JAMES

JOINT ACTION FOR MULTIMODAL EMBODIED SOCIAL SYSTEMS · james-project.eu

# What would you like to drink? Knowledge-level planning for a social robot bartender

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

**Workshop on Planning, Logic, and Social Intelligence**
Technical University of Denmark, Copenhagen, Denmark, 4 April 2014

# Two people walk into a bar...

*Two people, A and B, each individually approach a bartender.*

Bartender (to A): How can I help you?

Person A: A pint of cider, please.

*Person C approaches the bartender and attracts his attention by gesturing.*

Bartender (to C): How can I help you?

Person C: I'd like a pint of bitter.

Bartender: (Serves C)

Bartender (to B): What will you have?

Person B: A glass of red wine.

Bartender: (Serves B)

Bartender: (Serves A)

# Two people walk into *another* bar...

*Two people, A and B, each individually approach a bartender.*

Bartender (to A): How can I help you?

Person A: A pint of cider, please.

*Person C approaches the bartender and attracts his attention by gesturing.*

Bartender (to C): Just a moment please.

Bartender: (Serves A)

Bartender (to B): What will you have?

Person B: A glass of red wine.

Bartender: (Serves B)

Bartender (to C): Thanks for waiting. How can I help you?

Person C: I'd like a pint of bitter.

Bartender: (Serves C)

# Two interactions

*Two people, A and B, each individually approach a bartender*

| | |
|---|---|
| Bartender (to A): | How can I help you? |
| Person A: | A pint of cider, please. |

| | |
|---|---|
| Bartender (to A): | How can I help you? |
| Person A: | A pint of cider, please. |

*Person C approaches the bartender and attracts his attention by gesturing*

| | |
|---|---|
| Bartender (to C): | How can I help you? |
| Person C: | I'd like a pint of bitter. |
| Bartender: | (Serves C) |
| Bartender (to B): | What will you have? |
| Person B: | A pint of Guinness. |
| Bartender: | (Serves B) |
| Bartender: | (Serves A) |

| | |
|---|---|
| Bartender (to C): | Wait a moment please |
| Bartender: | (Serves A) |
| Bartender (to B): | What will you have? |
| Person B: | A pint of Guinness. |
| Bartender: | (Serves B) |
| Bartender (to C): | Thanks for waiting. How can I help you? |
| Person C: | I'd like a pint of bitter. |
| Bartender: | (Serves C) |

- Both interactions result in the customers achieving their task goals.
- The first interaction is shorter.
- The second interaction can be seen to be more **socially appropriate**.

# Meet the bartender: JAMES

Image: fortiss GmbH

- Part of the **JAMES Project** (`http://james-project.eu/`), funded by the European Commission, exploring social interaction in robotics domains.

# Why social interaction?

- **Successful task interaction often relies on social interaction.**
  - May be several ways to achieve a task-based goal.
  - Appropriate social behaviour can lead to higher participant satisfaction.
- **Social interaction can be seen as an instance of joint action.**
  - Involves coordination of participant actions.
  - Inherently multimodal: speech, gesture, gaze, expression, etc.
- **Social interaction is often multi-party, dynamic, short-horizon.**
  - In contrast to one-on-one, companion-style relationships.
  - Interactions are often "one shot"; may not have an opportunity to recover from a poor interaction.

⇒ **What impact does this have at the knowledge representation and planning level?**

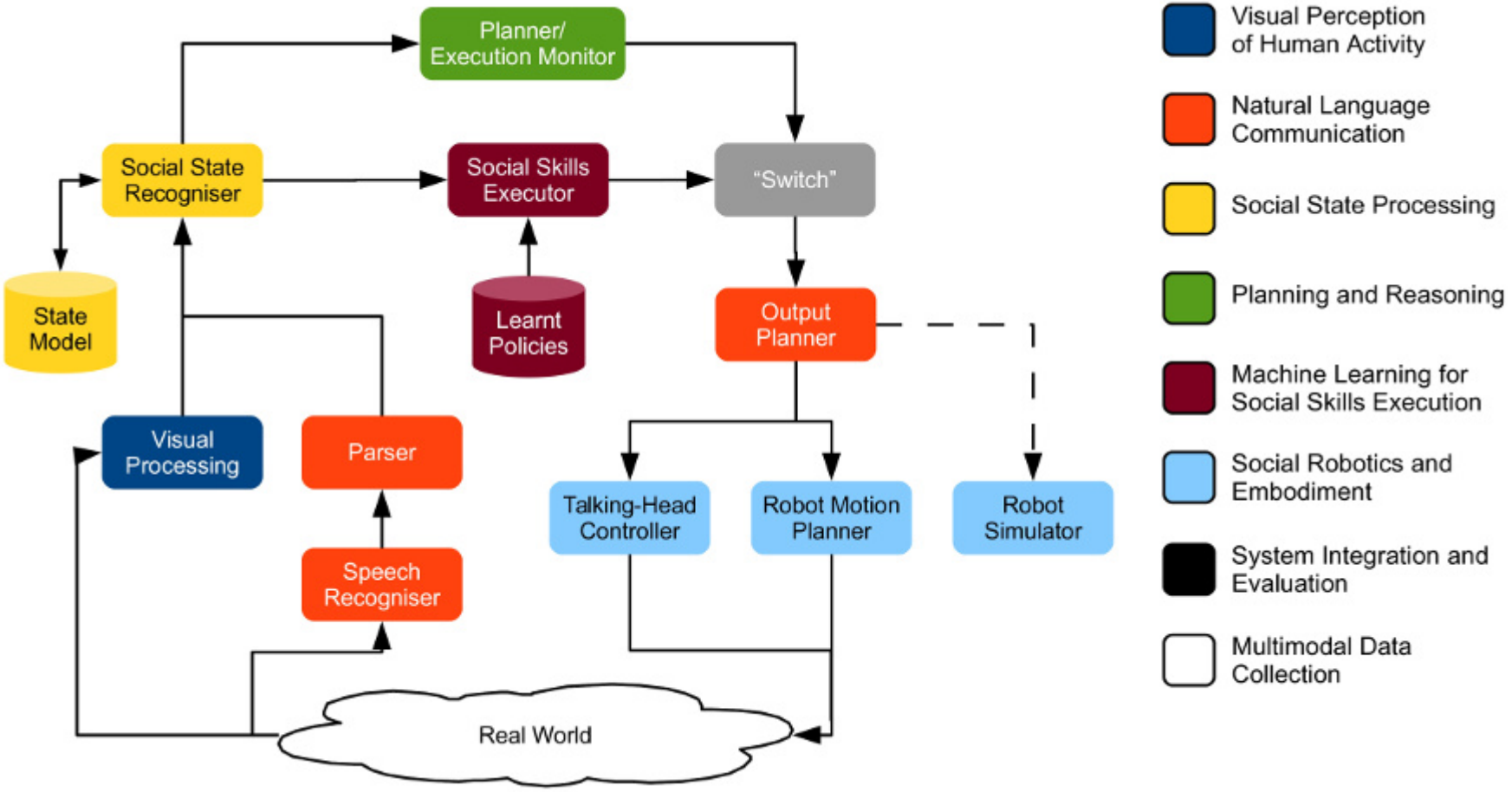
# Task-based social interaction

Image: fortiss GmbH

- Robot bartender must respond to user requests in a dynamic setting with multiple users and short interactions in German or English.
- Interactions incorporate both **task-based aspects** (e.g., ordering and serving drinks) and **social aspects** (e.g., managing multiple interactions).
- Supported activities include
  - Asking customers for drink orders
  - Handing over drinks
  - Tracking the order people arrive at the bar

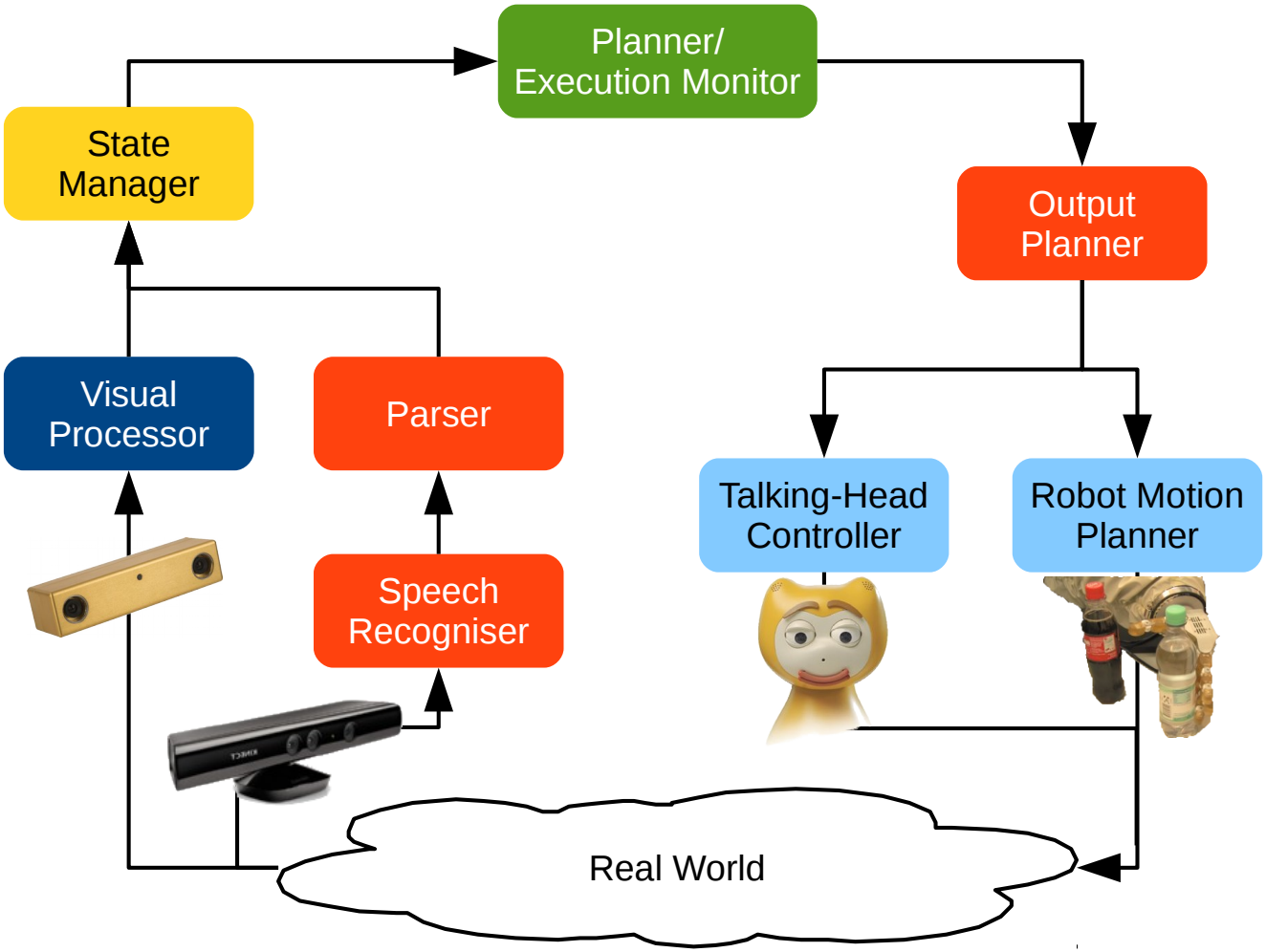
  but not…
  - Pouring drinks, handling money, small talk, …

# A multidisciplinary architecture



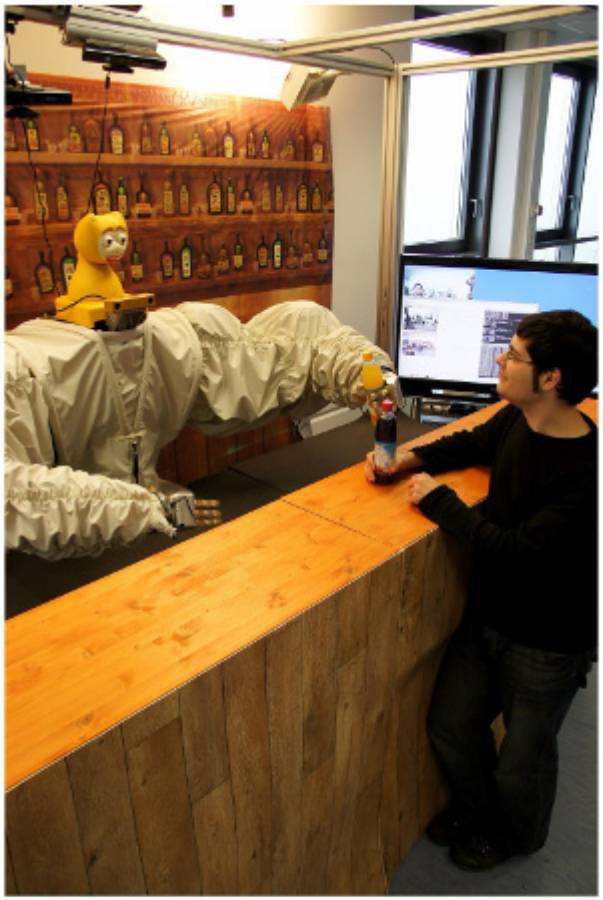

- An important aspect of the JAMES research is the collection and analysis of data collected from real bars investigating how human customers interact with human bartenders (Huth 2011; Loth et al. 2013).

# Simplified architecture

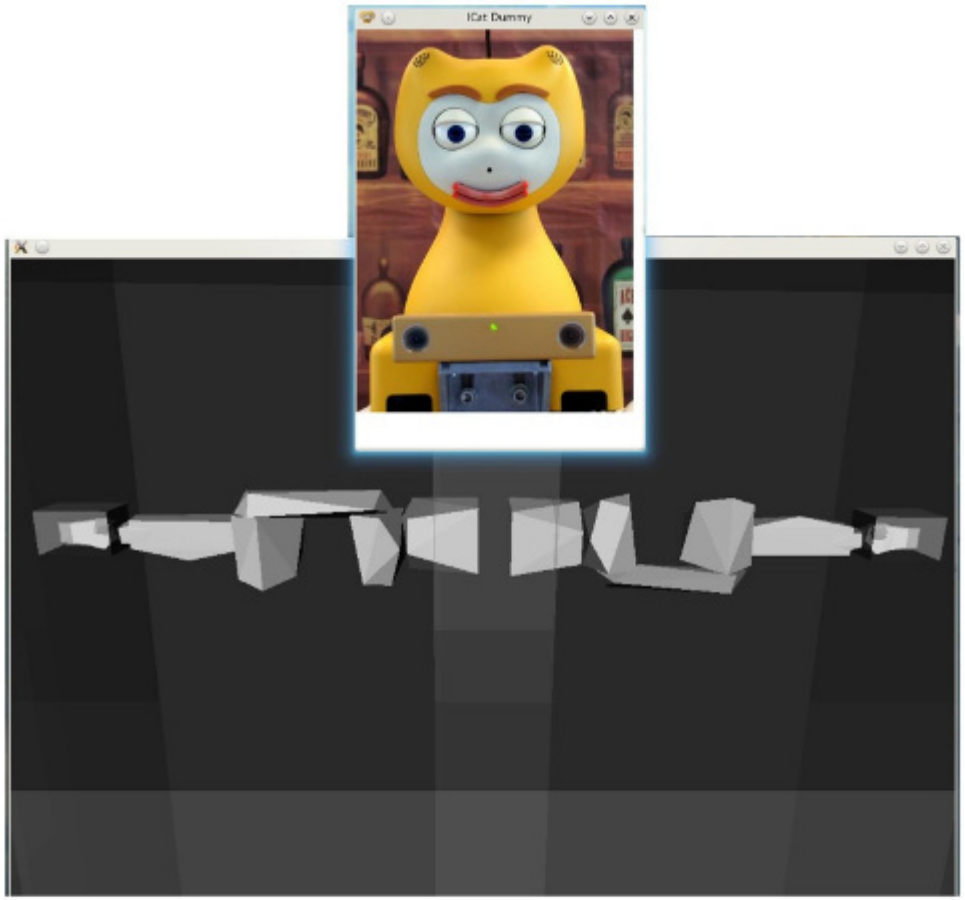



- See (Giuliani et al. 2013) for more details.

# Robot hardware



Physical hardware

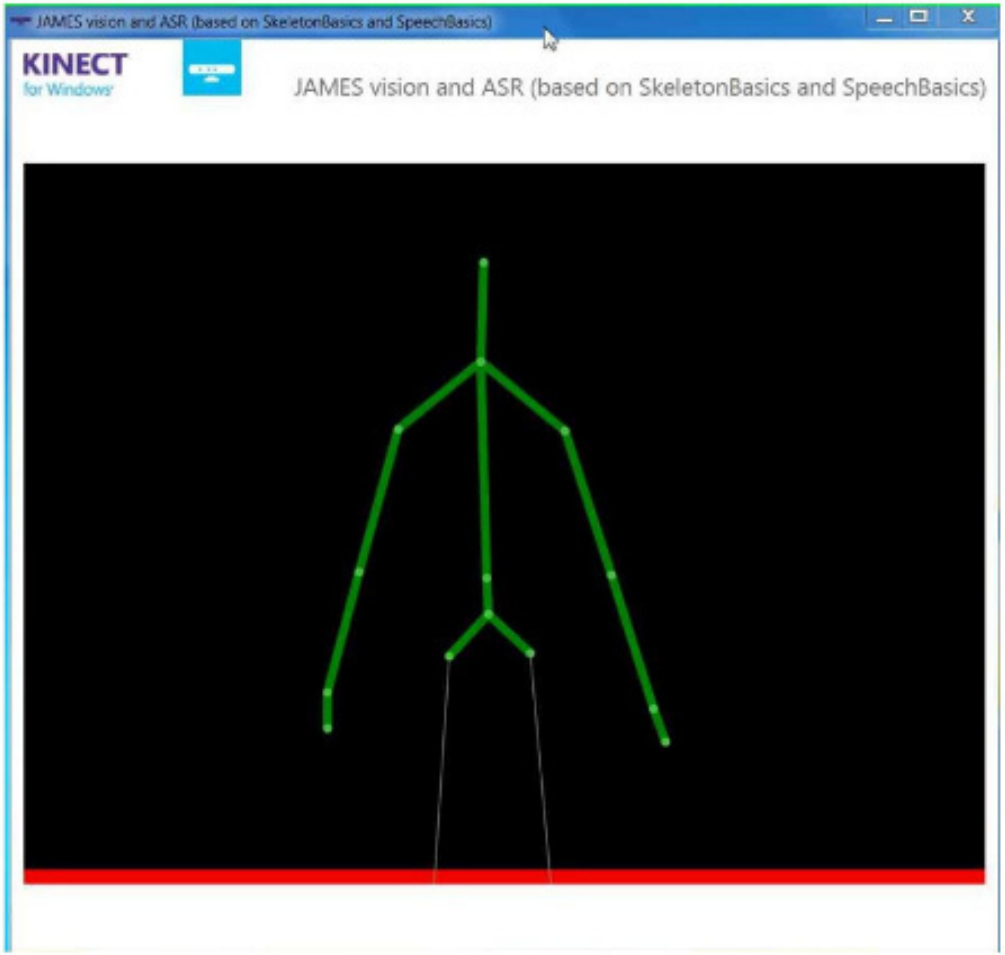



Simulated hardware

*Credit: A. Gaschler and M. Giuliani, fortiss GmbH*

# Vision system

Full system




Kinect-only

*Credit: M. Pateraki and M. Sigalas, FORTH*

# Example: asking a customer for a drink

Plan step

*ask-drink(A1)*

```
<output>
  <gesture-list>
    <gesture type="Smile"/>
  </gesture-list>
  <speech-list>
    <speech type="query" politeness="4">
      <person id="A3"/>
      <pred type="offer">
        <object type="drink"/>
      </pred>
    </speech>
  </speech-list>
</output>
```

Multimodal output specification



Parsed speech input

```
<lf>
  <node id="c1:drink" pred="water" mood="dcl" num="sg" />
</lf>
```

State update

*add(Kf, request(A1)=water))*

See (Petrick et al. 2012) for more information.

# Example: serving a drink to a customer

*K(request(A1)=water)*

*serve (A1, request(A1))*

```
<output>
  <gesture-list>
    <gesture type="Smile"/>
  </gesture-list>
  <action-list>
    <action type="give">
      <object id="idX" name="water" type="drink"/>
      <person id="A1"/>
    </action>
  </action-list>
  <speech-list>
    <speech type="inform" politeness="4">
      <person id="A1"/>
      <pred type="hand-over">
        <object type="drink" name="water" id="A1"/>
      </pred>
    </speech>
  </speech-list>
</output>
```


See (Petrick et al. 2012) for more information.

# Humans and robots as planning agents

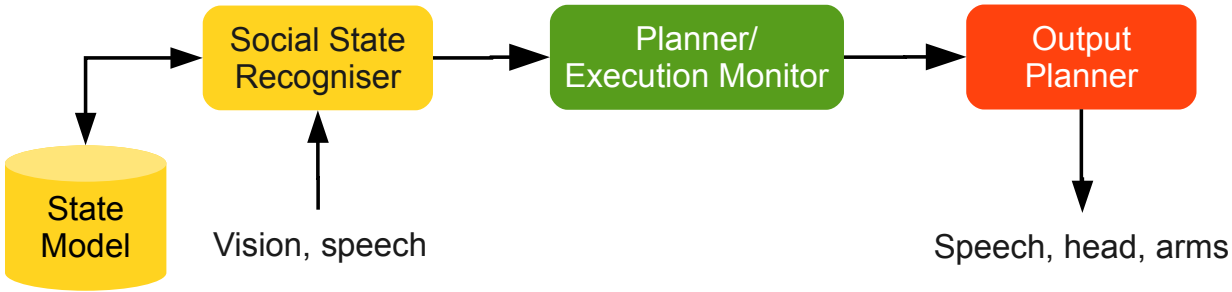
- Humans perceive and manipulate their environments by sensing, reasoning, and acting. Robots must do something similar.
- E.g., planning a trip, cooking a meal, communicating with other agents.
- Achieving particular goals in the world often involves reasoning about:
  - The actions that must be performed, and
  - The order it needs to perform those actions.
- This process may also involve reasoning about the information it requires to perform an action, the objects affected by an action, how long an action takes, how much it costs to perform an action, etc.

$\Rightarrow$ Humans are pretty good at many complex types of planning tasks. However, planning is a hard computational problem, especially in real-world robot environments.

# From dialogue and interaction to planning

- A key modality in human social interaction is natural language dialogue.
- Parallels to planning: a speaker tries to change the mental state of the hearer by applying actions that correspond to the utterance of words.
- The link between natural language and planning has a long tradition, e.g., Perrault and Allen (1980); Appelt (1985); Clark (1996); Stone (2000); Litman and Allen (1987); Cohen and Levesque (1990); Grosz and Sidner (1990), ...
- Recent work has tended to separate task planning from other types of natural language planning, which use more specialised approaches, e.g., finite state machines, information state (e.g., TrindiKit (Larsson and Traum 2000)), rule-based approaches to speech act theories, dialogue games, ...
- There has been a renewed interest in applying modern planning techniques to natural language problems, e.g., Koller and Stone (2007); Benotti (2008); Brenner and Kruijff-Korbayová (2008); Koller and Petrick (2011).

# High-level action selection in JAMES




- What action should the robot perform next?
- We use **automated planning** techniques from the symbolic artificial intelligence community, which are good at building goal-directed plans of action under many challenging conditions, given a formal description of a domain.
- **Goal:** build plans for serving all agents seeking attention in the bar (Loth et al. 2013; Foster et al. 2013).
- **Challenge**: replace the behaviour of a traditional interaction/dialogue manager with a general-purpose AI planner (Petrick and Foster 2013).

# Automated planning

- Automated **planning** techniques are good at building goal-directed plans of action under many challenging conditions, given a suitable description of a 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 and Nilsson 1971)

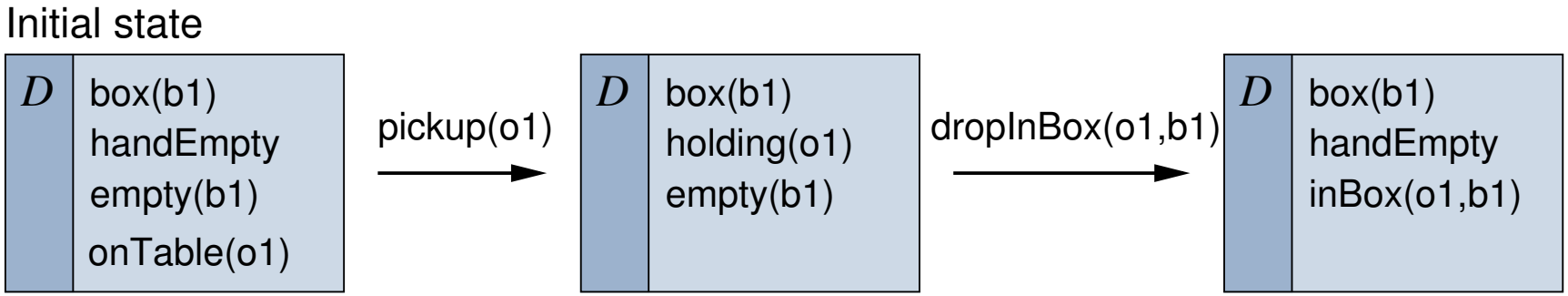
- A **world state** is represented by a **closed world** database $\mathcal{D}$ and negation as failure. This gives rise to a simple and efficient way of representing facts about the world:
  - $\phi$ is true if $\phi \in \mathcal{D}$,
  - $\neg\phi$ is true if $\phi \notin \mathcal{D}$, where $\phi$ is a ground atom.
- **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 against $\mathcal{D}$ **(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 and Hayes 1969).

# Example: STRIPS actions

| Action | Preconditions | Add list | Delete list |
|---|---|---|---|
| pickup$(x)$ | handEmpty<br>onTable$(x)$ | holding$(x)$ | handEmpty<br>onTable$(x)$ |
| dropInBox$(x, y)$ | holding$(x)$<br>box$(y)$ | inBox$(x, y)$<br>handEmpty | holding$(x)$<br>empty$(y)$ |

- Action operators: pickup, dropInBox
- Action parameters: $x$, $y$
- Properties: handEmpty, onTable, ...
- Objects: b1, o1, ...

# Example: planning with STRIPS actions




- Actions are state transforming: applying the effects of an instantiated action $A$ to a database $\mathcal{D}$ updates the database to produce a new database (denoting a new state) resulting from the execution of $A$.
- We can generate **plans** by chaining together fully instantiated actions.
- E.g., one plan that achieves a state where $\texttt{inBox}(\texttt{o1}, \texttt{b1})$ holds is the action sequence:

$$[\texttt{pickup}(\texttt{o1}),\ \texttt{dropInBox}(\texttt{o1}, \texttt{b1})].$$

# PKS: Planning with Knowledge and Sensing

- Our approach on JAMES: treat the problem as an instance of **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 build using forward search (+ some heuristics): 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

$$\phi \in K_w \text{ : the planner } \textit{knows whether } \phi$$

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

$$f \in K_v \text{ : the planner } \textit{knows the value} \text{ of } f$$

- $K_x$: 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)

# Reasoning about knowledge in PKS

- A primitive query language is used to ask simple 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.
- Reasoning is restricted by querying the databases, but often involves more than just a single database lookup.
- Used to evaluate preconditions, conditional rules, and goals.

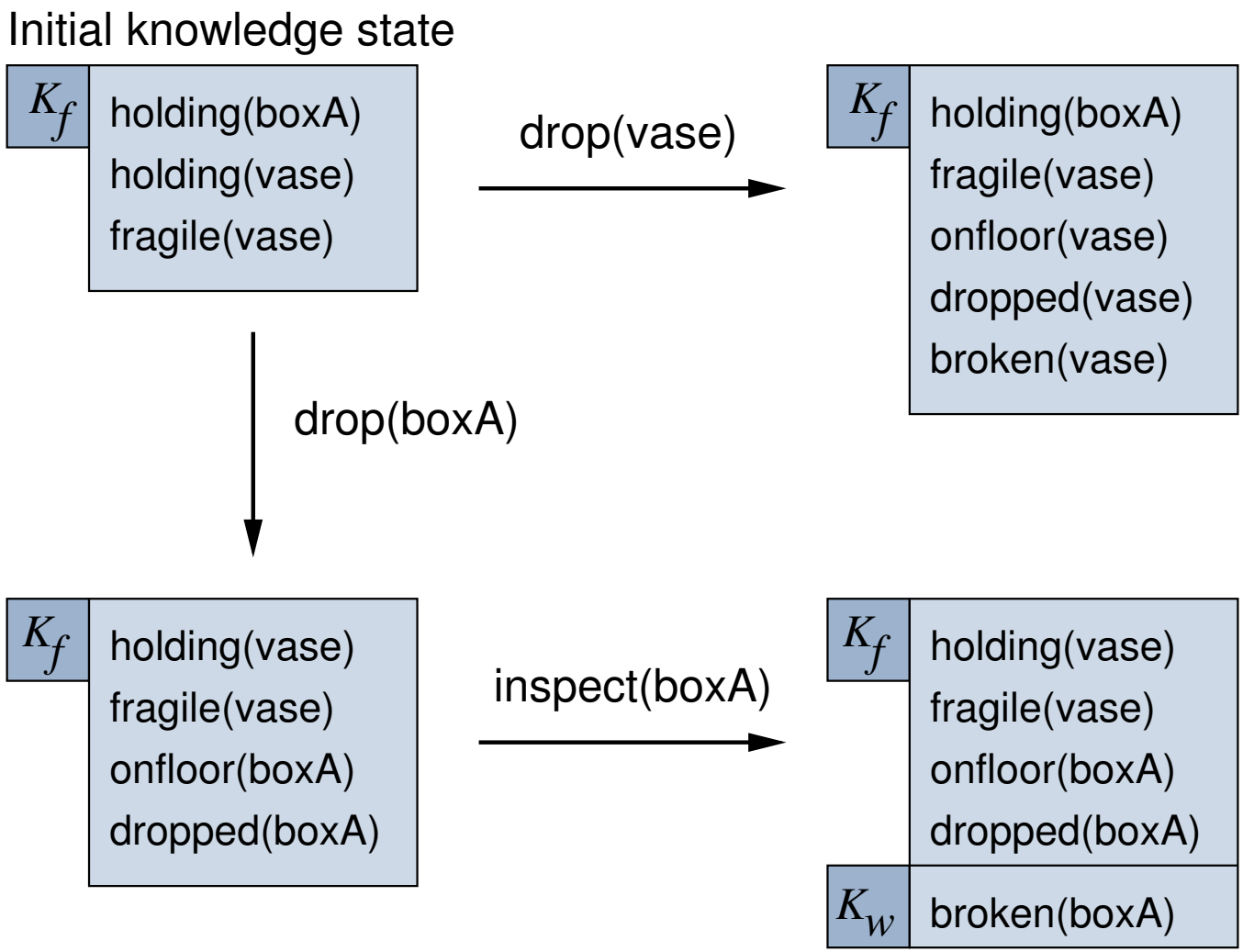
# Modelling actions in PKS

```
action drop(?x)
      preconds: K(holding(?x))
      effects:  del(Kf, holding(?x)),
                add(Kf, onFloor(?x)),
                add(Kf, dropped(?x)),
                del(Kf, !broken(?x)),
                K(fragile(?x)) => add(Kf, broken(?x))

action inspect(?y)
      preconds: true
      effects:  add(Kw, broken(?y))
```

- Actions capture the changes they make to PKS's knowledge state.
- New knowledge states are computed by forward chaining:
  - Evaluate preconditions against a set of databases **DB** (**KB**),
  - Effects update **DB** $\Rightarrow$ update **KB**.

# Example: drop and inspect



# Target interaction

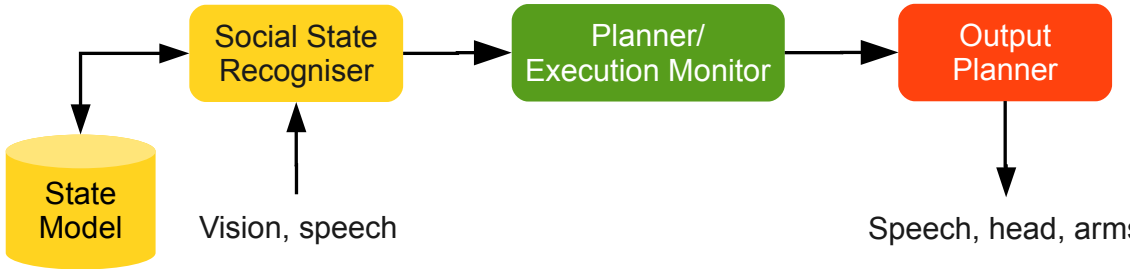
*Two people, A and B, each individually approach a bartender.*

| | | |
|---|---|---|
| Bartender (to A): | **How can I help you?** | Sensing action |
| Person A: | A pint of cider, please. | |

*Person C approaches the bartender and attracts his attention.*

| | | |
|---|---|---|
| Bartender (**nods at A**, then to C): | **Just a moment please**. | Social action |
| Bartender: | (**Serves A**) | Physical action |
| Bartender (to B): | **What will you have?** | Sensing action |
| Person B: | A glass of red wine. | |
| Bartender (**nods at B**): | | Social action |
| | (**Serves B**) | Physical action |
| Bartender (to C): | **Thanks for waiting.** | Social action |
| | **How can I help you?** | Sensing action |
| Person C: | I'd like a pint of bitter. | |
| Bartender (**nods at C**): | | Social action |
| | (**Serves C**) | Physical action |

# Planning in JAMES




- Input (fusion): sensor information from vision and speech.

  Output (fission): actions are postprocessed to generate arm motions, head behaviour, and speech.

- Domain includes:
  - Physical actions (e.g., handing over a drink),
  - Information-gathering (sensing) actions (e.g., asking a customer for a drink order) $\rightarrow$ often correspond to dialogue acts, and
  - Social behaviour (e.g., acknowledgements, thanking a customer).

- Goal: build plans to transact with all customers (agents) seeking attention at the bar (Petrick and Foster 2013; Foster et al. 2013).

# A social bartender domain

- Actions

| | |
|---|---|
| `greet(?a,?g)` | greet agent ?a in group ?g |
| `ask-drink(?a,?g)` | ask agent ?a in group ?g for a drink order |
| `ask-drink-next(?a,?g)` | ask the next agent ?a in group ?g for a drink order |
| `serve(?a,?d,?g)` | serve drink ?d to agent ?a in group ?g |
| `bye(?a,?g)` | end an interaction with agent ?a in group ?g |
| `wait(?a,?g)` | tell agent ?a in group ?g to wait |
| `ack-order(?a,?g)` | acknowledge the order of agent ?a in group ?g |
| `ack-wait(?a,?g)` | thank agent ?a in group ?g for waiting |
| `ack-thanks(?a,?g)` | acknowledge agent ?a's thanks |
| `inform-drinklist(?a,?t)` | inform agent ?a of the available drinks of type ?t |

- Properties

| | |
|---|---|
| `seeksAttn(?a)` | agent ?a seeks attention |
| `visible(?a)` | agent ?a is visible |
| `inGroup(?a) = ?g` | agent ?a is in group ?g |
| `inTrans = ?g` | the robot is interacting with group ?g |
| `request(?a) = ?d` | agent ?a has requested drink |
| … | |

⇒ Domain is inspired by data collected from studies in real bars (Huth 2011).

# Example actions

```
action ask-drink(?a : agent, ?g : group)
     preconds: K(inTrans = ?g) & K(inGroup(?a) = ?g) &
               K(!ordered(?a)) & & K(!otherAttnReq)
     effects:  add(Kf,ordered(?a)),
               add(Kv,request(?a))


action ack-order(?a : agent, ?g : group)
     preconds: K(inTrans = ?g) & K(inGroup(?a) = ?g) &
               K(ordered(?a)) & K(!ackOrder(?a)) &
               K(!otherAttnReq)
     effects:  add(Kf,ackOrder(?a))
```

# A plan for serving a single customer

```
greet(a1,g1),                    [Greet group g1]
ask-drink(a1,g1),                [Ask a1 for drink order]
ack-order(a1,g1),                [Acknowledge a1's order]
serve(a1,request(a1),g1),        [Give the drink to a1]
bye(a1,g1).                      [End the transaction]
```

- Simplest possible plan in the single customer case.
- Many aspects of the operating environment are dynamic and cannot be determined a priori: agents in the bar, agents seeking attention, initial utterances, etc. This information is provided by the **state manager**.
- Plans are built in response to customers seeking attention in the bar.
- Represent best-case scenarios based on current state information.

# A plan for two customers in two groups

| | |
|---|---|
| `wait(a2,g2),` | [Tell group g2 to wait] |
| `greet(a1,g1),` | [Greet group g1] |
| `ask-drink(a1,g1),` | [Ask a1 for drink order] |
| `ack-order(a1,g1),` | [Acknowledge a1's order] |
| `serve(a1,request(a1),g1),` | [Give the drink to a1] |
| `bye(a1,g1),` | [End g1's transaction] |
| `ack-wait(a2,g2),` | [Thank g2 for waiting] |
| `ask-drink(a2,g2),` | [Ask a2 for drink order] |
| `ack-order(a2,g2),` | [Acknowledge a2's order] |
| `serve(a2,request(a2),g2),` | [Give the drink to a2] |
| `bye(a2,g2).` | [End g2's transaction] |

- If a new customer arrives while the bartender is occupied, it nods at them and serves them later.

# A plan for two customers in one group

| | |
|---|---|
| `greet(a1,g1),` | [Greet group g1] |
| `ask-drink(a1,g1),` | [Ask a1 for drink order] |
| `ack-order(a1,g1),` | [Acknowledge a1's order] |
| `ask-drink-next(a2,g1),` | [Ask a2 for drink order] |
| `ack-order(a2,g1),` | [Acknowledge a2's order] |
| `serve(a1,request(a1),g1),` | [Give the drink to a1] |
| `serve(a2,request(a2),g1),` | [Give the drink to a2] |
| `bye(a2,g1).` | [End g1's transaction] |

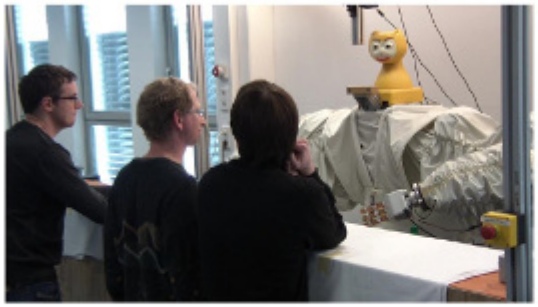
- When groups are detected, all individuals in a group are asked for their drink orders before any drinks to the group are served.
- Note: we can also support "round-buying" behaviour by making small changes to the planning domain.

# A single customer conditional plan

```
greet(a1),                          [Greet agent a1]
ask-drink(a1),                      [Ask a1 for drink order]
branch(request(a1))                 [Form branching plan]
   K(request(a1)=juice):            [If order is juice]
      …
      serve(a1,juice)               [Serve juice to a1]
   K(request(a1)=water):            [If order is water]
      …
      serve(a1,water)               [Serve water to a1]
   K(request(a1)=beer):             [If order is beer]
      …
      serve(a1,beer)                [Serve beer to a1]
bye(a1).                            [End the transaction]
```

- Branches let the planner consider order-specific actions/subdialogues.

# A more complex interaction



Three customers:
A1 and A2 in group G1
A3 is alone (singleton group G2)
Bartender serves members of G1 in sequence, then deals with G2.

Other social behaviour:
- First-come/first-served ordering
- All orders are acknowledged immediately
- If a new customer arrives while the bartender is occupied, it nods at them and serves them later

Social behaviour is based on the observation of bartenders in real bars (Huth et al., 2012); see Foster et al. (2013) for details on the planning domain.

| | |
|---|---|
| wait(A3, G1) | Tell G2 to wait (with a nod) |
| greet(A1, G1) | Greet group G1 |
| ask-drink(A1, G1) | Ask A1 for drink order |
| ack-order(A1, G1) | Acknowledge A1's order |
| ask-drink(A2, G1) | Ask A2 for drink order |
| ack-order(A2, G1) | Acknowledge A2's order |
| serve(A1, request(A1), G1) | Give the drink to A1 |
| serve(A2, request(A2), G2) | Give the drink to A2 |
| bye(A2, G1) | End G1's transaction |
| ack-wait(A3, G2) | Acknowledge G2's wait |
| ask-drink(A3, G2) | Ask A3 for drink order |
| ack-order(A3, G2) | Acknowledge A3's order |
| serve(A3, request(A3), G3) | Give the drink to A3 |
| bye(A3, G2) | End G2's transaction |

# Replanning when things go wrong

- Interactions are continually monitored to detect problems that may trigger replanning.
- Low-confidence speech recognition / timeouts

| | |
|---|---|
| … | |
| `ask-drink(a1)` | [Ask a1 for drink order] |
| **???** | [a1 was not understood] |
| **[Replan]** | [Replan] |
| `not-understand(a1)` | [Alert a1 not understood] |
| `ask-drink(a1)` | [Ask a1 again for drink order] |
| … | [Continue with old plan] |

- Overanswering

| | |
|---|---|
| `greet(a1)` | [Greet a1] |
| **???** | [a1 says “I’d like a beer”] |
| **[Replan]** | [Replan] |
| `serve(a1,request(a1))` | [Serve a1 their drink] |
| `bye(a1)` | [End the transaction with a1] |
| … | |

# JAMES interaction video


Image/video: fortiss GmbH

http://youtu.be/8k7Pd-CbbhE
http://james-project.eu/

# Experimental results

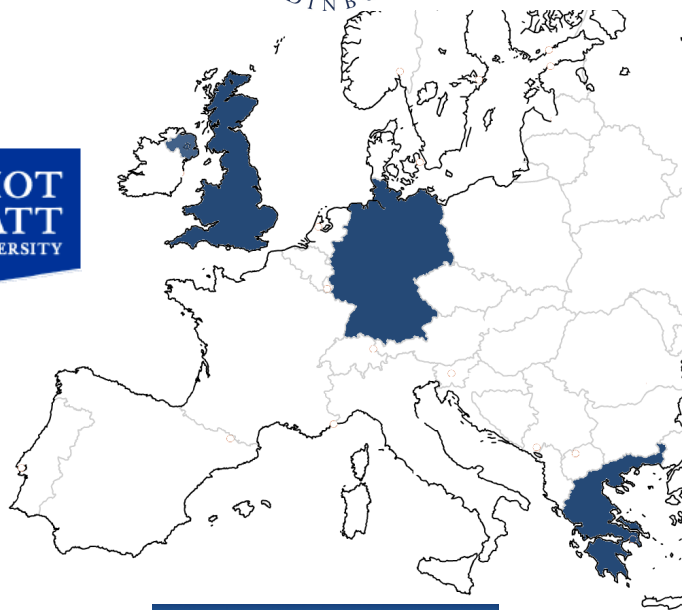
- Planning time is typically quite short, which doesn't negatively impact the system's reaction time (e.g., plans for 3 customers require 17 steps and $<$0.1s generation time).
  - Anything less than 2s is usually okay.
  - Robot motions are relatively slow which offers future opportunities for parallelising planning with other activities.
  - Frequent replanning in this domain.
- **Study 1**: system tested with 2 customers at a time in a drink ordering scenario (31 participants $\times$ 3 interactions each), 95% success rate on delivering correct drinks (Foster et al. 2012).
- **Study 2**: more complex scenario (3 customers at a time, 40 participants) involving group detection and a comparison of the full social domain vs. a task-only version of the domain, 87% success rate (Giuliani et al. 2013).
- Another study is planned for later this year.

# Conclusions

- Social interaction places additional requirements on the components of a cognitive robotics system: achieving a task goal isn't always enough.
- The same mechanisms used for general-purpose, symbolic planning can be applied to problems in dialogue planning, as an alternative to more mainstream approaches of natural language interaction management.
- Planning time in the bartender domain is typically quite short and doesn't negatively impact the system's reaction time. Replanning is frequent.
- The application area offers a potential testbed for exploring other types of planning problems: planning with preferences, planning under uncertainty, planning with constraints, …
- Ongoing/future work: planning with multiagent knowledge similar to (Steedman and Petrick 2007).

# JAMES

## Joint Action for Multimodal Embodied Social Systems · james-project.eu



Thanks to the many researchers who contributed to the design, implementation, and evaluation of the JAMES robot system, especially:

- Mary Ellen Foster (Heriot-Watt University)
- Andre Gaschler (fortiss GmbH)
- Manuel Giuliani (fortiss GmbH)
- Amy Isard (University of Edinburgh)
- Maria Pateraki (FORTH)
- Markos Sigalas (FORTH)
- Richard Tobin (University of Edinburgh)

This research received funding from the European Commission's Seventh Framework Programme (Grant No. 270435).

# References


Appelt, D. (1985). *Planning English Sentences*. Cambridge University Press, Cambridge.

Benotti, L. (2008). Accommodation through tacit sensing. In *Proceedings of the Workshop on the Semantics and Pragmatics of Dialogue (LONDIAL 2008)*, pages 75–82.

Brenner, M. and Kruijff-Korbayová, I. (2008). A continual multiagent planning approach to situated dialogue. In *Proceedings of the Workshop on the Semantics and Pragmatics of Dialogue (LONDIAL 2008)*, pages 67–74.

Clark, H. (1996). *Using Language*. Cambridge University Press, Cambridge.

Cohen, P. and Levesque, H. (1990). Rational interaction as the basis for communication. In *Intentions in Communication*, pages 221–255. MIT Press, Cambridge, MA.

Etzioni, O., Golden, K., and Weld, D. (1994). Tractable closed world reasoning with updates. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR 1994)*, pages 178–189.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Foster, M. E., Gaschler, A., and Giuliani, M. (2013). How can I help you? Comparing engagement classification strategies for a robot bartender. In *Proceedings of the International Conference on Multimodal Interaction (ICMI 2013)*, pages 255–261.

Foster, M. E., Gaschler, A., Giuliani, M., Isard, A., Pateraki, M., and Petrick, R. P. A. (2012). Two people walk into a bar: Dynamic multi-party social interaction with a robot agent. In *Proceedings of the International Conference on Multimodal Interaction (ICMI 2012)*, pages 3–10.

Giuliani, M., Petrick, R. P. A., Foster, M. E., Gaschler, A., Isard, A., Pateraki, M., and Sigalas, M. (2013). Comparing task-based and socially intelligent behaviour in a robot bartender. In *Proceedings of the International Conference on Multimodal Interaction (ICMI 2013)*, pages 263–270.

# References ... (2)


Grosz, B. and Sidner, C. (1990). Plans for discourse. In Cohen, P., Morgan, J., and Pollack, M., editors, *Intentions in Communication*, pages 417–444. MIT Press, Cambridge, MA.

Huth, K. (2011). Wie man ein bier bestellt. MA thesis, Fakultät für Linguistik und Literaturwissenschaft, Universität Bielefeld, Bielefeld, Germany.

Koller, A. and Petrick, R. P. A. (2011). Experiences with planning for natural language generation. *Computational Intelligence, Special Issue on Scheduling and Planning Applications: Selected Papers from the SPARK Workshop Series*, 27(1):23–40.

Koller, A. and Stone, M. (2007). Sentence generation as planning. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, pages 336–343.

Larsson, S. and Traum, D. (2000). Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3–4):323–340.

Litman, D. and Allen, J. (1987). A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200.

Loth, S., Huth, K., and De Ruiter, J. P. (2013). Automatic detection of service initiation signals used in bars. *Frontiers in Psychology*, 4(557).

McCarthy, J. and Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502.

Perrault, C. R. and Allen, J. F. (1980). A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, 6(3–4):167–182.

Petrick, R. P. A. and Bacchus, F. (2002). A knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, pages 212–221.

# References ... (3)

Petrick, R. P. A. and Bacchus, F. (2004). Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 2–11.

Petrick, R. P. A. and Foster, M. E. (2013). Planning for social interaction in a robot bartender domain. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2013), Special Track on Novel Applications*, pages 389–397.

Petrick, R. P. A., Foster, M. E., and Isard, A. (2012). Social state recognition and knowledge-level planning for human-robot interaction in a bartender domain. In *AAAI 2012 Workshop on Grounding Language for Physical Systems*, pages 32–38.

Steedman, M. and Petrick, R. P. A. (2007). Planning dialog actions. In *Proceedings of the SIGdial Workshop on Discourse and Dialogue (SIGdial 2007)*, pages 265–272.

Stone, M. (2000). Towards a computational account of knowledge, action and inference in instructions. *Journal of Language and Computation*, 1:231–246.

For more information about the **JAMES Project**, please visit **`http://james-project.eu/`**.