Dialogue as Planning with Knowledge and Sensing

Ron Petrick
Joint work with Mark Steedman

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

Workshop on Situated Understanding of Intention
University of Pennsylvania, Philadelphia, PA, USA

23–24 July 2009
Travelling to Philadelphia (Plan I)

“I want to fly to Philadelphia.”

Buy ticket, go to the airport, check the departures board, go to the gate, board the plane, . . . , enjoy Philadelphia.
Travelling to Philadelphia (Plan II)

“I want to fly to Philadelphia.”

Buy ticket, go to the airport, ask someone for gate information, go to the gate, board the plane, . . . , enjoy Philadelphia.
## Two plans

<table>
<thead>
<tr>
<th>Action step</th>
<th>Dialogue step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both actions serve as information gathering steps in the plan.</td>
<td>In this work we investigate the problem of planning <strong>dialogue actions</strong> as an instance of the general AI problem of <strong>planning with incomplete information and sensing.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plan I</th>
<th>Plan II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy ticket</td>
<td>Buy ticket</td>
</tr>
<tr>
<td>Go to the airport</td>
<td>Go to the airport</td>
</tr>
<tr>
<td><strong>Check the departures board</strong></td>
<td><strong>Ask for gate information</strong></td>
</tr>
<tr>
<td>Go to the gate</td>
<td>Go to the gate</td>
</tr>
<tr>
<td>Board the plane</td>
<td>Board the plane</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Remainder of talk

Outline

1. STRIPS planning
2. Planning with Knowledge and Sensing (PKS)
3. Extending PKS for dialogue planning
4. PACO-PLUS project

Message

⇒ The same mechanisms used for ordinary task planning can often be used for dialogue planning.
⇒ Speech acts are modelled as planning actions and plans are constructed by reasoning about the knowledge state.
⇒ No special machinery for intent, direct vs. indirect speech acts.
⇒ To what extent can we use this approach for real dialogue?
Background and motivation


- 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, e.g., Koller & Stone (2007), Benotti (2008), Brenner & Kruijff-Korbayová (2008), Koller & Petrick (2008; 2009).
Automated planning

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

- A **planning problem** consists of:
  1. A representation of the properties and objects in the world and/or the agent’s knowledge, usually described in a logical language,
  2. A set of state transforming actions,
  3. A description of the initial world/knowledge state,
  4. 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)

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

• A **world state** is represented by a **closed world** database $\mathcal{D}$.

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

• An action’s **effects** specify the changes the action makes to the world, applied by updating $\mathcal{D}$ (and offer a solution to the **frame problem**, see (McCarthy & Hayes, 1969)).
We can generate plans by chaining together fully instantiated STRIPS actions. E.g., achieve a state where object $o1$ is in box $b1$, i.e., $inBox(o1, b1)$. STRIPS forms the core of PDDL (McDermott et al., 1998), the language of many modern planners and the International Planning Competition (IPC): http://ipc.icaps-conference.org/
• GIVE: “Generating Instructions in Virtual Environments”
• Many efficient planners exist, e.g., FF (Hoffmann & Nebel, 2001), SGPlan (Hsu et al., 2006),…
• NLG-inspired domains as challenges for the planning community.
Planning with incomplete information

- Problem: classical STRIPS assumes complete knowledge and deterministic action effects, which are not realistic in many domains.

- In general, an agent operating in a dynamic world must 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,
  - Gather information about the world (through sensing).

⇒ Reasoning about sensing requires the ability to reason effectively about the agent’s knowledge/beliefs.
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 (conditional planning).

• Representation supports non-propositional features like functions, run-time variables, and simple program structures.

• Tested on traditional benchmarks and novel scenarios.
Representing knowledge in PKS

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

- Knowledge is assumed to be correct but incomplete.

- 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 ($KB$).

- 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.
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(Kv, phoneNumber)</td>
</tr>
<tr>
<td>dial</td>
<td>KvphoneNumber</td>
<td>add(Kf, dialledOk) add(Kw, connected)</td>
</tr>
</tbody>
</table>

• PKS actions are based on an extension of STRIPS.
• Easy to compute new knowledge states by forward chaining
  – Evaluate preconditions against a set of databases DB (KB)
  – Effects update DB ⇒ update KB
• Plans are generated by searching the space of database states.
PKS planning

\[ K_f \text{ havePaper} \xrightarrow{\text{readPaper}} K_f \text{ havePaper} \xrightarrow{\text{dial}} K_f \text{ havePaper dialledOk} \]

\[ K_v \text{ phoneNumber} \]

\[ K_v \text{ connected} \]

\[ K^+ \]

\[ K^- \]

\[ K_f \text{ havePaper dialledOk connected} \]

\[ K_v \text{ phoneNumber} \]

\[ K_v \text{ phoneNumber} \]

\[ \text{write(phoneNumber)} \]
What about dialogue?
Dialogue planning with speech acts

• Motivation: certain types of dialogue acts can be viewed as instances 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? In what kinds of domains can we apply these techniques?
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

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

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

  \[
  \begin{align*}
  Kp & : \text{“Know } p\text{”} \\
  Kv_t & : \text{“Know the value of } t\text{”} \\
  Kw_p & : \text{“Know whether } p\text{”}
  \end{align*}
  \]

  \[\Rightarrow\ E.g., [S] \neg K_{open}(obj1), \ [S] [H] K_{gate}, \ [S] [C_{SH}] K_{w connected}\]
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. \([C] \phi \iff ([S] [C] \phi \land [H] [C] \phi)\)  
    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.

⇒ No specific conversational rules or rules governing intent recognition.
Example: boarding a flight
Initial knowledge

F1. “I don’t know what gate my flight leaves from.”

\[ [S] \neg K_v \text{gate} \]

F2. “If I know what time it is then I will know what gate to go to.”

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

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.
- We can build plans by using action chaining together with our extra rules for reasoning about modalities.
Plan generation

**Goal:** \([S] \ K_vgate\) (I need to know which gate to go to)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D1)</td>
<td>([H] \ K_vtime)</td>
<td>(F3),(A1)</td>
</tr>
<tr>
<td>(D2)</td>
<td>(\neg \ [S] \ K_vtime)</td>
<td>(F1),(A2),(F2)</td>
</tr>
<tr>
<td>(D3)</td>
<td><strong>Apply action:</strong> (ask(S, H, K_vtime))</td>
<td></td>
</tr>
<tr>
<td>(D4)</td>
<td>([C_{SH}] \neg \ [S] \ K_vtime)</td>
<td>(D3),(R2)</td>
</tr>
<tr>
<td>(D5)</td>
<td><strong>Apply action:</strong> (tell(H, S, K_vtime))</td>
<td></td>
</tr>
<tr>
<td>(D6)</td>
<td>([S] \ K_vtime)</td>
<td>(D5),(D2),(R4)</td>
</tr>
<tr>
<td>(D7)</td>
<td>([S] \ K_vgate)</td>
<td>(D6),(F2)</td>
</tr>
</tbody>
</table>

**Plan:** \([\text{ask}(S, H, K_vtime), \text{tell}(H, S, K_vtime)]\)
Plan generation...

**Goal:** \([S] K_v gate\) (I need to know which gate to go to)

\[(D1) \Rightarrow \Box [S] \neg [S] K_v time \quad (F1),(A2),(F2),(A3)\]
\[(D2) \Rightarrow \Box [S] \neg [C_{SH}] \neg [S] K_v time \quad (F4)\]
\[(D3) \Rightarrow \text{Apply action: } tell(S, H, \neg [S] K_v time)\]
\[(D4) \Rightarrow [C_{SH}] \neg [S] K_v time \quad (R2)\]
\[\Rightarrow \ldots\]
\[\Rightarrow \text{Apply action: } tell(H, S, K_v time)\]
\[\Rightarrow \ldots\]

**Plan:** \([tell(S, H, \neg [S] K_v time), tell(H, S, K_v time)]\)
Dialogue and plan generation

• Plan generation takes place in the space of multi-agent plans
  – No reasoning is done about other participants’ goals or intentions
  – 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.

• Work is ongoing to adapt/extend existing PKS mechanisms to generate such plans automatically.
“Perception, Action, and Cognition through learning of Object-Action Complexes”

http://www.paco-plus.org/

- Multiple robot platforms
- Portions of the robot’s high-level representation are induced from its interaction with the real world
- Task and dialogue planning using PKS
Task planning for object manipulation

**Action descriptions**

<table>
<thead>
<tr>
<th>Action Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grasp(?o,?l,?h)</td>
<td>Grasp object ?o from ?l using gripper ?h.</td>
</tr>
<tr>
<td>grasp-fromEdge(?o,?l,?h)</td>
<td>Grasp object ?o from the edge of ?l using gripper ?h.</td>
</tr>
<tr>
<td>move(?l1,?l2)</td>
<td>Move the robot from location ?l1 to location ?l2.</td>
</tr>
<tr>
<td>nudge-toEdge(?o,?l,?h)</td>
<td>Nudge flat object ?o to the edge of ?l using gripper ?h.</td>
</tr>
<tr>
<td>open(?l,?h)</td>
<td>Open ?l with gripper ?h.</td>
</tr>
<tr>
<td>open-partial(?l1,?h)</td>
<td>Partially open ?l1 with gripper ?h.</td>
</tr>
<tr>
<td>open-complete(?l,?h)</td>
<td>Finish opening ?l with gripper ?h.</td>
</tr>
<tr>
<td>close(?l,?h)</td>
<td>Close ?l with gripper ?h.</td>
</tr>
<tr>
<td>pass-object(?o,?h1,?h2)</td>
<td>Pass object ?o from gripper ?h1 to ?h2.</td>
</tr>
<tr>
<td>place-upright(?o,?l,?h)</td>
<td>Put object ?o upright at ?l using gripper ?h.</td>
</tr>
<tr>
<td>remove-from(?o,?l,?h)</td>
<td>Remove object ?o from ?l using gripper ?h.</td>
</tr>
<tr>
<td>sense-open(?o)</td>
<td>Sense whether object ?o is open or not.</td>
</tr>
</tbody>
</table>

**Example plan:** ensure the applejuice is in the fridge

place-upright(applejuice, sideboard, lefthand)
grasp(applejuice, sideboard, righthand)
move(sideboard, fridge)
open-partial(fridge, lefthand)
pass-object(applejuice, righthand, lefthand)
open-complete(fridge, righthand)
put-in(applejuice, fridge, lefthand)
close(fridge, lefthand)
Dialogue planning 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 robot task plans, using the same underlying planning techniques.
Summary and future work

• Certain speech acts can be viewed as instances of sensing actions.
• Dialogue actions like *ask* and *tell* can be encoded as knowledge-level planning operators.
• Plans can be generated by reasoning about the planner’s knowledge state, without reference to specific conversational rules or intent/goal recognition. (This work does not preclude the use of such techniques.)
• Direct and indirect speech acts result from the same processes.
• We are currently extending PKS to generate such plans automatically and are building a set of tools for integration with robot systems.
• Future work: evaluation
• To what extent can we use this approach for real dialogue?


References...(2)


References...(3)


