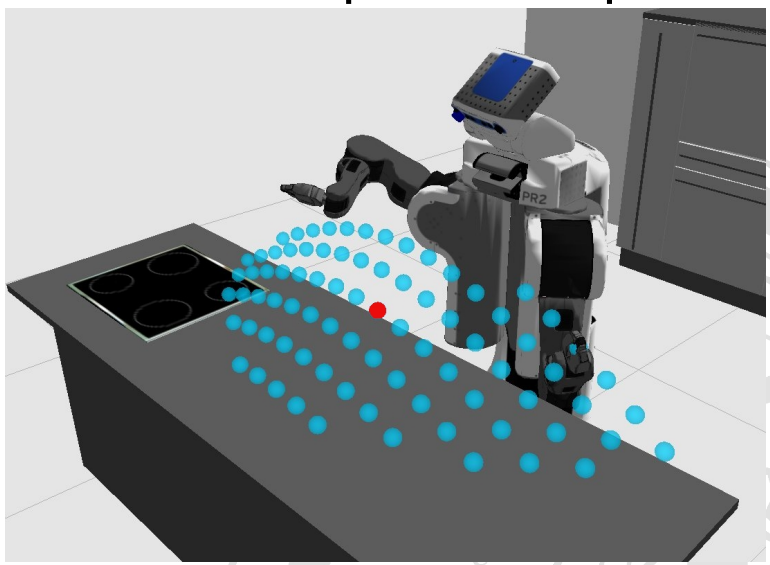


Inference algorithm

setof ?Pose On(Counter, ?Pose) ?Poses $\wedge$ member(?P, ?Poses)
$\wedge$ Pose(Cup, ?P) $\wedge$ stable(Cup)

1. setof ?Pose On(Counter, ?Pose) ?Poses
2. member(?P, ?Poses)
3. Pose(Cup, ?P)
4. stable(Cup)

Draw a pose sample

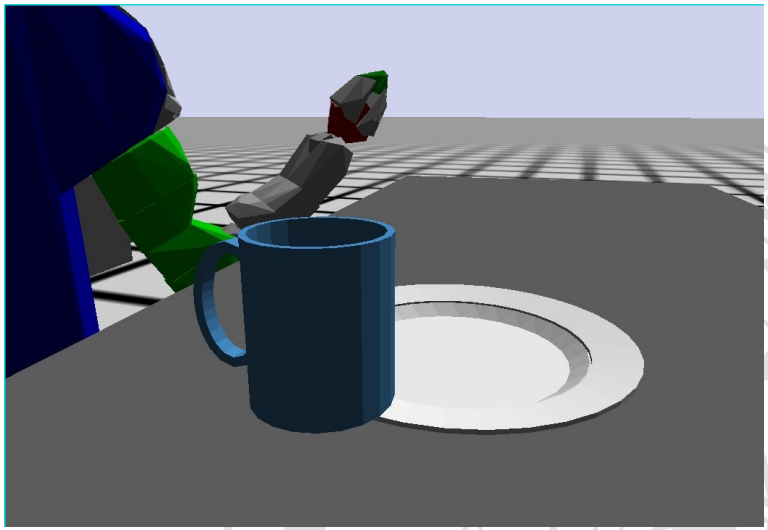


Inference algorithm

setof ?Pose On(Counter, ?Pose) ?Poses $\wedge$ member(?P, ?Poses)
$\wedge$ Pose(Cup, ?P) $\wedge$ stable(Cup)

1. setof ?Pose On(Counter, ?Pose) ?Poses
2. member(?P, ?Poses)
3. Pose(Cup, ?P)
4. stable(Cup)

Place the mug

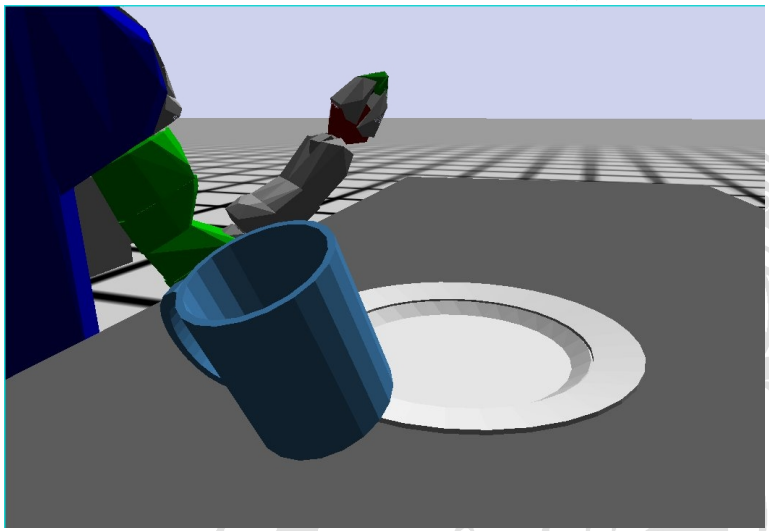


Inference algorithm

setof ?Pose On(Counter, ?Pose) ?Poses $\wedge$ member(?P, ?Poses) $\wedge$ Pose(Cup, ?P) $\wedge$ stable(Cup)

1. setof ?Pose On(Counter, ?Pose) ?Poses
2. member(?P, ?Poses)
3. Pose(Cup, ?P)
4. stable(Cup)

Simulate for 50ms, fail!

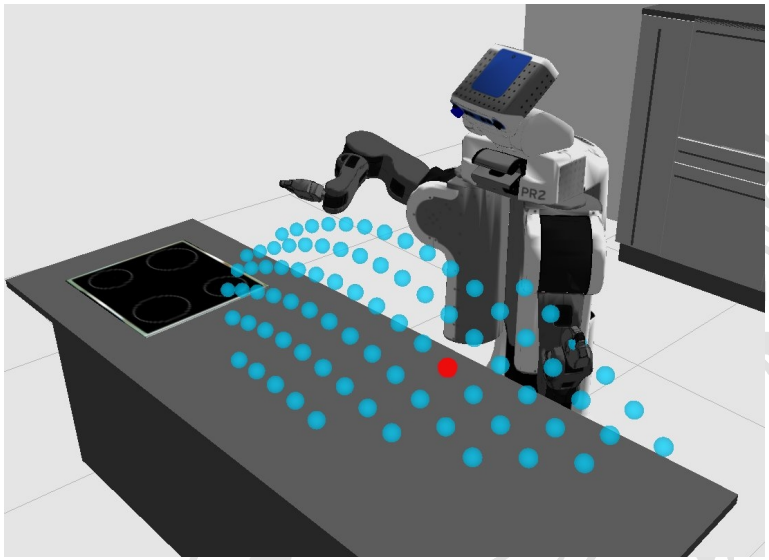


Inference algorithm

setof ?Pose On(Counter, ?Pose) ?Poses $\wedge$ member(?P, ?Poses) $\wedge$ Pose(Cup, ?P) $\wedge$ stable(Cup)

1. setof ?Pose On(Counter, ?Pose) ?Poses
2. member(?P, ?Poses)
3. Pose(Cup, ?P)
4. stable(Cup)

Backtrack, draw another pose sample

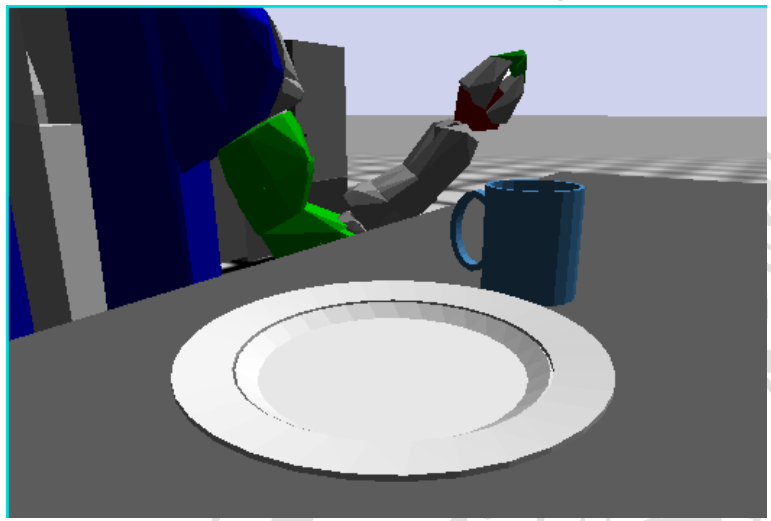


Inference algorithm

setof ?Pose On(Counter, ?Pose) ?Poses $\wedge$ member(?P, ?Poses)
$\wedge$ Pose(Cup, ?P) $\wedge$ stable(Cup)

1. setof ?Pose On(Counter, ?Pose) ?Poses
2. member(?P, ?Poses)
3. Pose(Cup, ?P)
4. stable(Cup)

Place the mug

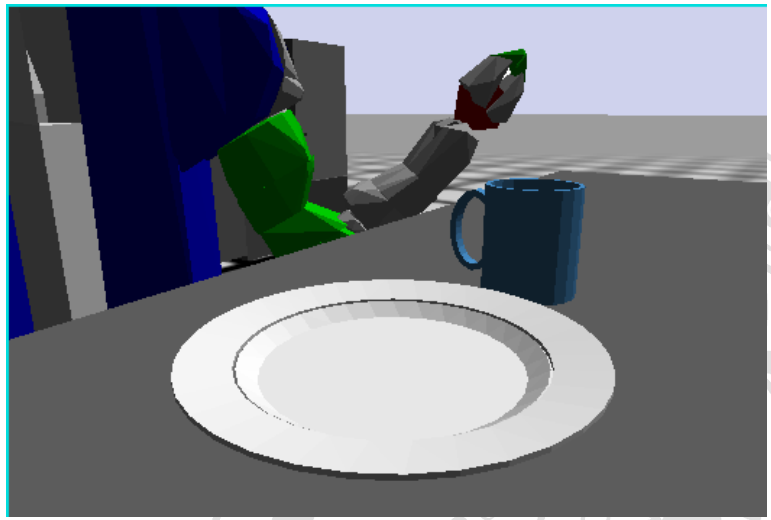


Inference algorithm

setof ?Pose On(Counter, ?Pose) ?Poses $\wedge$ member(?P, ?Poses) $\wedge$ Pose(Cup, ?P) $\wedge$ stable(Cup)

1. setof ?Pose On(Counter, ?Pose) ?Poses
2. member(?P, ?Poses)
3. Pose(Cup, ?P)
4. stable(Cup)

Simulate for 50ms, succeed!



Built-in Predicates of CRAM Reasoning

Stability	
contact(O_1, O_2)	Contact between objects
stable(O)	Stability of object
Visibility	
visible(P, O)	Object visible from pose P
occluding(P, O_1, O_2)	Object O_2 occludes object O_1
Reachability	
reachable(R, O)	Object O is reachable by robot R
blockingObjects(R, O, B)	B is the list of objects that “block” O

Temporal projection

(a location ...
(not-hindering (the activity (type (pick-up Cup1)))))

$\Downarrow$

projectPlan(PickUp(Cup1), ?TI) $\wedge$ bagof(?F, flawsInTimeline(?F), $\emptyset$)

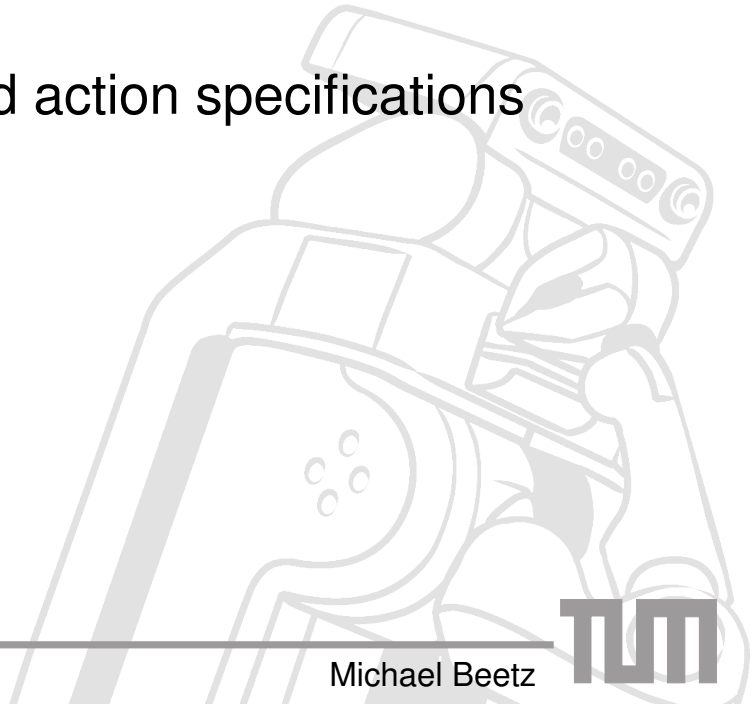
1. Execute plan in projection mode
2. Projection generates a timeline
3. Match pre-defined flaws on the timeline

Flip the pancake carefully

(perform (an action
(type flip)
(object (the object
(type pancake)
(on oven)))
(params argmin$_{params}$ P(fail(action(params),sit)
| execute(action(params),sit),
obj-acted-on(pancake),
props(sit))

Conclusions

- what is the problem? understanding by building
- take the human instructions as an indication for
  - what information is needed?
  - what not?
- shared representation
  - constraint- and optimization-based action specifications
- predictive decision making



Thank you for your attention

Questions?

PR2 Illustration by Josh Ellingson, Willow Garage