ARC Centre of Excellence for Autonomous Systems

A Relational Approach to Tool Use Learning in Robots

Solly Brown
Claude Sammut

School of Computer Science and Engineering
The University of New South Wales

Learning to Use Tools

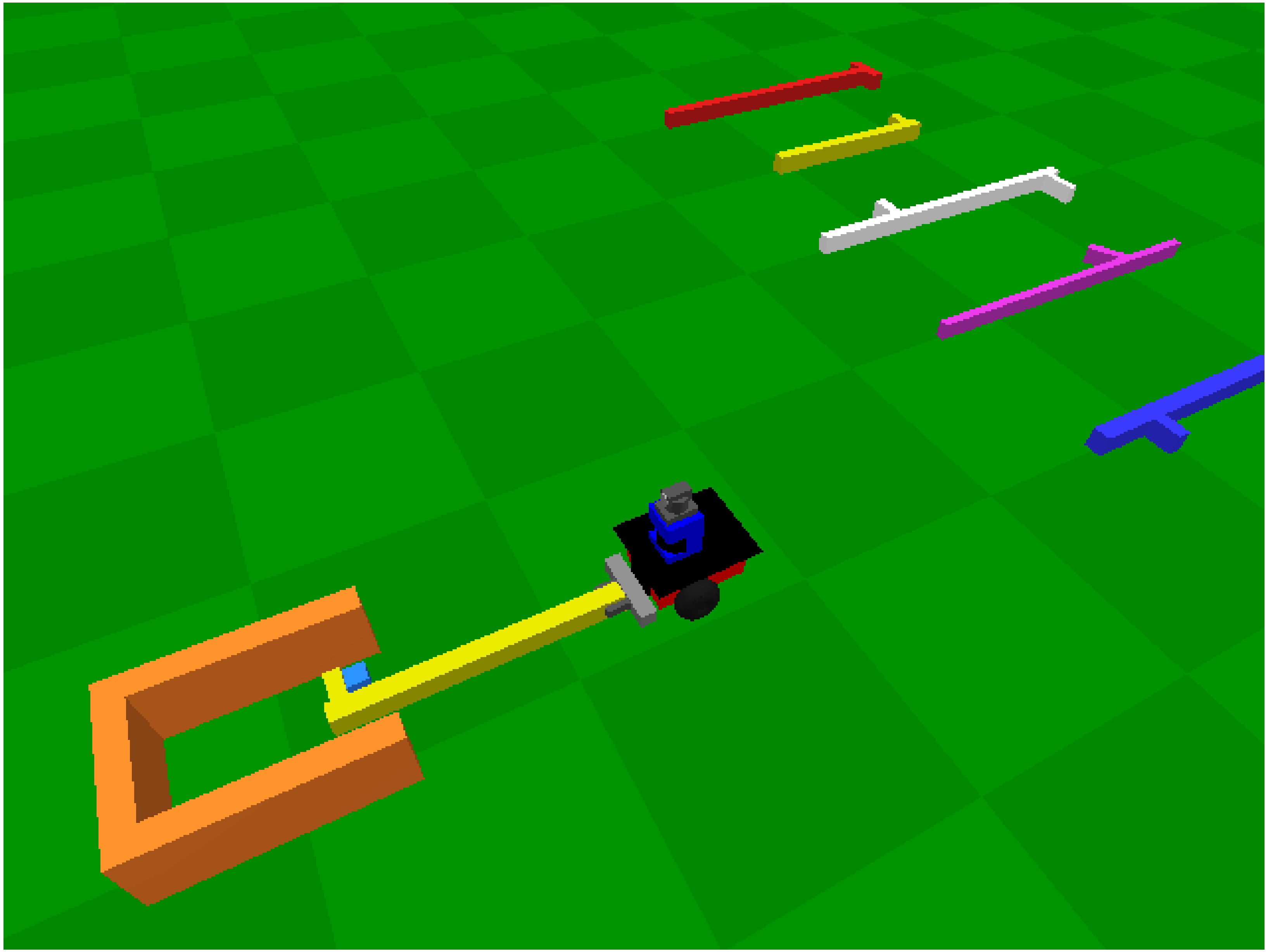
Learn to use an object as a tool by:

observing another agent use the tool to achieve a goal

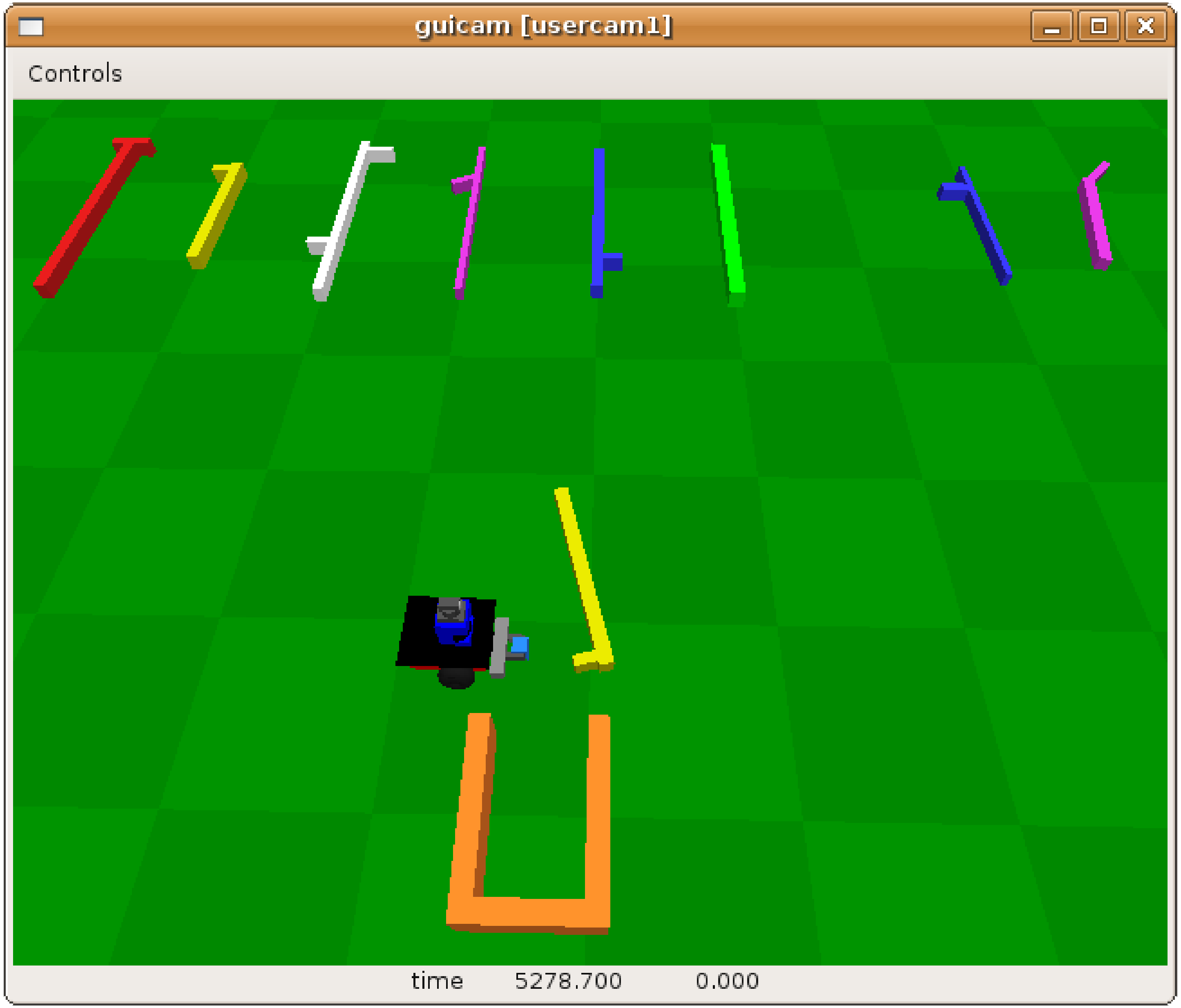
Trial-and-error to refine a theory about how to use the tool

Autonomous Systems

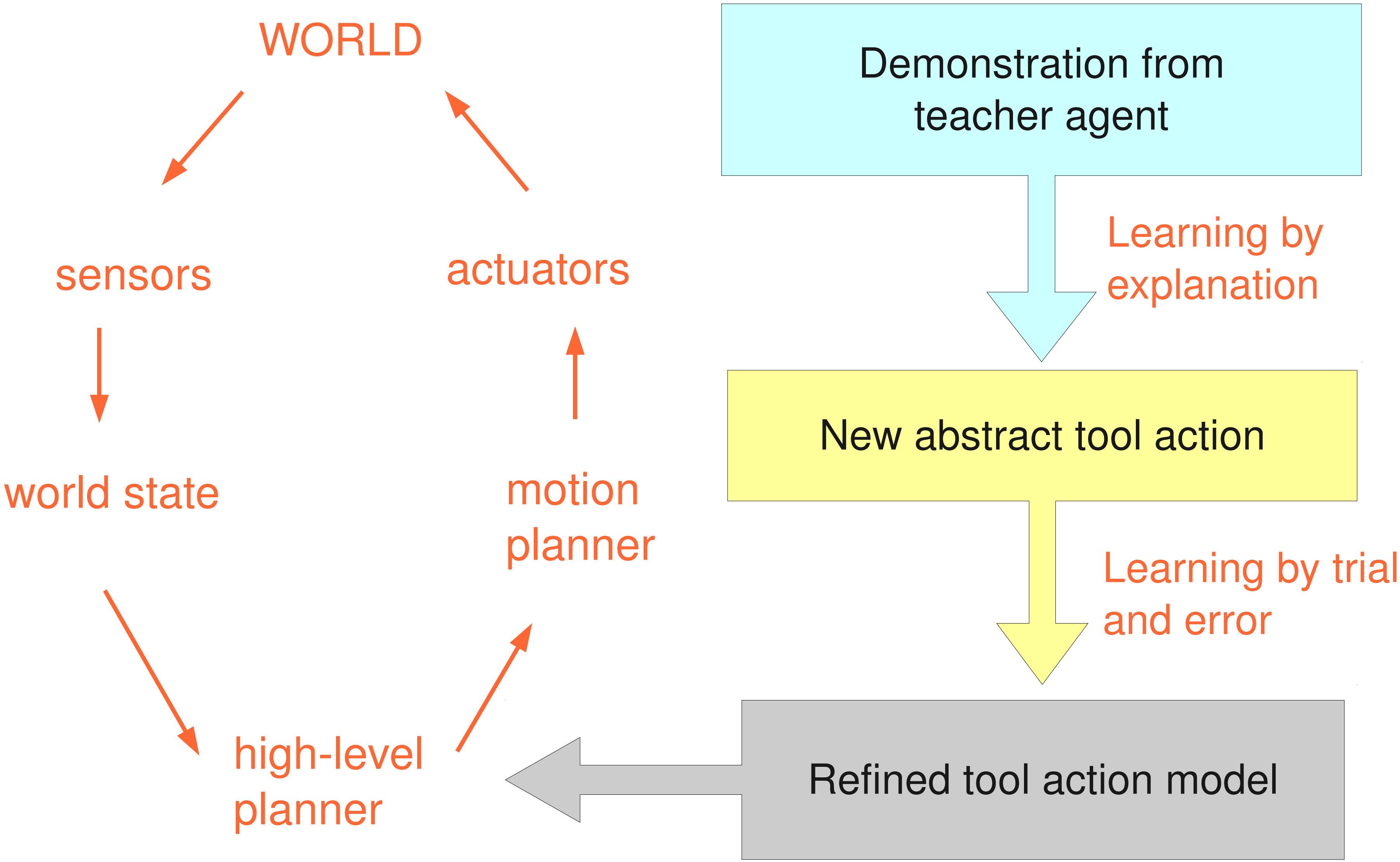
Simulated Robot World



Trainer's Demonstration



Learning Agent Architecture



Assumptions

All objects are graspable

i.e. no control problems

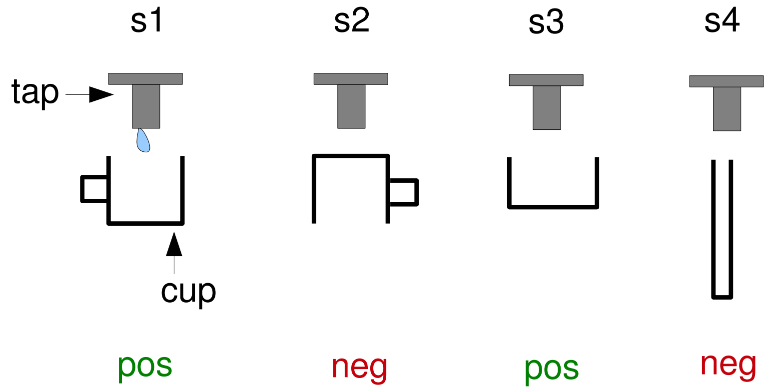
The robot has background knowledge of some actions

Unknown action sequence consists of

tool positioning actions

application of tool resulting in goal state

Action Models



ACTION put_under(Cup, Tap)

PRE in_gripper(Cup),

gripping(Cup),

clear_underneath(Tap),

orientation(Cup, vertical-up)

ADD below(Cup, Tap),

near(Cup, Tap),

aligned_vertically(Cup, Tap)

DEL clear_underneath(Tap)

MOVING robot-arm, Cup // manipulated objects

PRIMITIVES fwd, back, left, right, up, down, rotatecw, rotateccw

Background Knowledge

grip(Obj)

PRE	¬gripping, in_gripper(Obj)
ADD	gripping
DEL	-
PRIMTV	closeGrip
MOVING	-

ungrip(Obj)

PRE	gripping, in_gripper(Obj)
ADD	-
DEL	gripping
PRIMTV	openGrip
MOVING	-

remove_from_gripper(Obj)

PRE	in_gripper(Obj), ¬gripping
ADD	empty gripper
DEL	in_gripper(Obj)
PRIMTV	back
MOVING	robot

put_in_gripper(Obj)

PRE	forall(Tube:tube, ¬in(Obj,Tube)), empty_gripper, ¬gripping, forall(Obstacle:obj, ¬obstructing(Obstacle,Obj))
ADD	in_gripper(Obj)
DEL	empty_gripper
PRIMTV	fwd, back, rotleft, rotright
MOVING	robot

move_obstacle(ObjA,ObjB)

PRE	moveable obj(ObjA), obstructing(ObjA,ObjB), in_gripper(ObjA), gripping
ADD	-
DEL	obstructing(ObjA,ObjB)
PRIMTV	fwd, back, rotleft, rotright
MOVING	robot, ObjA

recognise_goto

PRE	empty_gripper
MOVING	robot

recognise_carry_obj(Obj)

PRE	in_gripper(Obj), gripping
MOVING	robot, Obj

State Predicates

Dynamic predicates:

- in_gripper(+obj,+state)
- touching(+obj,-obj,-side,+state)
- at_right_angles(+obj,+obj,+state)
- at_oblique_angle(+obj,+obj,+state)
- parallel(+obj,+obj,+state)
- on_axis(+obj,+obj,+state)
- on_perp_axis(+obj,+obj,+state)
- in_tube(+obj,-tube,+state)
- in_tube_end(+obj,-tube,-end,+state)
- in_tube_side(+obj,-tube,-side,+state)
- obstructing(+obj,+obj,+state)

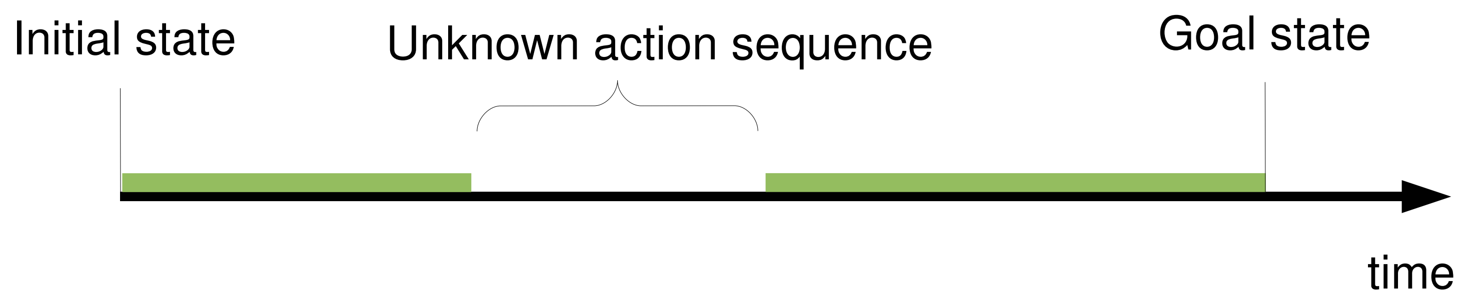
Static predicates:

- attached_side(+obj,-obj,-side)
- attached_end(+obj,-obj,-disttype)
- attached_angle(+obj,-obj,-angletype)
- num_attachments(+obj,-number)
- longest_component(+obj)
- attached_type(+obj,-obj,-attachtype)
- narrower(+obj,+obj)
- shorter(+obj,+obj)
- shape(+obj,-shape)
- closed_tube(-tube, -obj, -obj, -obj)

The world is continuous

First step is to discretise time-series observations of trainer

Convert to action sequence



Segmentation Heuristics

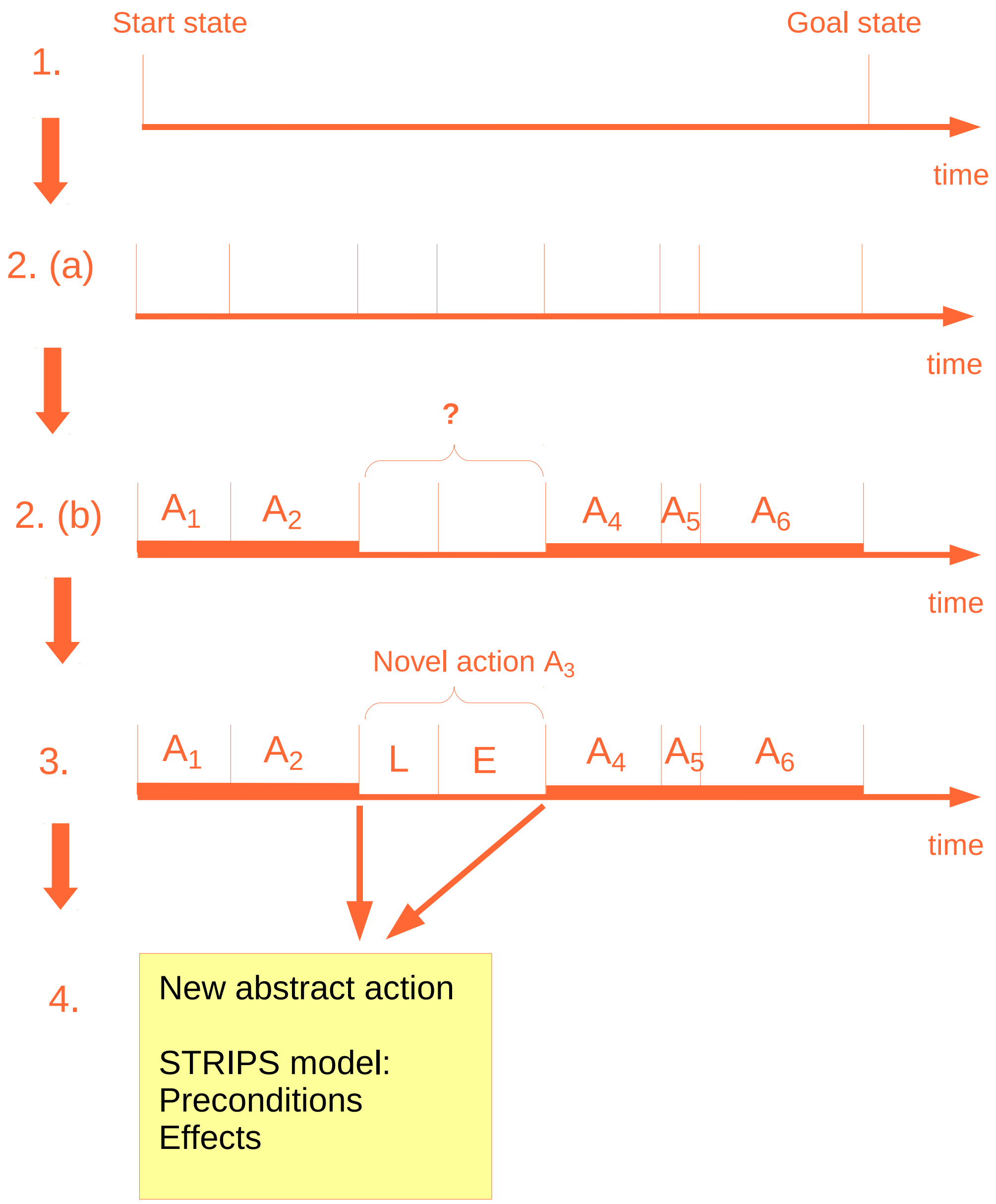
Object Motion

A distinct action begins or ends each time an object or agent starts or stops moving

Object Contact

An action starts or stops when two objects come into contact or break contact

Segmenting Observations



Segment on changes

Identify known action

Extract novel actions

ARC Centre of Excellence for Autonomous Systems

Matching Actions

A segment is matched with an abstract action if:

- Objects manipulated in the segment can be matched to MOVING list in action model
- Preconditions are true at beginning of segment
- Effects are true at end of segment

If more than one action matches:

- choose action with most specific preconditions

Segmentation

Segment	Moving objects	Teacher's action (unknown)
1	robot	Put tool in gripper
2	gripper	Close gripper
3	robot, tool	Put tool in pulling pose
4	robot, tool, box	Pull box with tool
5	robot, tool	Put tool aside
6	gripper	Open gripper
7	robot	Put box in gripper
8	gripper	Close gripper
9	robot, box	Carry away box

Explanation of Observed Actions

Segment	Moving objects	Explanation	Match Type
1	robot	put in gripper(hookstick)	exact
2	gripper	grip(hookstick)	exact
3	robot, tool	recognise_carry_obj(hookstick)	partial
4	robot, tool, box	??	none
5	robot, tool	move obstacle(hookstick,box)	exact
6	gripper	ungrip(hookstick)	exact
7	robot	remove_from_gripper(hookstick), put in gripper(box)	exact
8	gripper	grip(box)	exact
9	robot, box	recognise_carry_obj(box)	partial

Invented Actions

position_tool(Tool, Box)

PRE:
- in_gripper(Tool),
- gripping

ADD:
- tool_pose(Tool, Box),
- obstructing(Tool, Box)

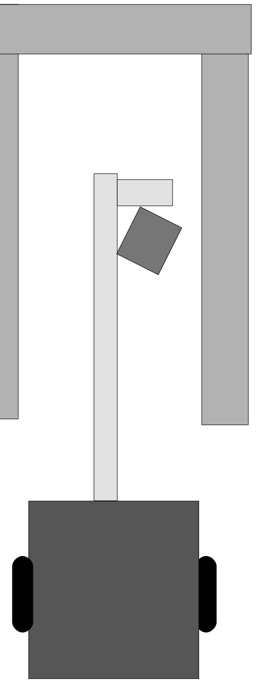
DEL: -

pull_from_tube(Tool, Box, Tube)

PRE:
- tool_pose(Tool, Box),
- in_gripper(Tool),
- gripping,
- in_tube(Box, Tube)

ADD: -

DEL: in_tube(Box, Tube)



Inductive Logic Programming

Initial version space boundaries:

```
tool_poseG(Tool, Box, State) :- true.

tool_poseS(Tool, Box, State) :- saturation(s1).
```

Most specific hypothesis is constructed by saturating trainer's example

Bottom Clause

tool poseS(Tool, Box, State):-

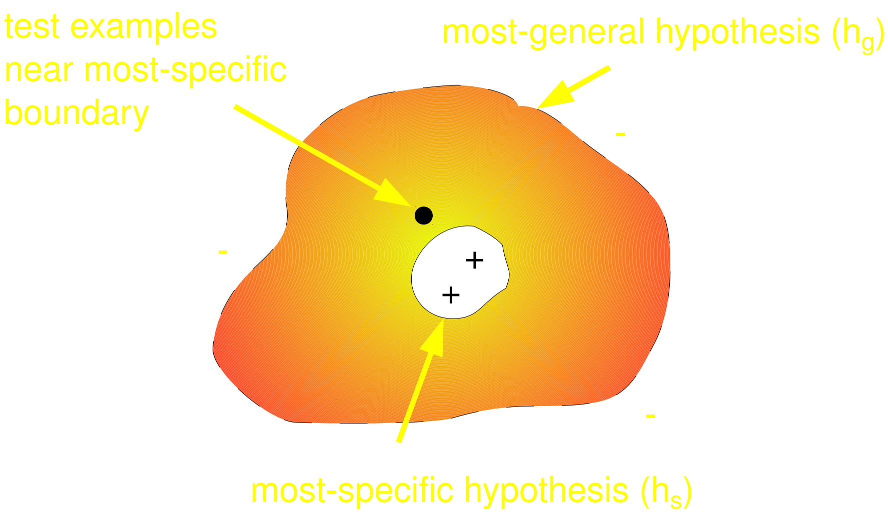
% static properties

attached(Tool, Hook),
num attachments(Tool, 1),
num attachments(Box, 0),
longest component(Tool),
narrower(Tool, Box),
shorter(Box, Tool),
shape(Tool, sticklike),
shape(Box, boxlike),
closed tube(Tube, TubeLeft, TubeRight, TubeBack),
attached side(Tool, Hook, right),
attached side(TubeLeft, TubeBack, right),
attached side(TubeRight, TubeBack, left),
attached end(Tool, Hook, front),
attached end(TubeLeft, TubeBack, front),
attached end(TubeRight, TubeBack, front),
attached angle(Tool, Hook, right angle),
attached angle(TubeLeft, TubeBack, right angle),
attached angle(TubeRight, TubeBack, right angle),
attached type(Tool, Hook, end to end),
attached type(TubeLeft, TubeBack, end to end),
attached type(TubeRight, TubeBack, end to end),
num attachments(Hook, 0),
num attachments(TubeLeft, 1),
num attachments(TubeRight, 1),
num attachments(TubeBack, 0),
narrower(Hook, Tool),
narrower(Hook, Box),
narrower(Tool, TubeLeft),
narrower(TubeLeft, Box),
narrower(Hook, TubeLeft),
narrower(Tool, TubeRight),
narrower(TubeRight, Box),
narrower(Hook, TubeRight),
narrower(Tool, TubeBack),
narrower(TubeBack, Box),
narrower(Hook, TubeBack),
narrower(TubeBack, TubeLeft),
narrower(TubeBack, TubeRight),
shorter(Hook, Tool),
shorter(Tool, TubeLeft),
shorter(Tool, TubeRight),
shorter(TubeBack, Tool),
shorter(Box, Hook),
shorter(Box, TubeLeft),
shorter(Hook, TubeLeft),
shorter(TubeBack, TubeLeft),
shorter(Box, TubeRight),
shorter(Hook, TubeRight),
shorter(TubeBack, TubeRight),
shorter(TubeBack, TubeLeft),
shorter(Box, TubeBack),
shorter(Hook, TubeBack),
shape(TubeLeft, sticklike),
shape(TubeRight, sticklike),
shape(TubeBack, sticklike),

% fluents

in gripper(Tool, State),
touching(Tool, Box, right, State),
at oblique angle(Tool, Box, State),
in tube(Box, Tube, State),
in tube end(Box, Tube, front, State),
in tube side(Box, Tube, right, State),
touching(Hook, Box, back, State),
at right angles(Hook, TubeLeft, State),
at right angles(Hook, TubeRight, State),
at right angles(Tool, TubeBack, State),
at oblique angle(Box, Hook, State),
at oblique angle(Box, TubeLeft, State),
at oblique angle(Box, TubeRight, State),
at oblique angle(Box, TubeBack, State),
parallel(Tool, TubeLeft, State),
parallel(Tool, TubeRight, State),
parallel(Hook, TubeBack, State),
in tube(Hook, Tube, State),
in tube end(Hook, Tube, front, State),
in tube side(Hook, Tube, right, State).

Generating an Experiment



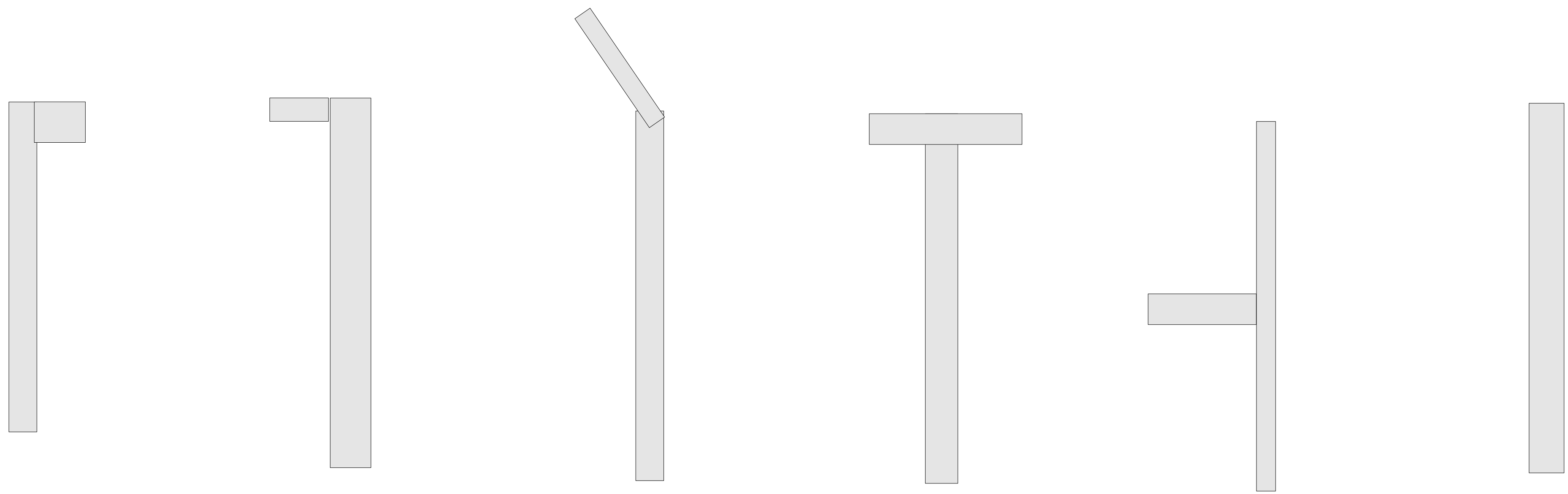
Start with H_G and add literals from H_S but ...

select tool that satisfies the most structural constraints in H_S

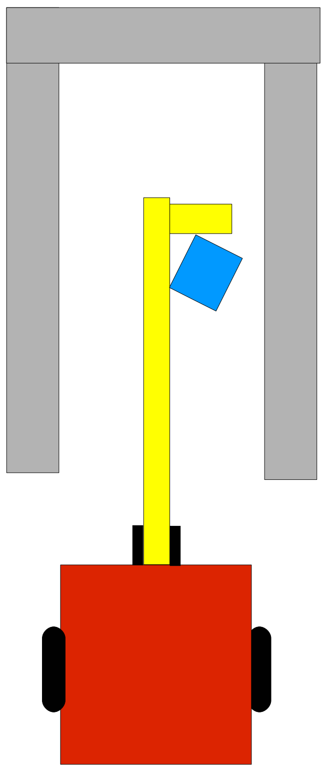
select pose to maximise the spatial literals satisfied in H_S

Bias specific-to-general search results by trying to generate experiments near most specific boundary

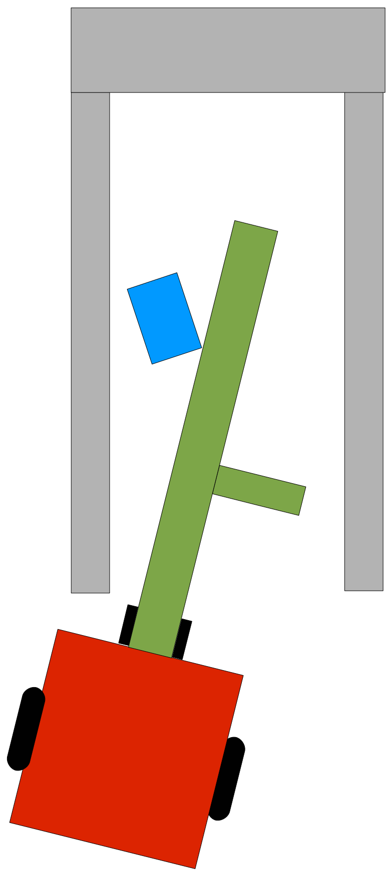
Tools Available for Experimentation



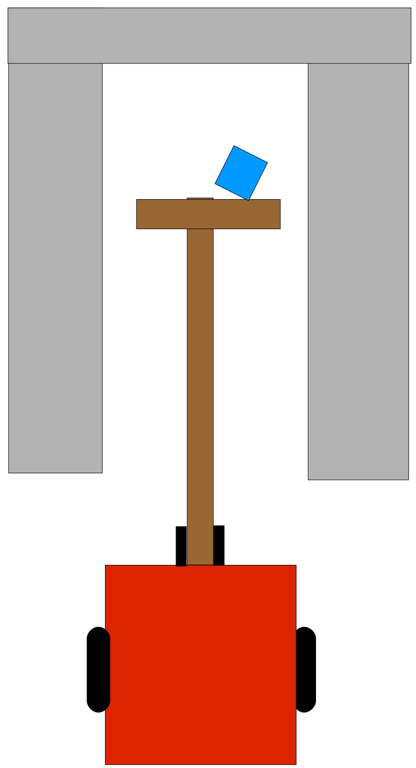
Positive and Negative Examples of Tool Use



pos



neg



neg

Updating the Version Space

An uncovered positive example generalises H_S

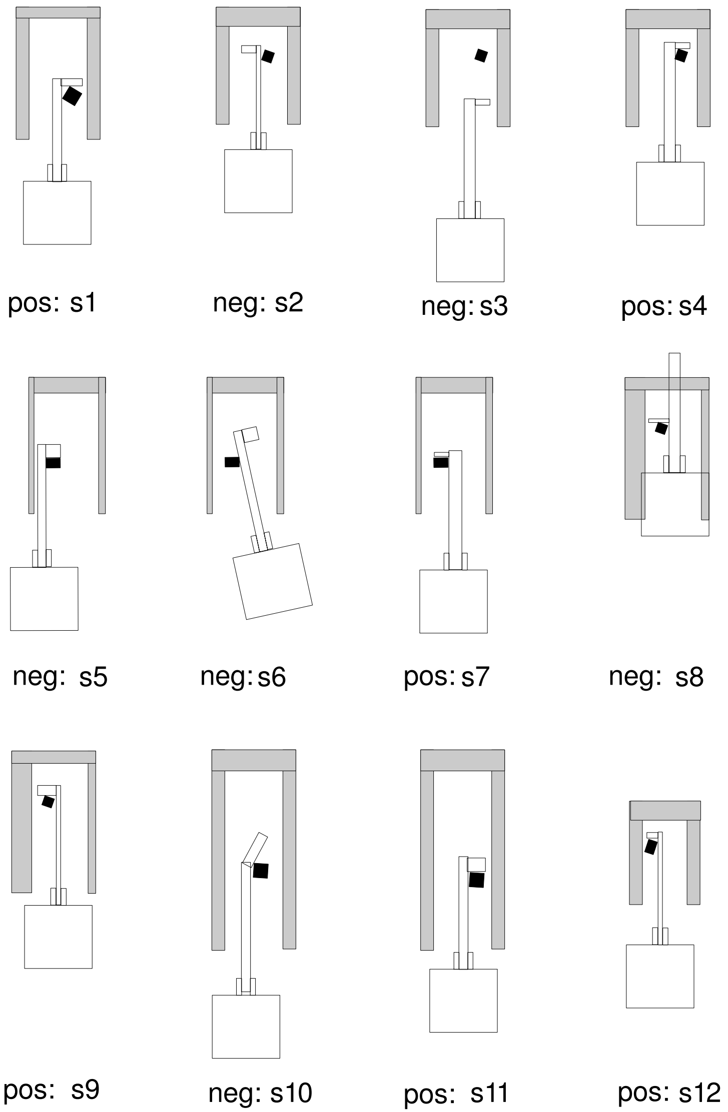
Like GOLEM, find RLGG of old hypothesis and new example

A covered negative example specialises H_G

Use negative-based reduction to recreate H_G

Find most general H_G by removing literals from H_S as long as negative examples are not covered

Experiments



Evolution of most general hypothesis:

tool_poseG(Tool, Box, State) :-
    attached side(Tool, Hook, right).

. . .

tool_poseG(Tool, Box, State) :-
    attached side(Tool, Hook, right),
    touching(Hook, Box, back, State).

. . .

tool_poseG(Tool, Box, State) :-
    attached(Tool, Hook),
    narrower(Hook, Box),
    touching(Hook, Box, back, State).

. . .

tool_poseG(Tool, Box, State) :-
    in_tube_side(Box, Tube, Side, State),
    attached_side(Tool, Hook, Side),
    touching(Hook, Box, back, State),
    attached_angle(Tool, Hook, rightangle),
    attached_end(Tool, Hook, back).

Making a plan operational

To run experiment:

learned actions must be incorporated into a plan

sequence of abstract actions generated by planer must be turned into action primitives

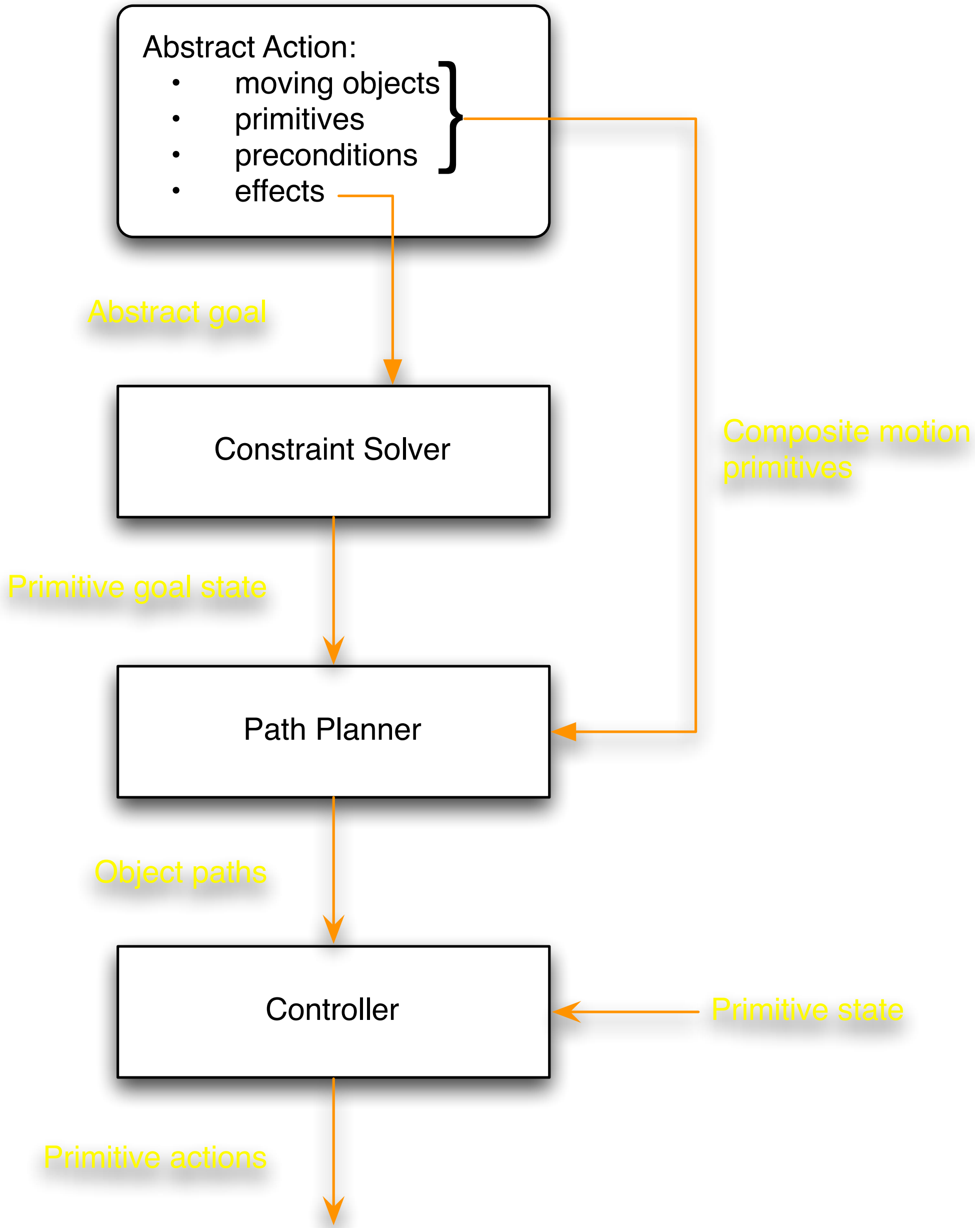
i.e. real numbers needed!

Behaviour Generation

Fast Forward
(Hoffman & Nebel)

ECLiPSe
(Apt & Wallace)

Rapid Random Trees
(Kuffer & LaValle)



Generating Behaviour from Abstract Action

Extract the spatial sub-goals from action model

Constraint solver finds a primitive world state satisfying spatial subgoals

Moveable objects in MOVING list are treated as a single geometric object to be manipulated by the motion planner. Allowed movement primitives are given by PRIMITIVES list.

Motion planner uses composite object motion primitives to generate a path from current world state to ground goal state generated in step 2

Track path using a generic controller

- Follow until the primitive goal state is reached
- Failures detected by timeout cause re-planning