An Extension of Knowledge-Level Planning to Interval-Valued Functions

Ron Petrick

School of Informatics
University of Edinburgh
Edinburgh, Scotland, United Kingdom
rpetrick@inf.ed.ac.uk

AAAI 2011 Workshop on Generalized Planning

San Francisco, USA

8 August 2011
Motivation

• An agent operating in a dynamic world must often do so with **incomplete information** about its environment, and
  – Make decisions based on what it knows or believes,
  – Reason about the effects of its actions to build plans, and
  – Gather information about the world through sensing.

• **Numerical** information often arises in real-world planning contexts:
  – State properties (e.g., the robot is 10m from the wall),
  – Resources (e.g., ensure the robot has enough fuel for the task)
  – Constraints (e.g., only grasp an object if its radius is < 10cm)
  – Arithmetic operations (e.g., a move action advances the robot 1m)

• Existing planners tend to work with only limited forms of numerical information.

⇒ This work: a knowledge-level approach to contingent planning with numerical information and sensing actions using the **PKS planner** (Petrick and Bacchus 2002, 2004).
Knowledge of numeric information

- Reasoning about sensing and action requires the ability to reason effectively about knowledge/belief.
- Good formal representations exist, e.g., possible worlds, belief states, etc.
- The problem: representations based on possible worlds require reasoning about sets of worlds that capture all the possible ways the agent’s knowledge could be configured.
- Example: if a function \( f \) could map to any natural number between 1–100 then modelling this (incomplete) knowledge requires a set of 100 possible worlds, one for each possible mapping of \( f \):

\[
\begin{align*}
  w_1 &: f = 1, \\
  w_2 &: f = 2, \\
  \vdots \\
  w_{100} &: f = 100.
\end{align*}
\]

- What about functions with open-ended ranges or functions that map to \( \mathbb{R} \)? How do we plan with these representations?
Reasoning without possible worlds

- Possible worlds can be avoided by restricting the types of knowledge that can be represented, leading to more tractable reasoning (Demolombe and Pozos Parra 2000; Soutchanski 2001; Petrick and Levesque 2002; Liu and Levesque 2005; Petrick 2006; Vassos and Levesque 2007, ...).

- Uncertain numerical information can be modelled in a compact form using interval-valued functions (Funge 1998) 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.

- The use of interval-valued representations in planning is limited, e.g.,
  - Time (Edelkamp 2002; Frank and Jónsson 2003; Laborie 2003, ...),
  - Numeric properties (Poggioni et al. 2003).

⇒ We focus on the problem of using interval-valued representations in the context of contingent planning with sensing actions.
Interval-valued functions and knowledge

• In this work, we use interval-valued functions 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 \( u \).

• Each interval-valued function is associated with an underlying number system \( X \) (e.g., \( X \in \{ \mathbb{R}, \mathbb{N}, \mathbb{Z}, \ldots \} \)), where \( u, v \in X \) and \( u \leq v \).

• The special interval \( \langle \bot, \top \rangle \) denotes the maximal interval for \( X \), e.g.,

\[ \langle \bot, \top \rangle \overset{\text{def}}{=} \langle -\infty, \infty \rangle \text{ for } \mathbb{R}. \]

Functions mapped to \( \langle \bot, \top \rangle \) are completely unknown meaning all mappings are considered possible.
Planning with Knowledge and Sensing

• We will extend PKS, a knowledge-level conditional planner that builds plans based on what an agent knows (Petrick and Bacchus 2002, 2004).

• PKS uses a collection of five databases, each of which models 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 ($KB$).

• A primitive query language is used to ask simple questions about the planner’s knowledge state.

• 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.

• During planning, actions update $DB \implies$ update $KB$. 
Representing knowledge in PKS

• $K_f$: knowledge of positive and negative facts (not closed world!)
  \[
  \phi \in K_f : \text{KB includes } K(\phi)
  \]

• $K_w$: knowledge of binary sensing effects
  \[
  \phi \in K_w : \text{KB includes } K(\phi) \lor K(\neg\phi)
  \]

• $K_v$: knowledge of function values, multi-valued sensing effects
  \[
  f \in K_v : \text{KB includes } \exists v. K(f = v)
  \]

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

• $LCW$: local closed world information (Etzioni et al. 1994)
Representing actions in PKS

<table>
<thead>
<tr>
<th>Action</th>
<th>Preconditions</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>readPaper</td>
<td>$K(\text{havePaper})$</td>
<td>$\text{add}(K_v, \text{phoneNumber})$</td>
</tr>
</tbody>
</table>
| dial      | $K_v(\text{phoneNumber})$ | $\text{add}(K_f, \text{dialledOk})$  
                                           $\text{add}(K_w, \text{connected})$ |

- PKS actions are based on an extension of STRIPS.
- New knowledge states are easily computed by forward chaining:
  - Evaluate preconditions against a set of databases $\text{DB}$ (a $\text{KB}$),
  - Effects update $\text{DB} \Rightarrow \text{update KB}$.
- Plans are generated by searching the space of database states.
Example: planning in PKS

1. $K_f$ havePaper
2. $K_f$ havePaper
3. $K_v$ phoneNumber
4. dial
5. $K_f$ havePaper dialledOk
6. $K_v$ phoneNumber
7. $K_w$ connected
8. $K_f$ havePaper dialledOk
9. $K_v$ phoneNumber
10. $K_w$ connected

(readPaper) $K^+$

(dial) $K^-$
Interval-valued functions in PKS

1. $K_f$ is extended to allow interval-valued functions to be represented. E.g.,

$$f(\vec{c}) = \langle u, v \rangle \in K_f$$

means $f(\vec{c})$ is known to map to a value between $u$ and $v$ (inclusive), where $u$ and $v$ are ground constants.

2. $K_v$ can already be used to represent the effects of actions that sense the definite values of functions. E.g.,

$$f(\vec{c}) \in K_v$$

means that “the value of $f(\vec{c})$ is known at plan time” and that $f(\vec{c})$ can be used as a run-time variable.

We use an interval schema to model noisy sensed information. E.g.,

$$f(\vec{c}) : \langle x - 1, x + 1 \rangle \in K_v$$

means that $f(\vec{c})$ is known and in the range $x \pm 1$ for some $x$. 
Interval-valued functions in PKS ... (2)

3. $K_w$ is extended to allow certain numeric relations to be explicitly represented and 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

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

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

$K^+\text{branch}: f > 5 \in K_f, f = \langle 6, 10 \rangle \in K_f,$

$K^-\text{branch}: f \leq 5 \in K_f, f = \langle 3, 5 \rangle \in K_f.$
4. Actions are extended to allow numeric preconditions and effects to be specified. E.g., a numeric precondition such as

$$K(f > 3)$$

will only evaluate as true given $$f = \langle u, v \rangle \in K_f$$ if $$u > 3$$.

Numeric database updates have the form

$$add(K_f, f := f \pm d)$$

where $$d$$ is a numeric constant or an interval. E.g.,

$$f = \langle 3, 5 \rangle \in K_f \Rightarrow add(K_f, f := f + 2) \Rightarrow f = \langle 5, 7 \rangle \in K_f,$$

$$f = \langle 3, 5 \rangle \in K_f \Rightarrow add(K_f, f := f + \langle 1, 2 \rangle) \Rightarrow f = \langle 4, 7 \rangle \in K_f.$$  

Interval schemata are updated in a similar fashion.
Example 1

<table>
<thead>
<tr>
<th>Action</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>moveForward</code></td>
<td>$add(K_f, \text{robotLoc} := \text{robotLoc} - 1)$</td>
</tr>
<tr>
<td><code>moveBackward</code></td>
<td>$add(K_f, \text{robotLoc} := \text{robotLoc} + 1)$</td>
</tr>
<tr>
<td><code>atTarget</code></td>
<td>$add(K_w, \text{robotLoc} = \text{targetLoc})$</td>
</tr>
</tbody>
</table>

• Domain properties: $\text{robotLoc}$ is the distance from a robot to a wall, $\text{targetLoc}$ is the desired location of the robot.

• Movement actions: `moveForward` one step, `moveBackward` one step.

• Sensing action: `atTarget` determines whether or not the robot is at the target location.

• Initial knowledge: $\text{robotLoc} = \langle 3, 5 \rangle \in K_f$, $\text{targetLoc} = 2 \in K_f$.

• Goal*: $K(\text{robotLoc} = \text{targetLoc})$. 
Example 1 ...

• One solution generated by PKS is the conditional plan:

| 0 | robotLoc = ⟨3, 5⟩ ∈ K_f |
| 1 | moveForward ; |
| 2 | atTarget ; |
| 3 | branch(robotLoc = targetLoc) |
| 4 | K^+ : nil. |
| 5 | K^- : |
|    | moveForward ; |
| 6 | atTarget ; |
| 7 | branch(robotLoc = targetLoc) |
| 8 | K^+ : nil. |
| 9 | K^- : |
|    | moveForward. |
Example 2

<table>
<thead>
<tr>
<th>Action</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>noisyForward</td>
<td>( \text{add}(K_f, \text{robotLoc} := \text{robotLoc} - \langle 1, 2 \rangle) )</td>
</tr>
<tr>
<td>withinTarget</td>
<td>( \text{add}(K_w, \text{robotLoc} \leq \text{targetLoc}) )</td>
</tr>
</tbody>
</table>

- New movement actions: \( \text{noisyForward} \) replaces \( \text{moveForward} \).
- New sensing action: \( \text{withinTarget} \) determines whether or not the robot’s location is closer than or equal to the target location.
- Initial knowledge: \( \text{robotLoc} = \langle 3, 4 \rangle \in K_f, \text{targetLoc} = 2 \in K_f \).
- Goal*: \( K(\text{robotLoc} = \text{targetLoc}) \).
Example 2 ...

- One solution generated by PKS is the conditional plan:

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>noisyForward ;</td>
</tr>
<tr>
<td>1</td>
<td>withinTarget ;</td>
</tr>
<tr>
<td>2</td>
<td>branch(robotLoc ≤ targetLoc)</td>
</tr>
<tr>
<td>3</td>
<td>K⁺ :</td>
</tr>
<tr>
<td>4</td>
<td>atTarget ;</td>
</tr>
<tr>
<td>5</td>
<td>branch(robotLoc = targetLoc)</td>
</tr>
<tr>
<td>6</td>
<td>K⁺ : nil.</td>
</tr>
<tr>
<td>7</td>
<td>K⁻ :</td>
</tr>
<tr>
<td>8</td>
<td>moveBackward.</td>
</tr>
<tr>
<td>9</td>
<td>K⁻ :</td>
</tr>
<tr>
<td>10</td>
<td>noisyForward ;</td>
</tr>
<tr>
<td>11</td>
<td>atTarget ;</td>
</tr>
<tr>
<td>12</td>
<td>branch(robotLoc = targetLoc)</td>
</tr>
<tr>
<td>13</td>
<td>K⁺ : nil</td>
</tr>
<tr>
<td>14</td>
<td>K⁻ :</td>
</tr>
<tr>
<td>15</td>
<td>moveBackward.</td>
</tr>
</tbody>
</table>
```

- robotLoc = ⟨3, 4⟩ ∈ K_f
- robotLoc = ⟨1, 3⟩ ∈ K_f
- robotLoc ≤ targetLoc ∈ K_w
- robotLoc = ⟨1, 2⟩ ∈ K_f
- robotLoc = targetLoc ∈ K_w
- robotLoc = 2 ∈ K_f
- robotLoc = 1 ∈ K_f
- robotLoc = 3 ∈ K_f
- robotLoc = ⟨1, 2⟩ ∈ K_f
- robotLoc = targetLoc ∈ K_w
- robotLoc = 2 ∈ K_f
- robotLoc = 1 ∈ K_f
- robotLoc = 2 ∈ K_f
Example 3

<table>
<thead>
<tr>
<th>Action</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>noisyLocation</td>
<td>$\text{add}(K_v, \text{robotLoc} : \langle x, x + 1 \rangle)$</td>
</tr>
</tbody>
</table>

- Movement actions: moveBackward.
- Sensing action: noisyLocation provides a range of two possible locations for the robot.
- Initial knowledge: $\text{targetLoc} = 2 \in K_f$, robotLoc is unknown.
- Goal: $K(\text{robotLoc} \geq 2)$.
- One solution produced by PKS is the plan:

1. noisyLocation; robotLoc = $\langle x, x + 1 \rangle \in K_v$
2. moveBackward; robotLoc = $\langle x + 1, x + 2 \rangle \in K_v$
3. moveBackward. robotLoc = $\langle x + 2, x + 3 \rangle \in K_v$

Since we are working in $\mathbb{N}$, it must be the case that $x \geq 0$. As a result, in the final step of the plan $x + 2 \geq 2$, satisfying the goal.
Conclusions and future work

• Interval-valued functions provide an interesting middle ground between those representations that do not capture uncertainty about numerical fluents and those that use possible world models or probabilistic distributions.

• Intervals provide a good fit with the restricted representations already used by PKS: they can take advantage of PKS’s ability to work at the knowledge level, and also provide a simple way of decomposing reasoning into subcases resulting from building conditional plans.

• Ongoing and future work:
  – Expanding the prototype implementation, with a focus on tracking uncertain information through sensing actions,
  – Establishing formal correctness results in the situation calculus,
  – Extending arithmetic interval operations beyond + and −,
  – Using interval sets to permit certain types of disjoint mappings, e.g., \( f = \{ \langle 2, 5 \rangle, \langle 8, 8 \rangle, \langle 12, 16 \rangle \} \),
  – Exploring an extension to planning with loops (Levesque 2005).


