(Dialogue) Planning with Knowledge and Sensing

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

Universität des Saarlandes, Saarbrücken, Germany

5 March 2009
Travelling to Saarbrücken (Scenario I)

“I want to take the train from München to Saarbrücken.”

Go to the station, check the timetable, go to the track, board the train, . . . , enjoy Saarbrücken!
Travelling to Saarbrücken (Scenario II)

“I want to take the train from München to Saarbrücken.”

Go to the station, ask someone for information, go to the track, board the train, . . . , enjoy Saarbrücken!
## Two plans

<table>
<thead>
<tr>
<th>Scenario I</th>
<th>Scenario II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go to the station</td>
<td>Go to the station</td>
</tr>
<tr>
<td><strong>Check the timetable</strong></td>
<td><strong>Ask someone for information</strong></td>
</tr>
<tr>
<td>Go to the track</td>
<td>Go to the track</td>
</tr>
<tr>
<td>Board the train</td>
<td>Board the train</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Enjoy Saarbrücken!</td>
<td>Enjoy Saarbrücken!</td>
</tr>
</tbody>
</table>

### Action step  
### Dialogue step (Speech act)

⇒ Both actions serve as information gathering steps in the plan

⇒ Planning **dialogue actions** can be viewed as an instance of the general AI problem of **planning with incomplete information and sensing**
Previous approaches


• Early approaches suffered due to inefficient planning techniques

• Recent work has tended to separate action planning and dialogue planning and has focused on specialized approaches, e.g., finite state machines, information state, rule-based approaches to speech act theories, dialogue games, . . .

• There has been a renewed interest in applying modern planning techniques to NLG problems, e.g., Koller & Stone (2007), Benotti (2008), Brenner & Kruijff-Korbayová (2008), Koller & Petrick (2008)
Remainder of talk

Outline

1. STRIPS planning
2. Planning with Knowledge and Sensing (PKS)
3. Extending PKS for dialogue planning

Message

⇒ Planning is cool! It may even help you solve your problems...
⇒ The same mechanisms used for ordinary action planning are applicable to problems in dialogue
⇒ Knowledge-level planning may be a useful tool for speech act based dialogue planning
Planning

• Automated AI planning techniques are good at building goal-directed plans of action under many challenging conditions, given a suitable description of the domain

• Planning problem
  – A representation of the properties and objects in the world and/or the agent’s knowledge, usually described in a logical language
  – A set of state transforming actions
  – A description of the initial world/knowledge state
  – A set of goal conditions to be achieved

• A plan is a sequence of actions that when applied to the initial state transforms the state in such a way that the resulting state satisfies the goal conditions
STRIPS (Fikes & Nilsson, 1971)

• A **world state** is represented by a **closed world** database \( \mathcal{D} \) and negation as failure
  
  – \( \phi \) is true if \( \phi \in \mathcal{D} \)
  
  – \( \neg \phi \) is true if \( \phi \notin \mathcal{D} \), where \( \phi \) is a ground literal.

• **Actions** are the sole means of change in the world

• An action’s **preconditions** specify the conditions under which an action can be applied, evaluated by querying \( \mathcal{D} \) (qualification problem)

• An action’s **effects** specify the changes the action makes to the world, applied by updating \( \mathcal{D} \)
  
  – **Add list**: properties \( A \) makes true are added to \( \mathcal{D} \)
  
  – **Delete list**: properties \( A \) makes false are removed from \( \mathcal{D} \)
  
  – All other properties are unchanged (frame problem) (McCarthy & Hayes, 1969)
STRIPS actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Preconditions</th>
<th>Add list</th>
<th>Delete list</th>
</tr>
</thead>
<tbody>
<tr>
<td>pickup(x)</td>
<td>handEmpty</td>
<td>holding(x)</td>
<td>handEmpty</td>
</tr>
<tr>
<td></td>
<td>onTable(x)</td>
<td></td>
<td>onTable(x)</td>
</tr>
<tr>
<td>dropInBox(x, y)</td>
<td>holding(x)</td>
<td>inBox(x, y)</td>
<td>holding(x)</td>
</tr>
<tr>
<td></td>
<td>box(y)</td>
<td>handEmpty</td>
<td>empty(y)</td>
</tr>
</tbody>
</table>

- Applying the effects of an instantiated action $A$ to a database $D$ updates the database to produce the state resulting from the execution of $A$.

- We can generate plans by chaining together fully instantiated STRIPS actions.
Planning with STRIPS actions

- Say the goal is to find a plan that satisfies \( \text{inBox}(o1, b1) \), i.e., to achieve a state where object \( o1 \) is in box \( b1 \).
- The goal is satisfied in the final state since \( \text{inBox}(o1, b1) \in \mathcal{D} \).
- The resulting plan is the action sequence:

  \[
  \text{[pickup}(o1); \text{dropInBox}(o1, b1)]
  \]
Recent STRIPS planning

- STRIPS forms the core of PDDL (McDermott *et al.*, 1998), the language of many modern planners and the language of the International Planning Competition.

- Many efficient planners exist, e.g., FF (Hoffmann & Nebel, 2001), SGPlan (Hsu *et al.*, 2006), ... 

- “Classical” planning with complete knowledge and deterministic action effects is the most popular track in the International Planning Competition. See [http://ipc.icaps-conference.org/](http://ipc.icaps-conference.org/)
Planning with incomplete information

• The classical STRIPS assumptions of complete knowledge are not realistic in many domains

• In general, an agent operating in a dynamic world must do so with incomplete information about its environment, by
  – Making decisions based on what it knows or believes
  – Reasoning about the effects of its actions
  – Gathering information about the world through sensing

⇒ Reasoning about sensing requires the ability to reason effectively about knowledge
Planning with Knowledge and Sensing

- PKS is a “knowledge-level” planner that builds plans based on what is known (Petrick & Bacchus, 2002; 2004)
- Knowledge is updated in a STRIPS-like manner, however, actions are modelled by their effects on the planner’s knowledge state
- Plans can be constructed with conditional branches to manage indefinite information (contingent planning)
- Representation supports non-propositional features like functions, run-time variables, and simple program structures
- Knowledge is restricted for tractable reasoning
- Efficiency has been demonstrated on traditional benchmarks and novel scenarios
Representing knowledge in PKS

• Based on an extension of the STRIPS representation

• PKS uses a collection of five databases, each of which is restricted to a particular types of knowledge: $K_f, K_v, K_w, K_x, LCW$

• Knowledge is correct but incomplete

• The contents of the databases have a fixed formal translation to formulae in a modal logic of knowledge

• Given a set of the five databases ($DB$), the translation formally defines the agent’s knowledge state ($KB$)

• Planning: actions update $DB \Rightarrow$ update $KB$
PKS databases

• $K_f$: knowledge of positive and negative facts (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

\[ \phi : \text{the planner "knows whether" } \phi \]

• $K_v$: knowledge of function values, multi-valued sensing effects

\[ f : \text{the planner "knows the value" of } f \]

• $K_x$: exclusive-or knowledge

\[ (\ell_1 | \ell_2 | \ldots | \ell_n) : \text{exactly one of the } \ell_i \text{ must be true} \]

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

• A primitive query language is used to ask questions about the planner’s knowledge state
  - $K(\alpha)$, is $\alpha$ known to be true?
  - $K_v(t)$, is the value of $t$ known?
  - $K_w(\alpha)$, is $\alpha$ known to be true or known to be false?
  - The negation of the above queries

• A sound, but incomplete, inference procedure checks the database contents to determine the truth of a query
  - Restricted reasoning (but more than single database lookup)
  - Efficient (usually)

• Used to evaluate preconditions, conditional rules, and goals
## PKS actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Preconditions</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>readPaper</td>
<td>KhavePaper</td>
<td>add($K_v$, phoneNumber)</td>
</tr>
<tr>
<td>dial</td>
<td>$K_v$phoneNumber</td>
<td>add($K_f$, dialledOk) add($K_w$, connected)</td>
</tr>
</tbody>
</table>

- Easy to compute new knowledge states by forward chaining
  - Evaluate preconditions against a set of databases $DB$ ($KB$)
  - Effects update $DB \Rightarrow$ update $KB$

- Plans are generated by searching the space of database states
PKS planning

- Conditional branches are formed from $K_w$ (and $K_v$) formulae
- Actions can be parametrized with ground $K_v$ terms
- Goal conditions must be satisfied along every plan branch
What about dialogue?
Dialogue planning with speech acts

- Motivation: certain types of dialogue actions can be viewed as an instance of the general problem of planning with incomplete information and sensing
- Can we apply knowledge-level planning techniques to this problem?
  - Dialogues involve multiple participants
  - Actions in a plan correspond to dialogue acts, e.g., *ask*, *tell*, ...  
  - Plans specify mixed-initiative discourse among participants
- Formal representations exist: many logical languages for reasoning about actions and change, e.g., (Moore, 1985), (Scherl & Levesque, 1994; 2003), (Stone, 1998), (Steedman, 1997; 2002), ...

⇒ What kinds of changes do we need to make to PKS? How tractable is the reasoning? What types of domains can we model?
Knowledge requirements for *ask* and *tell*

**R1.** “If X doesn’t know *p* and X knows Y does, X can ask Y about it.”

⇒ Knowledge-level preconditions for *ask*

**R2.** “If X asks Y about *p*, it makes it common ground X doesn’t know it.”

⇒ Knowledge-level effects for *ask*

**R3.** “If X knows *p*, and X knows *p* is not common ground, X can tell Y *p*.”

⇒ Knowledge-level preconditions for *tell*

**R4.** “If X tells Y *p*, Y stops not knowing it and starts to know it.”

⇒ Knowledge-level effects for *tell*
Participants and common ground

• We require modalities for referencing dialogue participants and common ground

\[
\begin{align*}
[S] & \quad \text{Speaker supposition} \\
[H] & \quad \text{Hearer supposition} \\
[X], [Y], \ldots & \quad \text{Other participant/agent suppositions} \\
[C_{XY}] & \quad \text{Common ground between X and Y}
\end{align*}
\]

Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[S] p</td>
<td>“The speaker supposes ( p ).”</td>
</tr>
<tr>
<td>[S] [H] p</td>
<td>“The speaker supposes the hearer supposes ( p ).”</td>
</tr>
<tr>
<td>[H] [C_{SH}] [S] p</td>
<td>“The hearer supposes it’s common ground between the speaker and hearer that the speaker supposes ( p ).”</td>
</tr>
</tbody>
</table>
Knowledge assertions

• We restrict the “content” of knowledge expressions to simple PKS-style knowledge assertions

\[
\begin{align*}
K_p & \quad \text{“Know } p\text{”} \\
K_v t & \quad \text{“Know the value of } t\text{”} \\
K_w p & \quad \text{“Know whether } p\text{”}
\end{align*}
\]

• Recall that \( K_v \) and \( K_w \) represent indefinite information, e.g., information returned by sensing actions

• Combine modalities with knowledge assertions

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>([S] \neg K_{open}(obj1))</td>
</tr>
<tr>
<td>([S] [H] K_v.train)</td>
</tr>
<tr>
<td>([S] [C_{SH}] K_w.connected)</td>
</tr>
</tbody>
</table>
Extra reasoning rules

• Rules for reasoning about speaker-hearer suppositions and common ground modalities (Steedman & Petrick, 2007)

A1. $[X] \phi \Rightarrow \phi$  
Supposition Veridicality
A2. $[X] \neg \phi \Rightarrow \neg [X] \phi$  
Supposition Consistency
A3. $\neg [X] \phi \Rightarrow [X] \neg [X] \phi$  
Negative Introspection
A4. $[C] \phi \Leftrightarrow ([S] [C] \phi \land [H] [C] \phi)$  
Common Ground
A5. $[X] [C] \phi \Rightarrow [X] \phi$  
Common Ground Veridicality

• We require rules similar to these to augment PKS’s standard inference procedure.

• How general do these rules need to be?
Example: taking a train
Initial knowledge

F1. “I don’t know what train I will catch.”

\([S] \neg K_v^{\text{train}}\)

F2. “If I know what time it is then I know what train I will catch.”

\([S] K_v^{\text{time}} \Rightarrow add([S] K_v^{\text{train}})\)

F3. “I suppose you know what time it is.”

\([S] [H] K_v^{\text{time}}\)

F4. “I suppose it’s not common ground I don’t know what time it is.”

\([S] \neg [C_{SH}] \neg [S] K_v^{\text{time}}\)
Knowledge-level dialogue actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Preconditions</th>
<th>Effects</th>
</tr>
</thead>
</table>
| `ask(X, Y, p)` | $\neg [X] p$  
$[X] [Y] p$ | `add([C_{XY}] \neg [X] p)` |
| `tell(X, Y, p)` | $[X] p$  
$[X] \neg [C_{XY}] p$ | `add([Y] p)` |

- We can encode the knowledge requirements for dialogue actions like `ask` and `tell` in terms of their preconditions and effects.
- These look like planning operators...
- We can build plans by using action chaining together with our extra rules for reasoning about modalities.
Planning a direct speech act

<table>
<thead>
<tr>
<th>Goal: $[S] K_v train$ (I need to know which train to catch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D1) $\Rightarrow [H] K_v time$</td>
</tr>
<tr>
<td>(D2) $\Rightarrow \neg [S] K_v time$</td>
</tr>
<tr>
<td>(D3) $\Rightarrow$ Apply action: $ask(S, H, K_v time)$</td>
</tr>
<tr>
<td>(D4) $\Rightarrow [C_{SH}] \neg [S] K_v time$</td>
</tr>
<tr>
<td>(D5) $\Rightarrow$ Apply action: $tell(H, S, K_v time)$</td>
</tr>
<tr>
<td>(D6) $\Rightarrow [S] K_v time$</td>
</tr>
<tr>
<td>(D7) $\Rightarrow [S] K_v train$</td>
</tr>
</tbody>
</table>

Plan: $[ask(S, H, K_v time), tell(H, S, K_v time)]$
Planning an indirect speech act

<table>
<thead>
<tr>
<th>Goal: ([S] \ K_v\text{train}) (I need to know which train to catch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D1) (\Rightarrow [S] \neg [S] \ K_v\text{time})</td>
</tr>
<tr>
<td>(D2) (\Rightarrow [S] \neg [C_{SH}] \neg [S] \ K_v\text{time})</td>
</tr>
<tr>
<td>(D3) (\Rightarrow \text{Apply action: } tell(S, H, \neg [S] \ K_v\text{time}))</td>
</tr>
<tr>
<td>(D4) (\Rightarrow [C_{SH}] \neg [S] \ K_v\text{time})</td>
</tr>
<tr>
<td>(\Rightarrow \ldots)</td>
</tr>
<tr>
<td>(\Rightarrow \text{Apply action: } tell(H, S, \ K_v\text{time}))</td>
</tr>
<tr>
<td>(\Rightarrow \ldots)</td>
</tr>
</tbody>
</table>

Plan: \([tell(S, H, \neg [S] \ K_v\text{time}), tell(H, S, \ K_v\text{time})]\)
Observations about plan generation

- Plan generation takes place in the space of multi-agent plans
  - No reasoning is done about other participants’ goals
  - Cannot guarantee other participants’ actions
  - Planning is offline

- Approach is driven by the knowledge state, i.e., what the planning agent knows about the world and the other agents’ beliefs

- Both direct and indirect speech acts result from the same mechanisms for reasoning about knowledge and common ground

- All speech acts are indirect in the sense they involving inference
Dialogue planning with PKS

- Work is ongoing to adapt/extend existing PKS mechanisms
  - “Index” queries and updates by participant and common ground modalities
  - All knowledge is stored and interpreted relative to the planner’s knowledge state, e.g., \( [H] K_v\text{time} \) is really \([S][H] K_v\text{time}\)
  - Restrict form of allowable knowledge assertions, e.g.,
    \[
    [X] [Y] \cdots [X] \phi_{PKS}
    \]
  - Manage the planner’s knowledge of common ground as an instance of local closed world (LCW) information
    (Etzioni et al., 1994)
  - Implement new inference rules

⇒ What kinds of problems can we represent and solve while maintaining tractable reasoning?
PACO-PLUS: Dialogue in robot domains

Robot1: Let’s make breakfast.
Robot2: I don’t know how to make breakfast.
Robot1: To make breakfast we must bring the cereal and the milk to the sideboard.
Robot2: Is the cereal at the sideboard?
Robot1: No.
Robot2: Where is the cereal?
Robot1: The cereal is in the cupboard.
Robot2: Is the milk at the sideboard?
Robot1: No.
Robot2: Where is the milk?
Robot1: The milk is in the fridge.
Robot2: Okay. I suggest I go to the cupboard, pickup the cereal, bring it to the sideboard, then go the fridge, pickup the milk, and bring it to the sideboard.

⇒ We are interested in generating dialogue plans in the same domains as ordinary robot action plans (e.g., plan explanation, cooperative information gathering for action plan construction), using the same underlying planning techniques: http://www.paco-plus.org/
Conclusions and future work

• Certain speech acts can be viewed as instances of sensing actions
• The preconditions and effects of dialogue actions like *ask* and *tell* can be encoded as knowledge-level planning operators
• Plans can be generated by reasoning about the planning agent’s evolving knowledge state, without reference to specific conversational rules (except common ground consistency)
• Direct and indirect speech acts result from the same process
• We are currently extending PKS to generate such plans automatically
• Future work: evaluation
• Is planning useful for your NLG problems?
References


References...(3)


