Epistemic Planning for Task-Based Action and Human-Robot Interaction

Ron Petrick
Edinburgh Centre for Robotics &
Department of Computer Science
Heriot-Watt University, Edinburgh, UK
R.Petrick@hw.ac.uk
http://petrick.uk/

Sapienza University, Rome, Italy
16 April 2019

How should an artificial agent sense, reason, and act to achieve its goals?

Humans perceive and manipulate their environments by sensing, reasoning, and acting. How should an artificial agent do this?

This talk: applications of **epistemic planning**.

Motivation

- An agent (human or artificial) trying to achieve its goals in the world must often reason about:
  - The actions (steps) that should be taken, and
  - The order they should be performed.
- This process is called **planning** and humans are pretty good at some types of planning tasks, e.g., taking a trip, making a meal, etc.
- **Automated planning** research attempts to understand this process computationally and build tools for solving this problem.
- **Epistemic planning** additionally involves reasoning about agent **knowledge** or **beliefs**.
- This is a hard computational problem, especially in real-world domains.

A task-based interaction

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

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

Person A: A lemonade, please.

Person C approaches and attracts the attention of the robot by gesturing.

Robot (to C): Just a moment please.

Robot: (Serves A)

Robot (to B): What will you have?

Person B: A glass of red wine.

Robot: (Serves B)

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

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

Robot: (Serves C)

What should the robot do next?

What should the robot do next?

Greet the customer
Ask the customer for a drink
Acknowledge the drink order
Pickup the correct bottle
Serve the customer
Close the transaction

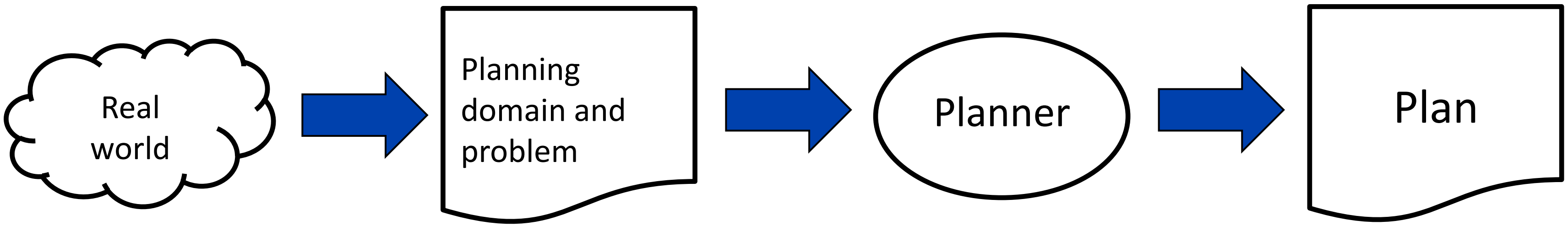
What should the robot do next?

Move to bar location 1
Pickup empty bottle1
Put bottle1 in bin
Move to bar location 2
Pickup bottle2
Put bottle2 in bin

Automated planning

- Automated **planning** techniques build goal-directed plans of action under challenging conditions, given a suitable domain description.
- A **planning problem** consists of:
  - 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 (tree) of actions that when applied to the initial state transforms the state to bring about the goal conditions.

Representation vs generation

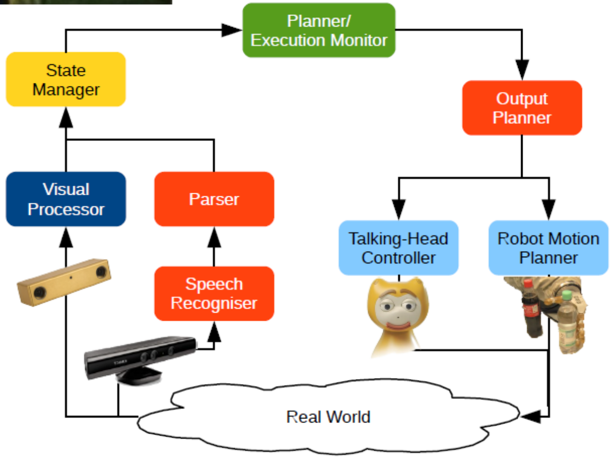


- A key idea in automated planning is that the **representation** of a planning problem is separated from the backend planning engine that **solves** the problem.

- **Domain independent planning**: planning engines support representation languages that can model a set of domains/problems, not just one domain/problem instance.

Automated planning in the real world

Plan



Classical planning

action pickup(?x)

preconds:

handEmpty

onTable(?x)

effects:

!handEmpty

!onTable(?x)

holding(?x)

action dropInBox(?x, ?y)

preconds:

holding (?x)

empty(?y)

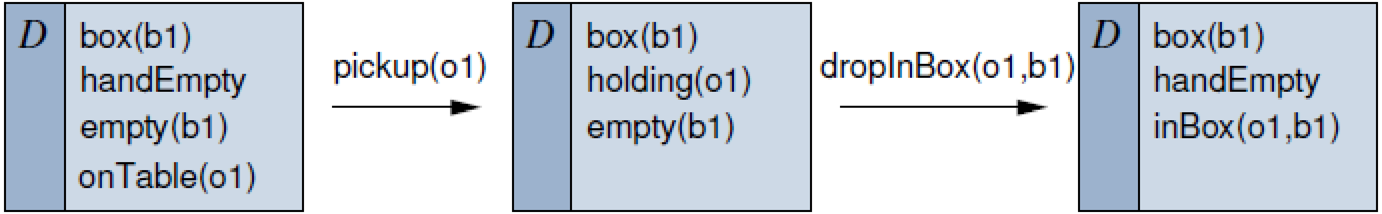
effects:

inbox(?x, ?y)

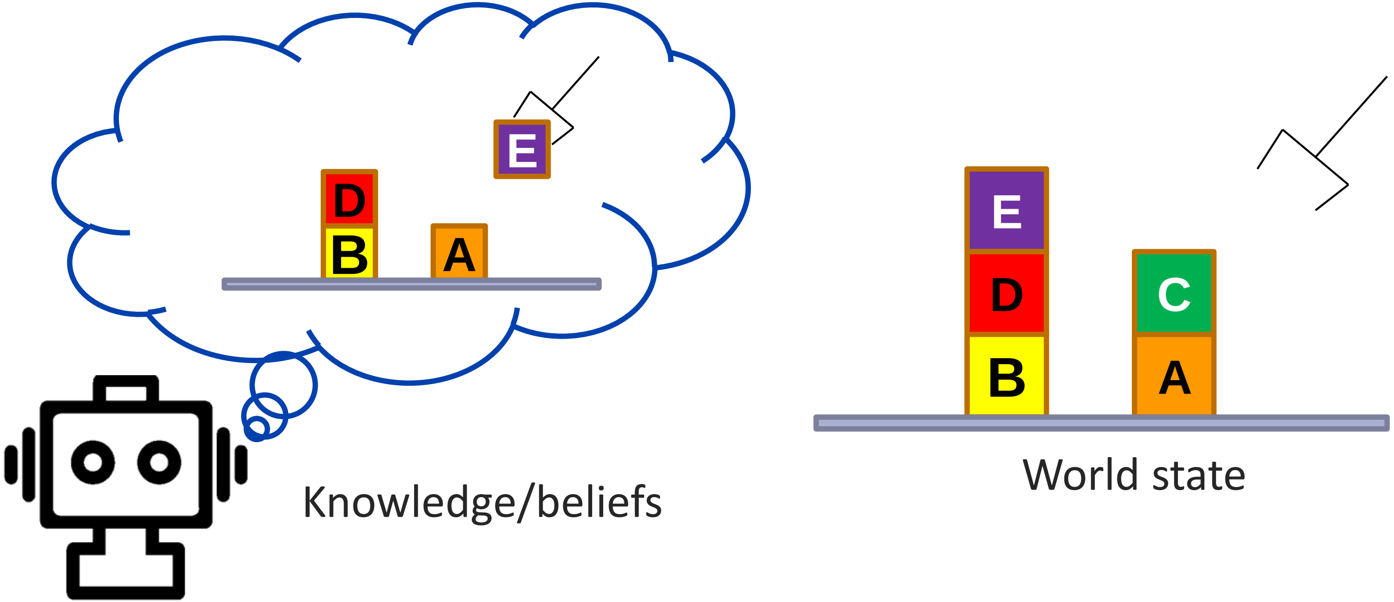
handempty

!holding

Planning = search through state-based transition system



Epistemic planning



- **Idea**: rather than planning in terms of world states, plan in terms of what the planner **knows** or **believes** about the world and other agents.

World level action vs knowledge level action

action pickup(?x)
    preconds:
        handEmpty
        onTable(?x)
    effects:
        !handEmpty
        !onTable(?x)
        holding(?x)

action pickup(?x)
    preconds:
        K(handEmpty)
        K(onTable(?x))
    effects:
        add(Kf,!handEmpty)
        add(Kf,!onTable(?x))
        add(Kf,holding(?x))

PKS: Planning with Knowledge and Sensing

- **PKS** is an epistemic planner that can reason with incomplete information and sensing actions.
- Based on a restricted knowledge representation (Kf, Kv, Kw, Kx, LCW).
- Actions are modelled as changing the planner's knowledge state, rather than the knowledge state.
- Knowledge state can be formally understood in terms of a modal logic of knowledge.

Representing knowledge in PKS

Kf: knowledge of positive and negative facts (but not closed world!)

$$p(c) \quad \neg q(b,c) \quad f(a)=c \quad g(b,c) \neq d$$

Kw: knowledge of binary sensing effects

$\phi \in$ Kw: the planner *knows whether* ϕ

Kv: knowledge of function values, multi-valued sensing effects

$f \in$ Kv : the planner *knows the value* of f

Kx: exclusive-or knowledge

$(p_1 \mid p_2 \mid \ldots \mid p_n) \in$ Kx: exactly one of the p_i must be true

LCW: local closed world information (Etzioni et al. 1994)

p(x) : the planner has LCW information of all instantiations of x

Reasoning in PKS

- A primitive query language is used to ask simple questions about the planner’s knowledge state:
  - K(ϕ), is ϕ known to be true?
  - Kv(f), is the value of f known?
  - Kw(ϕ), is ϕ 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.

Modelling actions in PKS

action grasp(?o : object)

preconds:

K(handEmpty)

K(onTable(?o))

effects:

add(Kf, !handEmpty)

add(Kf, !onTable(?o))

add(Kf, inHand(?o))

action senseWeight(?o : object)

preconds:

K(inHand(?o))

effects:

add(Kv, weight(?o))

action senseEmpty(?o : object)

preconds:

K(onTable(?o))

effects:

add(Kw,empty(?o))

Epistemic planning with PKS

Kf: handEmpty,onTable(bottle1),onTable(bottle2),!empty(bottle1)

Action: grasp(bottle1)

Kf: inHand(bottle1), onTable(bottle2), !handEmpty, !onTable(bottle1), !empty(bottle1)

Action: senseWeight(bottle1)

Kf: inHand(bottle1), onTable(bottle2), !handEmpty, !onTable(bottle1), !empty(bottle1)
Kv: weight(bottle1)

Action: senseEmpty(bottle2)

Kf: inHand(bottle1), onTable(bottle2), !handEmpty, !onTable(bottle1), !empty(bottle1)
Kw: empty(bottle2)
Kv: weight(bottle1)

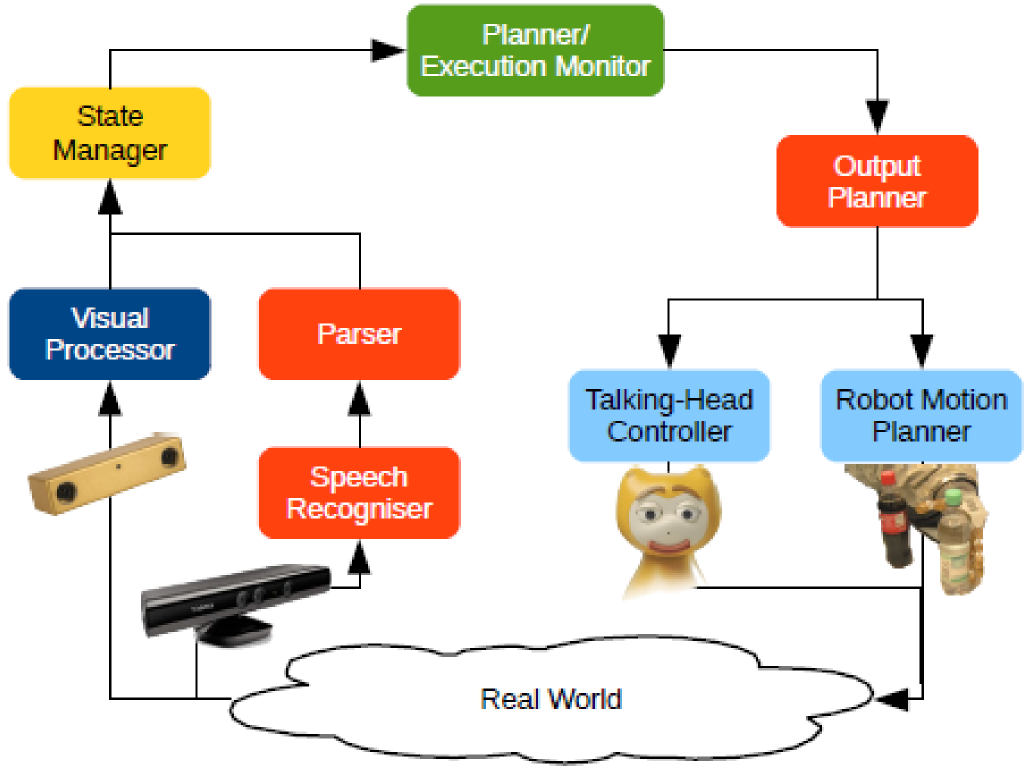
Epistemic planning = search through knowledge state transition system

Task-based interaction

JAMES
james-project.eu

Joint Action for Multimodal Embodied Social Systems (JAMES) – http://james-project.eu/

JAMES architecture



A task-based interaction

Two people, A and B, each individually approach a robot bartender.	
Robot (to A): How can I help you?	**Sensing action**
Person A: A lemonade, please.	
Person C approaches and attracts the attention of the robot by gesturing.	
Robot (to C): Just a moment please.	**Social action**
Robot: (Serves A)	**Physical action**
Robot (to B): What will you have?	**Sensing action**
Person B: A glass of red wine.	
Robot: (Serves B)	**Physical action**
Robot (to C): Thanks for waiting. How can I help?	**Social action** **Sensing action**
Person C: I'd like a pint of IPA.	
Robot: (Serves C)	**Physical action**

Planning domain actions

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

Planning domain properties

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

The JAMES domain is described symbolically for the planner, inspired by data collected from human studies in real bars (Huth 2011)

Epistemic planning actions

```
action greet(?a : agent)
    preconds:
        K(inTrans = nil) &
        K(!ordered(?a))
    effects:
        add(Kf, inTrans = ?a)
```

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

```
action serve-drink(?a : agent, ?d)
    preconds:
        K(inTrans = ?a) &
        K(ordered(?a)) &
        Kv(request(?a)) &
        K(request(?a) = ?d)
    effects:
        add(Kf, served(?a))
```

```
action bye(?a : agent)
    preconds:
        K(inTrans = ?a) &
        K(served(?a))
    effects:
        add(Kf, inTrans = nil)
```

Plan for 1 customer

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

- Simplest possible plan in the single customer case.
- Plans built in response to customers seeking attention.
- Represent best-case scenarios based on current beliefs.

Plan for 2 individual customers

wait(a2)	Tell a2 to wait
greet(a1)	Greet a1
ask-drink(a1)	Ask a1 for drink order
ack-order(a1)	Acknowledge a1’s order
serve(a1,request(a1))	Give the drink to a1
bye(a1)	End a1’s transaction
ack-wait(a2)	Thank a2 for waiting
ask-drink(a2)	Ask a2 for drink order
ack-order(a2)	Acknowledge a2’s order
serve(a2,request(a2))	Give the drink to a2
bye(a2).	End a2’s transaction

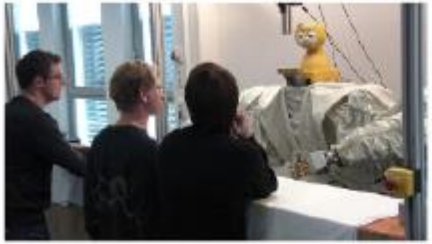
Plan for 2 customers in one group

greet(a1)	Greet a1
ask-drink(a1)	Ask a1 for drink order
ack-order(a1)	Acknowledge a1’s order
ask-drink-next(a2)	Ask a2 for drink order
ack-order(a2)	Acknowledge a2’s order
serve(a1,request(a1))	Give the drink to a1
serve(a2,request(a2))	Give the drink to a2
bye(a2).	End a1’s transaction

Single customer contingent 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

Plan for 3 customers



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

Plan execution: 1 customer

https://drive.google.com/open?id=11UYktwEd0wHEYlmopkUmwSmceMOL5IrA

Plan execution: 2 customers, 1 order

https://drive.google.com/open?id=1aUK1c07mJAfOSqSDJ1YXVL_1wr2m5qhJ

Plan execution: 2 customers, 2 orders

https://drive.google.com/open?id=1KMRQOIUgJF_0RI1r9NzsFaz__Kn3A1Uk

Combined task and motion planning

action pickup(?r : robot, ?o : obj, ?l : loc)

- **preconds**:
  - K(objectLocation(?o) = ?l)
  - K(handEmpty(?r))
  - K(extern(isReachable(?l,?r)))
- **effects**:
  - del(Kf, getObjectLocation(?o) = ?l)
  - del(Kf, handEmpty(?r))
  - add(Kf,inHand(?o,?r))

```
senseIfEmpty(b1)
senseIfEmpty(b2)
branch(isEmptyBottle(b1))
K+: branch(isEmptyBottle(b2))
      K+: pickup(left,b1,l1)
          putdown(left,b1,l5)
          pickup(right,b2,l2)
          putdown(right,b2,bin)
          pickup(right,b1,l5)
          putdown(right,b1,bin)
      K-:  …
K-: branch(isEmptyBottle(b2))
     K+: …
     K-: …
```

Plan execution: bottle removal

https://youtu.be/yMmZkhHr8ss

Different embodiment

https://drive.google.com/open?id=1JU_lMlfI3PHI8i4jqMWcSDz1Grgk4wHI

Execution monitoring and 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
...	

Overanswering

greet(a1)	Greet a1
???	a1 says "I'd like a beer"
[Replan]	[Replan]
serve(a1,request(a1))	Serve a1 their drink

Experiments and results

- System tested with 2 customers at a time in a drink ordering scenario (31 participants 3 interactions each): 95% success rate on delivering correct drinks. Details in (Foster et al. 2012).
- 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.
- We also performed a second set of experiments to compare the social bartender against a non-social version (Giuliani et al. 2013).

Conclusions

- General-purpose epistemic planning has been successfully applied to problems in task-based action and human-robot interaction.
- Current applications:
  - Explainable planning
  - Connecting task and motion planning
  - Action learning
  - Multiagent mission planning

References

- M.E. Foster and R. Petrick (2017). Separating Representation, Reasoning, and Implementation for Interaction Management: Lessons from Automated Planning, in K. Jokinen and G. Wilcock (eds.), Dialgogues with Social Robots: Enablements, Analyses, and Evaluation, Lecture Notes in Electrical Engineering, 427:93-107, Springer.
- M.E. Foster, A. Gaschler, M. Giuliani, A. Isard, M. Pateraki, and R. Petrick (2012). Two People Walk Into a Bar: Dynamic Multi-Party Social Interaction with a Robot Agent, Proceedings of the ACM International Conference on Multimodal Interaction (ICMI), pages 3-10.
- A. Gaschler, R. Petrick, O. Khatib, and A. Knoll (2018). KABouM: Knowledge-Level Action and Bounding Geometry Motion Planner, Journal of Artificial Intelligence Research, 61:323-362, doi:10.1613/jair.5560.
- M. Giuliani, R. Petrick, M.E. Foster, A. Gaschler, A. Isard, M. Pateraki, and M. Sigalas (2013). Comparing Task-Based and Socially Intelligent Behaviour in a Robot Bartender, Proceedings of the ACM International Conference on Multimodal Interaction (ICMI), pages 263-270.
- K. Huth (2011). Wie man ein bier bestellt, MA thesis, Fakultaet fuer Linguistik und Literaturwissenschaft, Universitaet Bielefeld, Bielefeld, Germany.
- R. Petrick and M.E. Foster (2016). Using General-Purpose Planning for Action Selection in Human-Robot Interaction, AAAI 2016 Fall Symposium on Artificial Intelligence for Human-Robot Interaction (AI-HRI).

References (2)

- R. Petrick and M.E. Foster (2016). Action Selection for Interaction Management: Opportunities and Lessons for Automated Planning, Workshop of the UK Planning and Scheduling Special Interest Group.
- R. Petrick and M.E. Foster (2013). Planning for Social Interaction in a Robot Bartender Domain, Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), pages 389-397.
- R. Petrick and F. Bacchus (2004). Extending the Knowledge-Based Approach to Planning with Incomplete Information and Sensing, Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), pages 2-11.
- R. Petrick and F. Bacchus (2002). A Knowledge-Based Approach to Planning with Incomplete Information and Sensing, Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS), pages 212-221.

- Additional information about the JAMES project can be found at:

http://james-project.eu/