Approaches to Plan-Based Robot Control:
An Affordance-Based Approach

Joachim Hertzberg

Osnabrück University
and DFKI Robotics Innovation Center

Background: Affordance-Based Robot Ctrl.

- FP6 EU project MACS:
  Multi-Sensory Autonomous Cognitive Systems interacting with dynamic environments for perceiving and using affordances
  09/2004–11/2007
  http://www.macs-eu.org/

- E. Rome, J. Hertzberg, G. Dorffner (Eds.):
  Towards Affordance-based Robot Control.
  Springer (LNAI vol. 4760), 2008

- C. Lörken, J. Hertzberg:
  Grounding Planning Operators by Affordances.
  in: Proc. Intl. Conf. Cognitive Systems (CogSys), 2008, pp. 79-84

Affordances

J.J. Gibson, 1979

The Ecological Approach to Visual Perception

Action possibilities in the environment in relation to the actor

- "directly" "picked-up" sensor info. induces functions/utilities of an object/entity for the animal
- functions describable (externally) by abstract features related to physical properties of the animal:
  sittable (for a human)
  ➥ knee-high, horizontal, stable support, minimal size
- Same object/entity affords different actions for different animals

Conclusions about Affordances à la Gibson

- Perception of functions is based on **features** related to the physical properties of the animal.
- The feature set related to a function works like a **matched filter** across various object categories.
- The ability to perceive affordances, i.e., to perceive **functions of entities** in the world, enables more possibilities for action:
  An animal (or agent) could even guess what to do with entities that it has never before perceived.

Affordance Representation

Caution!
Here we deviate from Gibson's original ideas!

A. Chemero, M.T. Turvey: Gibsonian Affordances for Roboticists.
Adaptive Behavior 15(4): 473–480 (2007)

Data structure (for individual agent!)

⟨cue descriptor, behavior descriptor, outcome descriptor⟩

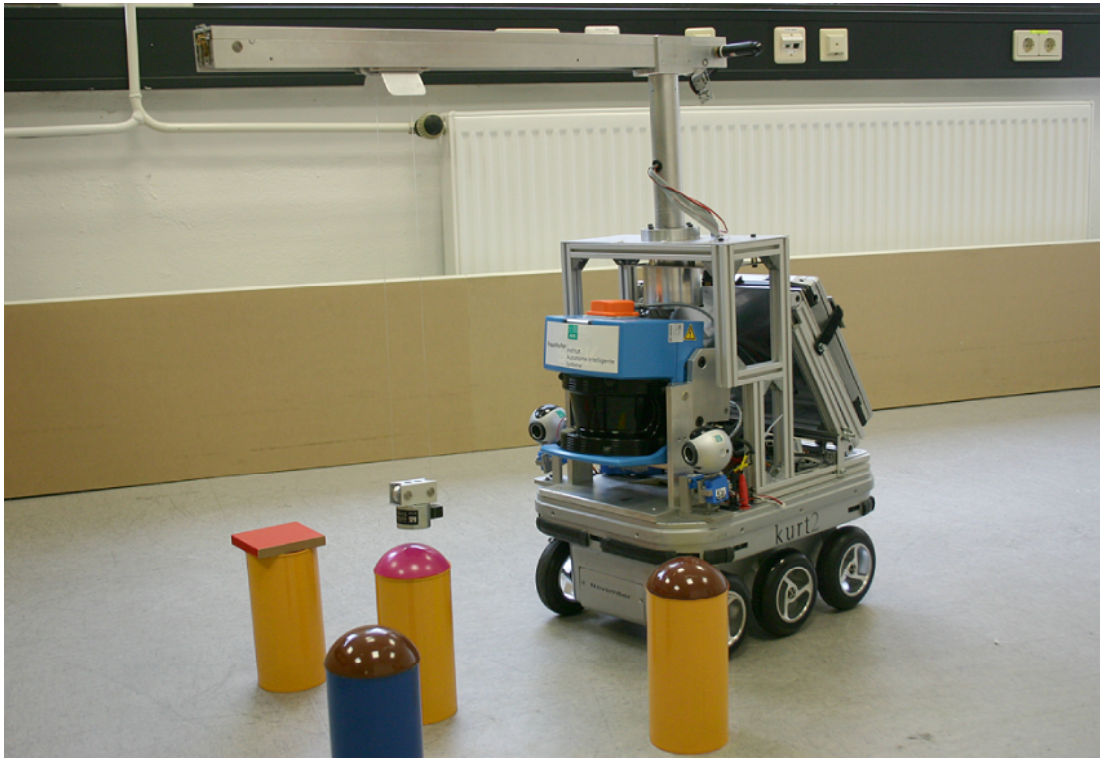
where

cue descriptor: entity representation, containing pairs of sensor attributes and associated value ranges

behavior descriptor: reference to robot behavior – reactive or high-level – plus optional set of behavior parameters

outcome descriptor: analogous to cue descriptor (sensor attributes / value ranges)

The MACS Demo Robot



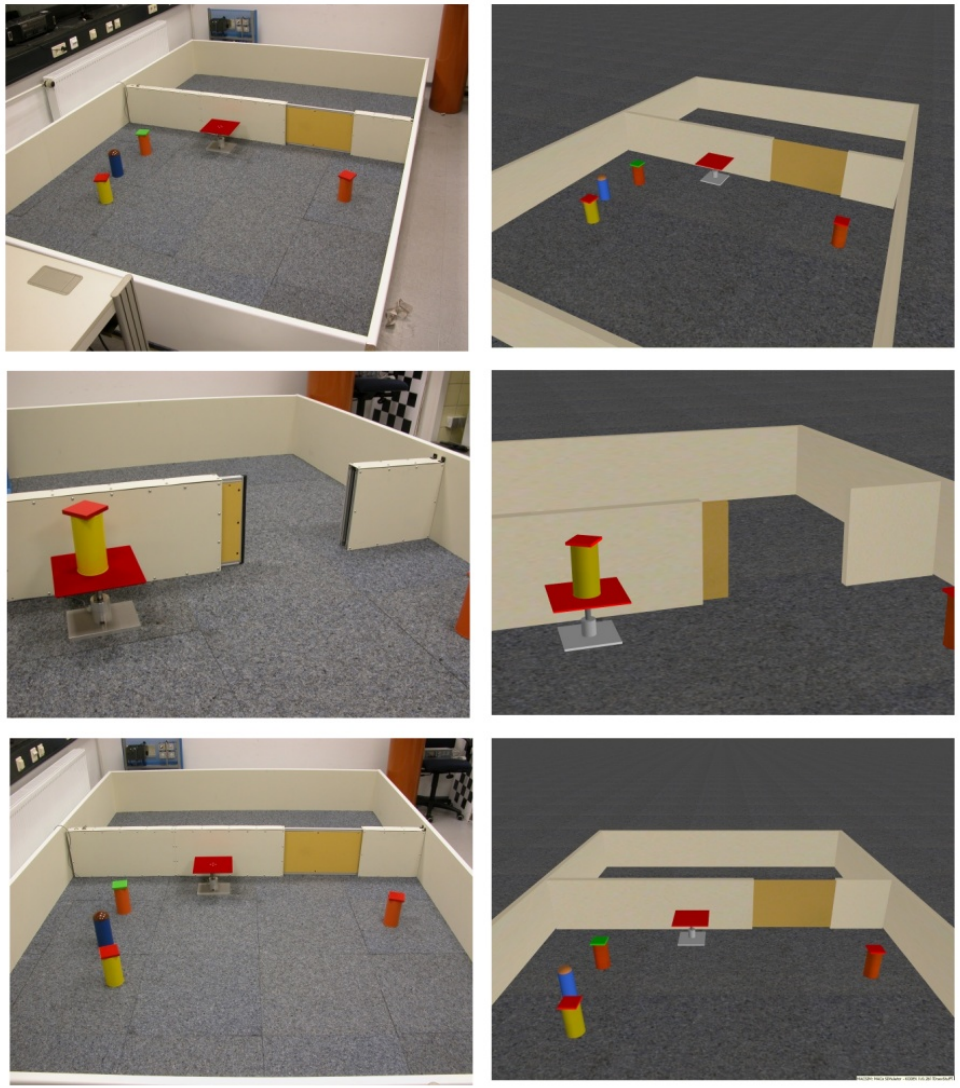
Sensors

- 2 cameras
- 3D Laser scanner
- weight sensor
- more …

Actuators

- 6 wheels, 2 drive motors
- 3-DOF crane with electromagnetic gripper

The MACS Demonstrator Scenario



- Two separated “rooms”
- Sliding door, operated by switch
- Switch triggerable by weight
- Test objects:
  - Cylinders, spheres, boxes
  - Different tops, materials, sizes, weights, color combinations
- Simulator (MACSim) in ODE
  - Device-level simulation for the important robot elements (drives, (crane), cameras, laser scanner)

Agent Affordances

- pass(-through)-able (region between things)
- push-able (thing)
- lift-able (thing)
- place-able (offered by region)
- switch-trigger-able (lift-able + place-able)
- removable-from-switch (lift-able + place-able)
- traversable (free space + push-able)

Basic Decisions for MACS Planning

Use off-the-shelf propositional planning
based on PDDL domain and problem descriptions
(planner FF)

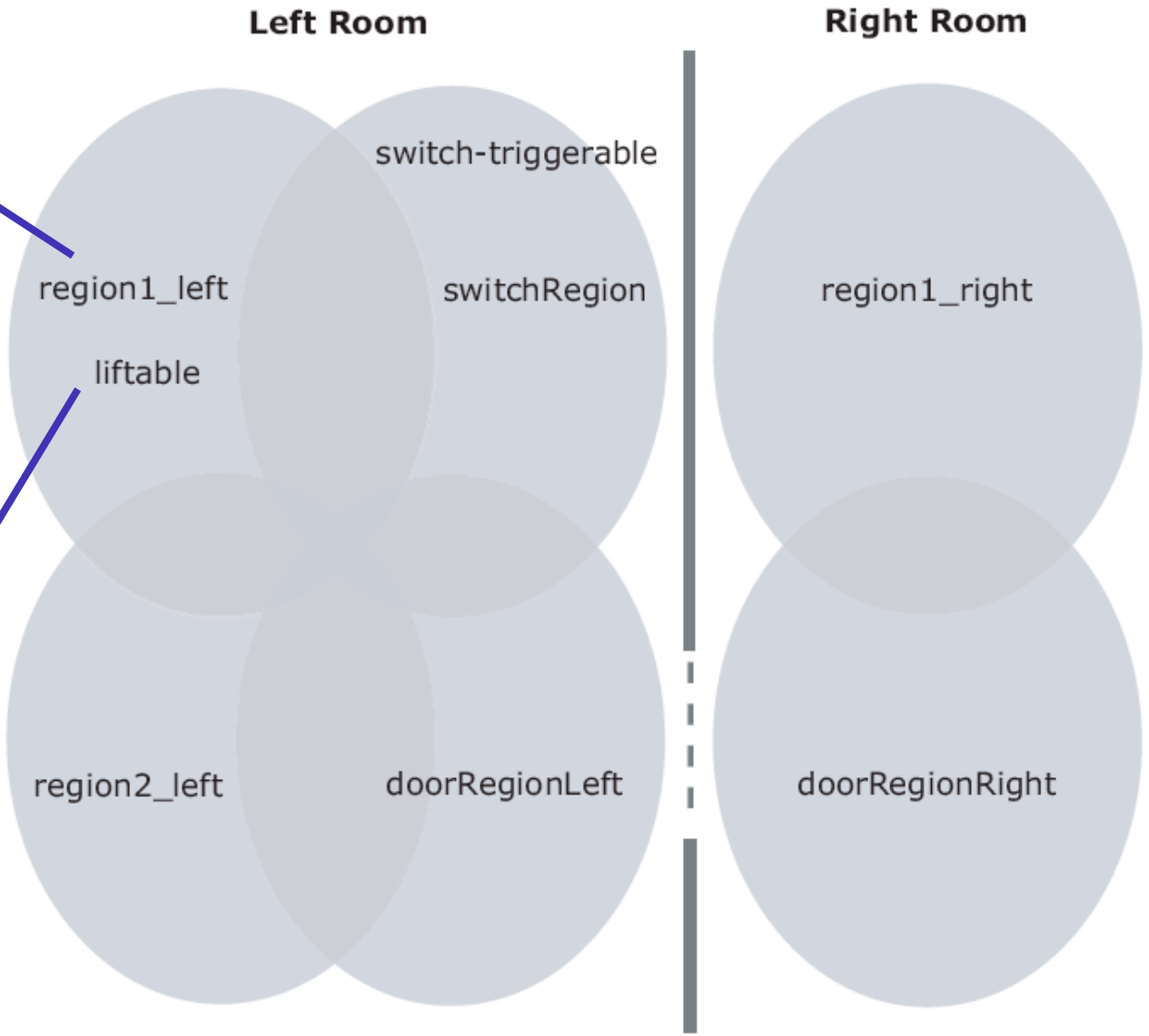
Some, not all plan operators model afforded actions

While performing in some environment,
log perceived affordance types per region!
(1 token/type/region; use map and standard localization)

Example: MACS Arena Affordance Map

Topological Region
(fuzzy boundaries)

Perceived Affordance Token
(arbitrarily often; possibly different cues)



MACS DD Example: Predicates

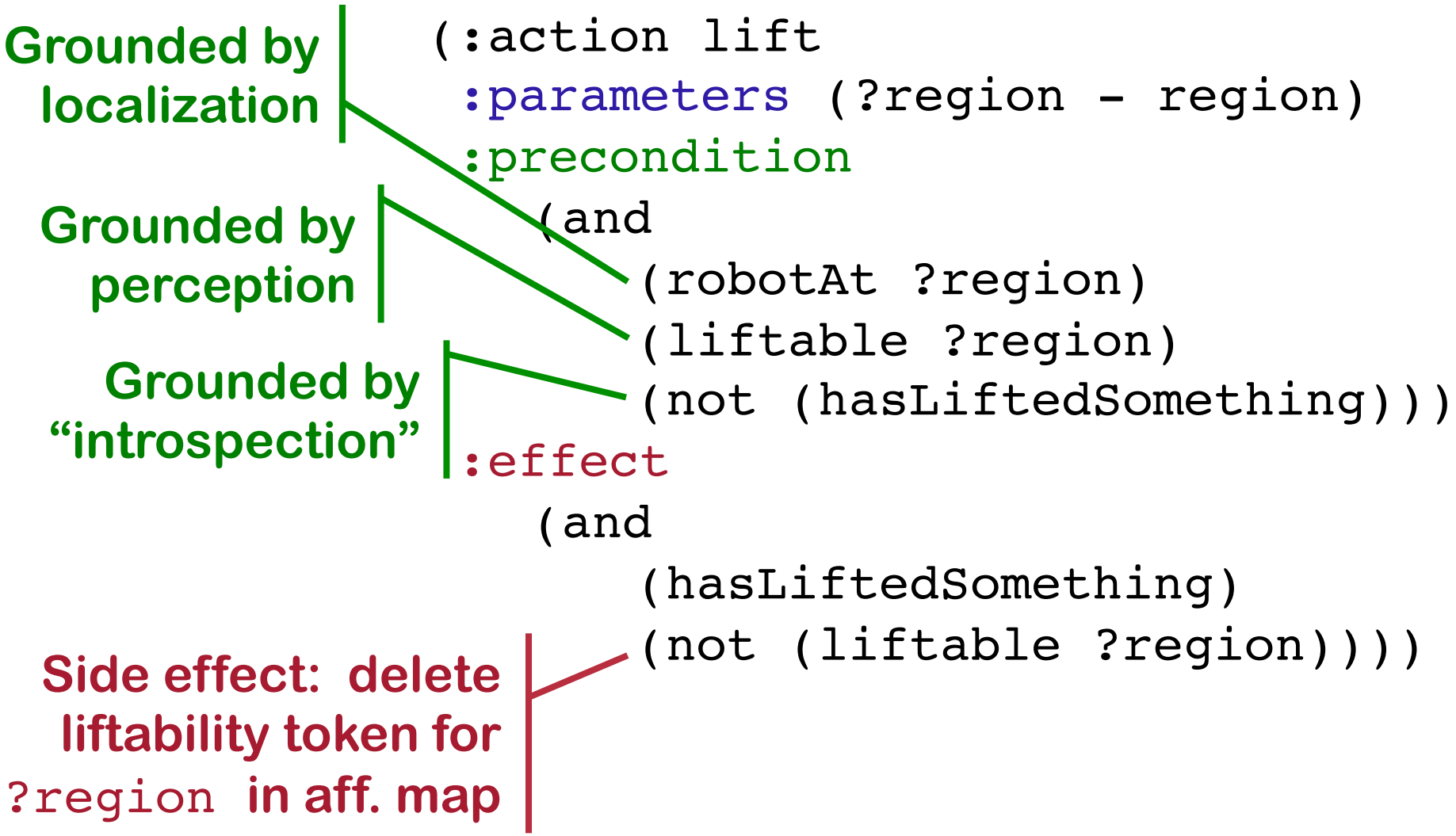
```
(:types region switchRegion doorRegion room)

(:predicates
      (robotAt ?region - region)
      (inRoom ?region - region ?room - room))
      (hasLiftedSomething)
      (liftable ?region - region )
      (switch-triggerable ?region - switchRegion)
      (passable ?startRegion ?targetRegion - region))
```

affordance tokens turn into properties of the regions where they have been perceived

(no objects sneaking in through the backdoor, except regions)

MACS DD Example: An Operator



Grounding Operators

- Ground operators in behaviors (↪ hybrid architecture)
  - e.g., lift operator is implemented using: DirectGoToPoseBehavior, 3DScanBehavior, ReachBehavior, PullBehavior, RaiseBehavior
- **Specialties induced by affordances**:
  If operator corresponds to afforded action, then grounding is provided by b,o in $\langle c,b,o \rangle$! Use **any** entity providing the affordance! (↪ **Opportunistic Execution**)
- Monitoring execution of afforded(!) action means to "**go with the flow of affordance**" …

… but exactly the selected one! (Selective attention)

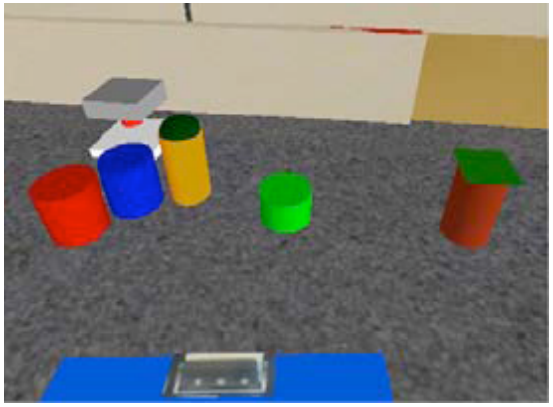
Opportunistic Execution

- At operator execution, perception is primed to attend to cues relevant for current operator execution
- Any(!) entity affording what is needed may be used ("lift something" vs. "lift object O_17")
- Naively object-based representations handle poppycock ("get me mug22 with coffee", rather than "get me a mug of coffee")

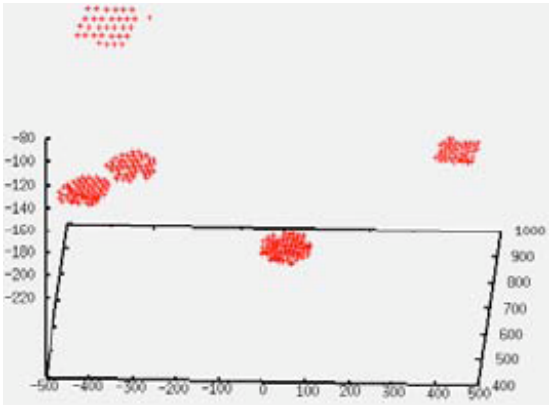
☛ Using entities & affordances in addition to(!) objects & properties appears to make a lot of sense! (future work)

MACS Liftability According to 3D Laser

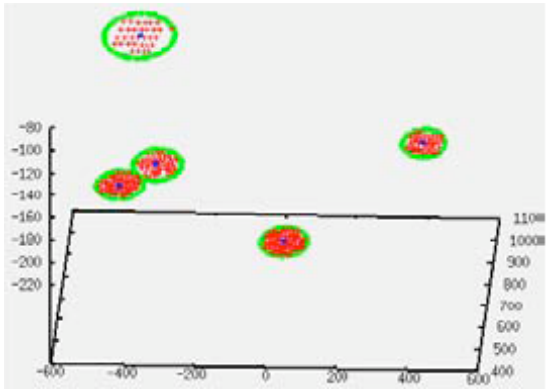
- Cues used in MACS based on vision (color, shape, SIFT features) and/or 3D laser scan data
- For plan execution implementation: only 3D laser scan data
- **Pro**: clean data; **Con**: measurement takes time
- Alternative today: Kinect (But mind sensor noise!)



Scene view



Horizontal patches above floor level



Clustered patches with center points

Grounding the Lift Operator

MACS Problem Description & Objects

```
(define (problem macs-prob)
 (:domain macs-example)
 (:objects rightRoom - room
   leftRoom - switchRoom
   region1_left region2_left region1_right - region
   switchRegion - switchRegion
   doorRegionLeft doorRegionRight - doorRegion )
 (:init (inRoom region1_left leftRoom)
   (inRoom region2_left leftRoom)
   (inRoom switchRegion leftRoom)
   …
   (robotAt region1_left)
   (liftable region1_left)
   (switch-triggerable switchRegion))
 (:goal (robotAt doorregionright)))
```

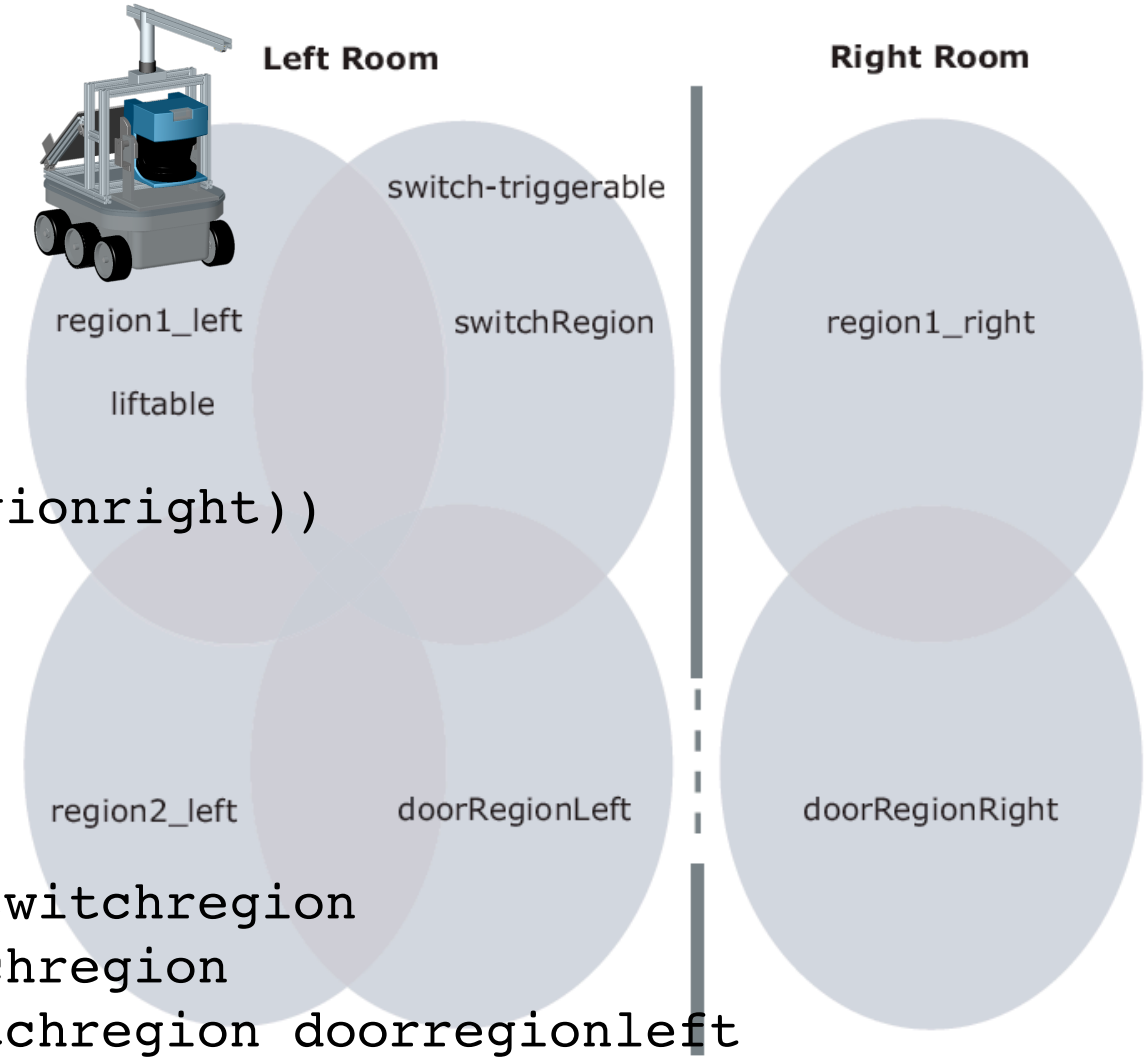
Types, predicates, actions

Yes, there are objects for the planner, but only few of them!

Static domain features

Dynamic fluents

Complete Example (Simulator)



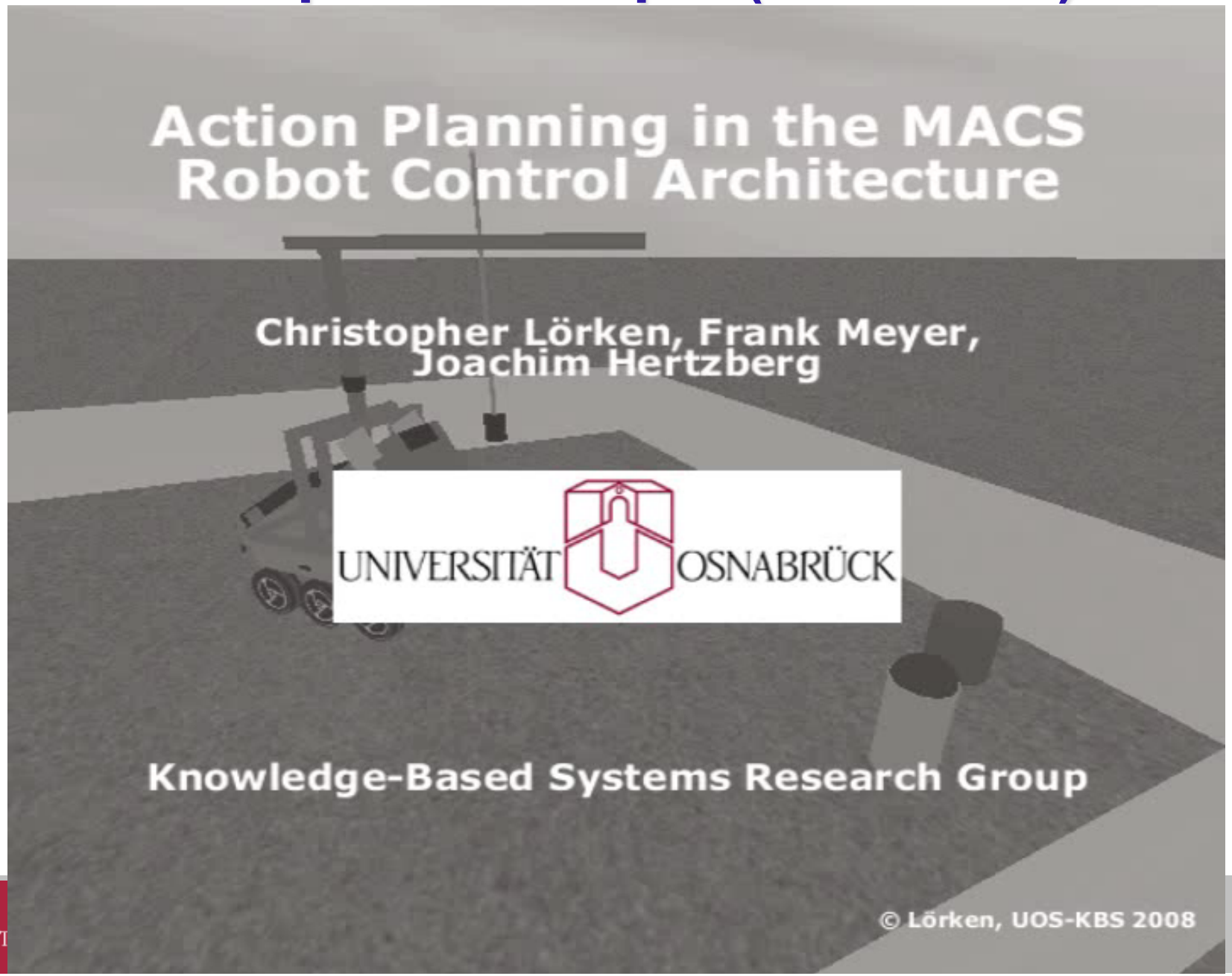
The Goal

```
(:goal (robotAt doorregionright))
```

The Plan (FF generated)

```
0: LIFT region1_left
1: CARRY region1_left switchregion
2: TRIGGER-SWITCH switchregion
3: APPROACH-REGION switchregion doorregionleft
4: CHANGE-ROOM doorregionleft doorregionright
```

Complete Example (Simulator)

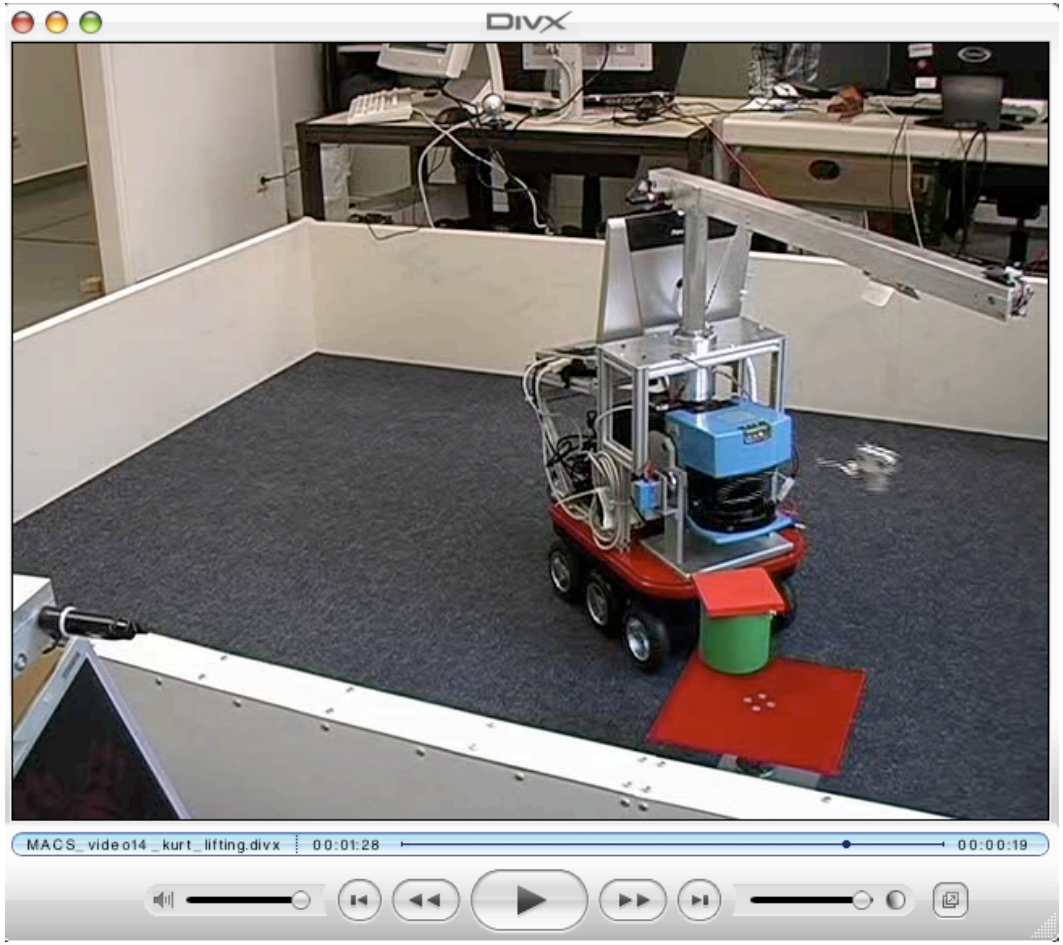


Execution Failure

- Planned operator execution by afforded actions may fail due to
  - **Model error** (affordance not present as mapped)
  - **Perception error** (cue is there but gets overlooked; affordance is perceived false-positively)
  - **Handling error** (afforded behavior fails)
- Reaction inventory in plan-based control: retry, re-plan, give up
- **Afforded actions** may be retried using **different affordance tokens of the same type**
  (spontaneously perceived or looked up in map and sought)

Failure & Retry: Simulation

Live Video



See video later,
if interested!
(and time)

Where are we?

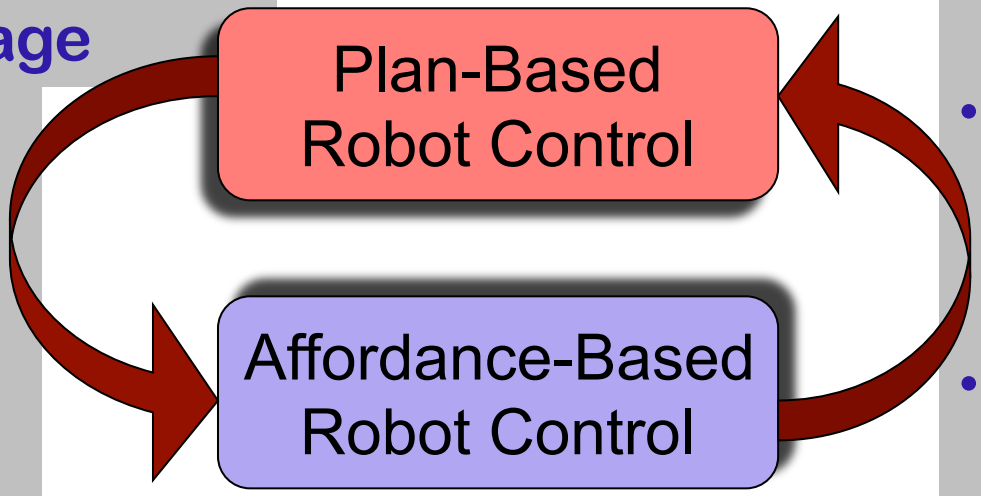
Use s-o-a AI planning technology
(PDDL language, FF planner)

Ground (some) domain concepts in $\langle c,b,o \rangle$s

- Tailor (some) operators after afforded actions
- Use affordances as preconditions for "opportunistic" execution
- Map perceived affordances
- Don't distinguish among function-ally equal objects
- Reduce search

Top-down influence on affordance usage

- Focus attention on affordances serving current operator; disregard others
- Search actively for cues signaling focused affordances



Use special modules
exec. control, behaviors, affordance repository

Summary of Contributions

The interplay between affordance-based and plan-based robot control has been explored for the first time

- ✓ p-b ctrl helps a-b ctrl focus top-down on relevant affordances
- ✓ a-b ctrl helps p-b ctrl ground actions and predicates
- ✓ employed s-o-a propositional planner in robot ctrl
- ✓ found operational model for opportunistic plan execution
- ✓ found simple mechanism for handling anonymous entities/ objects in propositional robot planning

Thank you for your time!