Extending Knowledge-Level Contingent Planning for Robot Task Planning

Ron Petrick  
University of Edinburgh, Edinburgh, Scotland, United Kingdom

Andre Gaschler  
fortiss GmbH, An-Institut Technische Universität München, Munich, Germany

Workshop on Planning and Robotics (PlanRob)  
Portsmouth, New Hampshire, USA  
23 June 2014
Robot task planning

• A robot operating in a real-world domain often needs to do so with **incomplete** or **uncertain information** about the state of the world.

• A robot with the ability to **sense** the world can gather information to generate plans with **contingencies**, allowing it to reason about the outcome of sensed data.

• Symbolic planning with incomplete information and sensing provides a promising approach.

• An added complication for robot task planning: bridging the gap between geometric and symbolic representations.
  
  – **Robot level**: continuous representations for modelling properties like joint angles and spatial coordinates.
  
  – **Planning level**: discrete representations in logic-like languages.

• **Our approach**: combine general purpose high-level symbolic planning with low-level geometric and motion planning.
• **3D geometric volumes** are a natural intermediate representation for bridging motion planning and high-level task planning.

• **Continuous geometry** is preferable over indiscriminate discretisation, and it is efficient enough for real systems.

• **Knowledge-level planning** naturally enables reasoning about acting and sensing in structure, partially known environments.
Robotics side: 3D volume intermediate representation

- Efficient swept volume computation with sets of convex bodies (Gaschler et al. 2013a,b,c), building on the approach of (Mamou and Ghorbel 2009).
This talk: the planning side

1. Knowledge-level contingent planning

2. Planning with external reasoning

3. Planning with interval-valued fluents
PKS: Planning with Knowledge and Sensing

- Our approach: an application of general purpose planning with incomplete information and sensing.
- Plans are generated using PKS (Petrick and Bacchus 2002, 2004), a knowledge-level contingent planner that builds plans based on the planner’s knowledge state.
- PKS uses an extended STRIPS-style representation, based collection of five databases, each of which is restricted to a particular type of knowledge: $K_f$, $K_v$, $K_w$, $K_x$, LCW.
- The contents of the databases ($DB$) have a fixed formal translation to formulae in a modal logic of knowledge which formally defines the planner’s knowledge state ($KB$).
- Actions are defined in terms of the changes they make to the planner’s knowledge state (i.e., the databases), rather than the world state.
- Plans are generated by forward search: actions update $DB \Rightarrow$ update $KB$. 
Representing knowledge in PKS

- $K_f$: knowledge of positive and negative facts (but not closed world!)
  \[ p(c) \quad \neg q(b, c) \quad f(a) = c \quad g(b, c) \neq d \]

- $K_w$: knowledge of binary sensing effects
  \[ \phi \in K_w : \text{the planner knows whether } \phi \]

- $K_v$: knowledge of function values, multi-valued sensing effects
  \[ f \in K_v : \text{the planner knows the value of } f \]

- $K_x$: exclusive-or knowledge
  \[ (\ell_1|\ell_2|\ldots|\ell_n) \in K_x : \text{exactly one of the } \ell_i \text{ must be true} \]

- $LCW$: local closed world information (Etzioni et al. 1994)
action senseWeight(?o : object)
  preconds:
    !Kw(isSpillable(?o)) &
    K(isGrasped(?o))
  effects:
    add(Kw, isSpillable(?o))

action transfer(?o : object)
  preconds:
    K(!isSpillable(?o)) &
    K(isGrasped(?o)) &
    K(!isRemoved(?o))
  effects:
    add(Kf, isRemoved(?o))

• The planner builds plans by chaining actions together using search.
• Contingent plans are built from the planner’s $K_w$ and $K_v$ knowledge.
Example 1: Force Sensing domain

- **Task**: remove cans to a special location using weight-dependant grasps.
A contingent planning solution

```plaintext
action senseWeight(?o : object)
action transfer(?o : object)
action transferUpright(?o : object)
   preconds:
   K(isSpillable(?o)) &
   K(isGrasped(?o)) &
   K(!isRemoved(?o))
   effects:
   add(Kf, isRemoved(?o))
action grasp(?o : object)
   preconds:
   K(emptyGripper) &
   K(!isRemoved(?o))
   effects:
   add(Kf, isGrasped(?o)),
   add(Kf, !emptyGripper)
action ungrasp(?o : object)
   preconds:
   K(isGrasped(?o)) &
   K(isRemoved(?o))
   effects:
   add(Kf, !isGrasped(?o)),
   add(Kf, emptyGripper)
```

1. grasp(can1);  
2. senseWeight(can1);  
3. branch(isSpillable(can1))  
4. K+:  
5. transferUpright(can1);  
6. ungrasp(can1);  
7. grasp(can2);  
8. senseWeight(can2);  
9. branch(isSpillable(can2))  
10. K+:  
11. transferUpright(can2);  
12. ungrasp(can2).  
13. K-:  
14. transfer(can2);  
15. ungrasp(can2).  
16. K-:  
17. transfer(can1);  
18. ungrasp(can1);  
19. grasp(can2);  
20. senseWeight(can2);  
21. branch(isSpillable(can2))  
22. K+:  
23. transferUpright(can2);  
24. ungrasp(can2).  
25. K-:  
26. transfer(can2);  
27. ungrasp(can2).
Example plan execution

A: grasp(can1)
senseWeight(can1)
branch(isSpillable(can1))

B: transferUpright(can1)
ungrasp(can1)

C: grasp(can2)
senseWeight(can2)
branch(¬isSpillable(can1))

D: transfer(can2)
ungrasp(can2)
Planning with external reasoning

action pickUp(?r:robot, ?o:object, ?l:location)
preconds:
  K(?l = getObjectLocation(?o)) &
  K(handEmpty(?r)) &
  K(extern(isReachable(?l,?r)))
effects:
  del(Kf,?l = getObjectLocation(?o)),
  del(Kf,handEmpty(?r)),
  add(Kf,inHand(?o,?r))

• We have extended PKS to allow calls to external procedures, to help evaluate preconditions and effects. This allows optimised or special purpose libraries to be more easily integrated with the planner, e.g.,

  extern(proc(\vec{x}))

• Similar to semantic attachments and related techniques, e.g.,
  (Dornhege et al. 2009; Erdem et al. 2011; Eiter et al. 2006).
Example 2: Bimanual domain

- **Task**: recognise and remove empty bottles from the table to a dishwasher location.
A solution with external geometric reasoning


action senseIfEmpty(?o:obj)
  preconds:
    !Kw(isEmptyBottle(?o))
  effects:
    add(Kw, isEmptyBottle(?o))

  preconds:
    K(inHand(?o, ?r)) &
    K(extern(isReachable(?l, ?r)))
  effects:
    del(Kf, inHand(?o, ?r)),
    add(Kf, ?l = getobjectLoc(?o)),
    add(Kf, handEmpty(?r))

1. senseIfEmpty(bottle0);
2. senseIfEmpty(bottle1);
3. senseIfEmpty(bottle2);
4. senseIfEmpty(bottle3);
5. branch(isEmptyBottle(bottle0))
6. K+:
7. branch(isEmptyBottle(bottle1))
8. K+:
9. K-:
10. branch(isEmptyBottle(bottle2))
11. K+:
12. branch(isEmptyBottle(bottle3))
13. K+:
14. K-:
15. pickUp(left, bottle0, 10);
16. putDown(left, bottle0, 15);
17. pickUp(right, bottle2, 12);
18. putDown(right, bottle2, dishwasher);
19. pickUp(right, bottle0, 15);
20. putDown(right, bottle0, dishwasher).
21. K-:
22. K-:
Example plan execution

A  senseIfEmpty(bottle0)
    ...
    senseIfEmpty(bottle3)

B  pickUp(robotleft, bottle0, loc0)

C  putDown(robotleft, bottle0, loc5)

D  

E  pickUp(robotright, bottle2, loc2)

F  putDown(robotright, bottle2, dishwasher)

G  pickUp(robotright, bottle0, loc5)

H  putDown(robotright, bottle0, dishwasher)
Planning with interval-valued fluents

- Uncertain numerical information can be modelled in a compact form using **interval-valued fluents (IVFs)** (Funge 1998; Poggioni et al. 2003; Petrick 2011) whose mappings denote the range of possible values for the function. E.g.,

\[ f = \langle 1, 100 \rangle \]

means that \( f \) can possibly map to any value between 1–100.

- We extend PKS to use IVFs of the form

\[ f(\vec{c}) = \langle u, v \rangle \]

where \( u \) and \( v \) indicate the (closed) range of possible mappings for \( f(\vec{c}) \).

- **Point intervals** of the form

\[ f(\vec{c}) = \langle u, u \rangle \]

denote definite knowledge, i.e., \( f(\vec{c}) \) is known to be equal to \( u \).
Contingent planning with IVFs

- $K_w$ allows certain numeric relations to be explicitly represented and can be used for building conditional plan branches. E.g.,

$$f(\vec{c}) \; \text{op} \; d \in K_w$$

is allowed, where $\text{op} \in \{=, \neq, >, <, \geq, \leq\}$ and $d$ is a numeric constant.

- $K_w$ information can also be used together with interval-valued knowledge to reason about certain restricted subcases. E.g., if

$$f = \langle 3, 10 \rangle \in K_f, 
\quad f > 5 \in K_w,$$

then the planner can introduce a conditional branch that “splits” the $K_w$ information into two parts and updates the $K_f$ knowledge appropriately:

- $K^+$ branch: $f > 5 \in K_f, f = \langle 6, 10 \rangle \in K_f,$
- $K^-$ branch: $f \leq 5 \in K_f, f = \langle 3, 5 \rangle \in K_f.$
Example 3: Localisation domain (simulated)

action noisyForward
   effects: add(Kf, robotLoc := robotLoc - <1,2>)

action moveBackward
   effects: add(Kf, robotLoc := robotLoc + 1)

action atTarget
   effects: add(Kw, robotLoc = targetLoc)

action withinTarget
   effects: add(Kw, robotLoc <= targetLoc)

• Domain properties: \( robotLoc \) is the distance from a robot to a wall, \( targetLoc \) is the desired location of the robot.

• Initial knowledge: \( robotLoc = \langle 3, 4 \rangle \in K_f, targetLoc = 2 \in K_f. \)

• Task: move the robot to the target location.
A solution with interval-valued reasoning

<table>
<thead>
<tr>
<th>0</th>
<th>( \text{noisyForward} );</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \text{withinTarget} );</td>
</tr>
<tr>
<td>2</td>
<td>( \text{branch}(\text{robotLoc} \leq \text{targetLoc}) )</td>
</tr>
<tr>
<td>3</td>
<td>( \text{branch}(\text{robotLoc} = \text{targetLoc}) )</td>
</tr>
<tr>
<td>4</td>
<td>( \text{atTarget} );</td>
</tr>
<tr>
<td>5</td>
<td>( \text{atTarget} );</td>
</tr>
<tr>
<td>6</td>
<td>( \text{moveBackward} ).</td>
</tr>
<tr>
<td>7</td>
<td>( \text{moveBackward} ).</td>
</tr>
<tr>
<td>8</td>
<td>( \text{noisyForward} );</td>
</tr>
<tr>
<td>9</td>
<td>( \text{atTarget} );</td>
</tr>
<tr>
<td>10</td>
<td>( \text{branch}(\text{robotLoc} = \text{targetLoc}) )</td>
</tr>
<tr>
<td>11</td>
<td>( \text{atTarget} );</td>
</tr>
<tr>
<td>12</td>
<td>( \text{moveBackward} ).</td>
</tr>
</tbody>
</table>

\[
\text{robotLoc} = \langle 3, 4 \rangle \in K_f \\
\text{robotLoc} = \langle 1, 3 \rangle \in K_f \\
\text{robotLoc} \leq \text{targetLoc} \in K_w \\
\text{robotLoc} = \langle 1, 2 \rangle \in K_f \\
\text{robotLoc} = \text{targetLoc} \in K_w \\
\text{robotLoc} = 2 \in K_f \\
\text{robotLoc} = 1 \in K_f \\
\text{robotLoc} = 2 \in K_f \\
\text{robotLoc} = 1 \in K_f \\
\text{robotLoc} = 2 \in K_f \]
Conclusions

• Knowledge-level contingent planning continues to be a promising approach for the robot task planning domains we are considering.

• Current work: scalability experiments to determine the cost/limits of our approach.

• External reasoning is a powerful mechanism for enhancing the basic operation of the planner.
  
  Future work: timeouts, limited state queries within extern calls.

• IVFs provide an interesting middles ground between representations that do not capture numerical uncertainty about fluents and those that use full belief space models.
  
  Future work: applications on robot platform.

• For more information on the project visit the JAMES website at http://james-project.eu/.
Thanks to the many researchers across the JAMES partner sites who contributed to the design, implementation, and evaluation of the JAMES robot system.

This research received funding from the European Commission’s Seventh Framework Programme under Grant No. 270435.
References


