Using Internal Agent Fluents to Represent Knowledge, Beliefs and Intentions

Tracey Lall
Department of Computer Science
Rutgers, The State University of New Jersey

Abstract

This position paper advocates a simple fluent representation of intention, knowledge, belief and memory that due to its grounded and integrated nature allows the use of constraint based planning algorithms for all aspects of agent reasoning and execution, including planning under uncertainty and integrated diagnosis and repair.

Introduction

The representation described in this paper was developed for the implementation of an agent whose role is to schedule jobs in a computer batch job environment, monitor progress and diagnose and correct any problems (by running scripts or operating system commands). The requirements for this application include:

- Deliberative planning under partial observability
- Planning with time based goals
- Execution monitoring
- Planning for sensing (including formulation of knowledge goals)
- Diagnosis and planning for repair.
- Memory of past events (required for diagnosis)

Existing representations

Existing approaches include the belief-desire-intention agent implementations such as PRS (M. Georgeoff 1998), JAM (Huber 1999) and deliberative planning languages, such as NuPDDL (Bertoli & et al. 2003) , SADL, (Golden & Weld 1996) and Golog (Levesque *et al.* 1997). Many of the representational requirements are covered by these approaches, however the facilities they do not offer are:

- Ability to fully represent incomplete plans and future replanning actions.
- Flexible belief formation (though SADL offers a retrospectively checked knowledge precondition mechanism).
- Ability to represent past events.
- Integration of diagnosis and repair.

Fundamentals of the representation

The representation uses a concept of state which is a ground assignment of all fluents (including time which is represented as a fluent). The evolution of the world is described via a set of constraints which define the dynamics of the world - how the state at one point in time is related to the state at other times. To enable a dynamical description of the agent itself, the concept of a *plan execution machine* is used which allows the plan dynamics to be defined via the use of fluents which represent the agent actions and beliefs involved in that plan. Actions in the external world are described using constraints which define the evolution of the external world when actions are applied (essentially viewing the agent + external world as a combined system which obeys a set of dynamical constraints)

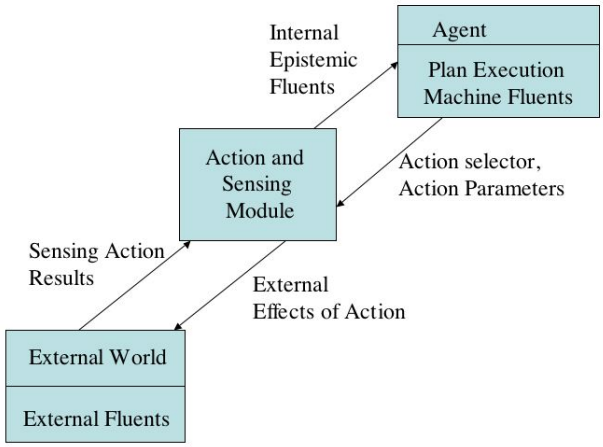

Figure 1: Agent's interaction with external world

Action representation Actions are described by constraints that define how the state of the combined system evolves when a particular action is selected. As an example of how this can be done, the action of starting a batch job is described using constraints involving time, the action selector, action parameters and the external fluents. (The action

selector being the internal agent fluent whose value controls which particular action is applied at that time - effectively corresponding to the $happens$ predicate in the event calculus).
$(time = t \implies actionSelector = start) \implies$
$(time = t + 1 \implies state = Starting)$
(Where t is a free variable)
The external world dynamics of a batch job which runs for between 20 and 50 seconds and then completes may be represented with the following constraints:
$(time = t \implies state = Starting) \implies$
$(time = t + 1 \implies (state = Running \wedge$
$20 \leq timeToComplete \leq 50))$
$(time = t \implies (state = Running \wedge$
$timeToComplete = r)) \implies$
$(time = t + 1 \implies timeToComplete = r - 1)$
$(time = t \implies state = Running \wedge$
$timeToComplete = 0) \implies$
$(time = t + 1 \implies state = Completed)$
The law of inertia is handled through standard explanation closure axioms for each fluent and the agent has a special action to represent the act of doing nothing.

Plan execution machine The plan execution machine representation is based on the standard partial order planner graph and consists of a set of action nodes (represented as a tuplet of fluents, including the parameters of the action), and epistemic fluents (or *belief fluents*) which are internal plan fluents which represent the value of propositions or fluents in the external world. When a node is activated, the action selector is set to the action associated with that node and is converted by the action module into the corresponding real-world action. Sensing actions update the belief fluents associated with the sensing action node. A node is activated when it's predecessor nodes have been executed and when the belief fluents upon which the node depends are true. (These belief fluents may represent either the preconditions of the node's action or specific test conditions upon which the action is contingent). This plan execution machine (which supports conditional and cyclic execution) has clear dynamics defined in the same manner as the external world using dynamical constraints. In essence the problem of planning becomes the problem of searching for an execution machine such that the combined evolution of the execution machine + external world gives rise to the goal state. In effect the dynamics of the plan execution machine mimics the causal dependencies and dynamics of the actions in the external world and in a loose sense therefore the agent uses an internal model of the external world for its planning. Knowledge of an external world fluent value is defined as the condition when the internal belief fluent has the same value as the fluent or proposition which it represents in the outside world . This correspondence between internal belief fluents with external fluents is achieved via sensing actions.

Advantages of this representation

Having such a representation allows standard partial order plan algorithms to be used, but the representation of the plan using a state based execution machine allows higher flexibility since the fungibility of the belief and action node fluents allows the use of the results of other actions (for example, sensing actions) to control the plan execution. The grounded plan representation gives a clear and simple semantics for conditional and iterative plans and gives the agent's planner the ability to reason about its own behaviour.

This fungibility also makes it possible to define a replanning action whose effects includes updates of unexecuted node and belief fluents. Since the effects of the replanning action are open ended (the agent doesnt know apriori what the effects of its replanning will be), the agent would model this as a non deterministic action. Such re-planning actions can be represented in the plan as action nodes which are only executed under non nominal conditions.

Giving the belief fluents an independent existence from their logical definitions allows for the planner to use flexible belief formation and defining knowledge of a value as equivalence between internal belief fluent and the external fluent allows a state based search planner to construct a plan which can achieve this equivalence using a combination of sensing actions which individually may only give incomplete information about the fluent (as in for example the safe combination problem (Petrick & Bacchus 2002)

Since the representation is a direct description of the agents inner state, it is also capable of representing both past and future states. Nodes which have already been executed represent actions which have already taken place and the belief fluents related to such nodes represent the value of the fluent, *at the time that the node was executed*.

Practical evaluation

Implementation has commenced on an agent based on this representation using a constraint based search planner. The agent will be evaluated on scenarios gathered from the target domain. It is hoped that due to the concrete and flexible nature of the representation, it will be capable of supporting the majority of the typical failure and repair scenarios occurring in batch job environments.

References

Bertoli, P., and et al. 2003. Extending pddl to nondeterminism, limited sensing and iterative conditional plans. In *ICAPS Workshop on PDDL, Informal Proceedings*, 15–24.

Golden, K., and Weld, D. 1996. Representing sensing actions: The middle ground revisited. In *KR'96: Principles of Knowledge Representation and Reasoning*. 174–185.

Huber, M. J. 1999. JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of The Third International Conference on Autonomous Agents*, 236–243.

Levesque, H. J.; Reiter, R.; Lesperance, Y.; Lin, F.; and Scherl, R. B. 1997. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming* 31(1-3):59–83.

M. Georgeoff, F. I. 1998. Decision-making in an embedded reasoning system.

Petrick, R., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of AIPS'02*, 212–221.