Planning, Executing, and Monitoring Communication in a Logic-based Multi-agent System

Martin Magnusson and **David Landén** and **Patrick Doherty**[1]

1 Introduction

Imagine the chaotic aftermath of a natural disaster. Teams of rescue workers search the affected area for people in need of help, but they are hopelessly understaffed and time is short. Fortunately, they are aided by a small fleet of autonomous unmanned aerial vehicles (UAVs). The UAVs help in quickly locating injured by scanning large parts of the area from above using infrared cameras and communicating the information to the command and control center (CCC) in charge of the emergency relief operation.

An autonomous agent carrying out tasks in such dynamic environments must automatically construct plans of action adapted to the current situation and the other agents. Its multi-agent plans involve both physical actions, that affect the world, and communicative actions, that affect the other agents' mental states. In addition, assumptions made during planning must be monitored during execution so that the agent can autonomously recover, should its plans fail.

The strong interdependency between these capabilities can be captured in a formal logic. We take advantage of this by building a multi-agent system that reasons directly with the logical specification using automated theorem proving. Our implementation and its integration with a physical robot platform, in the form of an autonomous helicopter, goes some way towards demonstrating that this idea is not only theoretically interesting, but practically feasible.

2 Speech Acts in TAL

The system is based on automated reasoning in Temporal Action Logic (TAL) [1], a first-order logic for commonsense knowledge about action and change. Inspired by Morgenstern's work [3] we extend TAL with *syntactic* operators for representing agents' mental states and beliefs. A formula preceded by a quote is a regular first-order term that serves as a *name* of that formula. Alternatively one may use a backquote, which facilitates *quantifying-in* by exposing variables inside the backquoted expression to binding by quantifiers.

With quotation one may pass (names of) formulas as arguments to regular first-order predicates, without introducing modal operators. E.g., the fact that the UAV believes, at noon, that there were, at 11:45, five survivors in cell 2,3 in a coordinate grid of the disaster area can be expressed by:

(Believes uav 12:00 '(= (value 11:45 (survivors (cell 2 3))) 5))

This epistemic extension of TAL enables us to characterize communication in terms of actions that affect the mental states of others. Such *speech acts* form a basis for planning both physical and communicative actions in the same framework, as is done e.g. by Perrault, Allen, and Cohen [4]. Speech acts have also been adopted by research on agent communication languages (ACL) such as the widely used FIPA ACL[2], which establish standards that ensure interoperability between different multi-agent systems. With the help of quotation in TAL we formulate the FIPA *inform*, *informRef*, and *request* speech acts. These can be used by agents to communicate beliefs to, and to incur commitment in, other agents.

3 Planning

Planning with speech acts is, in our framework, the result of proving a goal while abductively assuming action occurrences that satisfy three kinds of preconditions. The action must be physically *executable* by an agent during some time interval (*b e*], the agent must have a belief that *identifies* the action, and the agent must be *committed* to the action occurring, at the start of the time interval:

(→ (∧ (Executable *agent* (*b e*] *action*)
(Believes *agent b* '(ActionId '*action* '*actionid*))
(Committed *agent b* '(Occurs *agent* (*b e*] *action*)))
(Occurs *agent* (*b e*] *action*))

Executability preconditions are different for each action and are therefore part of the specification of an action.

The belief preconditions are satisfied when the agent knows identifiers for the arguments of a primitive action [2]. But the time point at which an action is executed is also critically important. However, it seems overly restrictive to require that the agent holds beliefs that *identify* the action occurrence time points. Actions that do not depend on external circumstances can be executed whenever the agent so chooses, without deciding upon an identifiable clock time in advance. Actions that do depend on external circumstances can also be successfully executed as long as the agent is sure to know the correct time point when it comes to pass.

This is precisely what the concept of *dynamic controllability* captures. Following Vidal and Fargier [6] we denote time points controlled by the agent by *b* and time points over which the agent has no control by *e*. The temporal dependencies between actions form a simple temporal network with uncertainty (STNU) that can be checked for dynamic controllability to ensure an executable plan.

Finally, the commitment precondition can be satisfied in one of two ways. Either the agent adds the action to its own planned execu-

[1] Linköping University, Sweden, email: {marma,davla,patdo}@ida.liu.se
This work is supported in part by a grant from the Swedish Research Council (VR), the National Aeronautics Research Program NFFP04 S4203, CENIIT, and the Strategic Research Center MOVIII, funded by the Swedish Foundation for Strategic Research, SSF.

[2] http://www.fipa.org/specs/fipa00037/

tion schedule (described below), or it uses the request speech act to delegate the action to another agent, thereby ensuring commitment.

4 Execution

Scheduled actions are tied to the STNU through the explicit time points in the Occurs predicate. An STNU *execution* algorithm propagates time windows during which these time points need to occur. Executed time points are bound to the current clock time and action occurrences scheduled at those time points are proved *dispatched* using the following axiom:

(→ (∧ (ActionId ‘*action* ‘*id*)
(ProcedureCall *agent* (*b e*] *id*))
(Dispatch *agent* (*b e*] *action*))

The axiom forces the theorem prover to find an action identifier with standardized arguments for ProcedureCall predicate. This is the link between the automated reasoner and the execution sub-system in that the predicate is proved by looking up the procedure associated with the given action and calling it. But the actions are often still too high-level to be passed directly to the low-level system. An example is the action of scanning a cell of the coordinate grid with the infrared camera. This involves using a scan pattern generator, flying the generated trajectory, and applying the image processing service to identify humans in the video footage. The assumption is that the scanning of a grid cell will always proceed in the manner just described, so there is no need to plan its sub-actions.

Such macro-actions, and primitive physical actions, are realized (in simulation, so far) by an execution framework, built using the Java agent development framework[3] (JADE). It encapsulates the agent so that all communication is channeled through a standardized interface as FIPA ACL speech acts.

5 Monitoring

Executing the plan will satisfy the goal as long as abduced assumptions hold up. But the real world is an unpredictable place and unexpected events are sure to conspire to interfere with any non-trivial plan. To detect problems early we continually evaluate the assumptions that are possible to monitor.

E.g., the agent’s plan might rely on some aspect of the environment to persist, in effect making a frame assumption. A failure of such an assumption produces a percept that is added to the agent’s knowledge base. A simple truth maintenance system removes assumptions that are contradicted by observations and unchecks goals that were previously checked off as completed but that depended on the failed assumptions. This immediately gives rise to plan revision and failure recovery as the theorem prover tries to reestablish those goals.

If the proof of the unchecked goals succeeds, the revision will have had minimal effect on the original plan. A failed proof means that the current sub-goal is no longer viable, in the context of the execution failure, and the revision is extended by dropping the sub-goals one at a time. This process continues until a revision has been found or the main goal is dropped and the mission fails.

6 Multi-agent Scenario

The theory presented so far needs to be complemented with an automated reasoner. The current work utilizes a theorem prover named ANDI, which is based on Pollock’s *natural deduction* system that makes use of unification [5].

Natural deduction is an interesting alternative to the widely used resolution method. The set of proof rules is extensible and easily accommodates special purpose rules that make reasoning more efficient. ANDI incorporates specialized inference rules for reasoning with quoted expressions and beliefs according to the rules and axioms of TAL. This enables ANDI to process the follwing scenario in less than two seconds on a laptop with a 2.4GHz Intel Core 2 Duo Mobile T7700 processor.

Suppose that the CCC wants to know the number of survivors in grid cell 2,3 at 13:00. The UAV produces the following plan (in addition to an STNU that orders the actions in time):

(fly (cell 2 3))
(scan (cell 2 3))
(informRef ccc ’(value 13:00 (survivors (cell 2 3))))

The success of the plan depends on two persistence assumptions that were made during planning and that are monitored during execution, namely that (location uav) is not affected between flying and scanning, and that (radio uav ccc) indicates a functioning radio communication link. There is also an assumption of the persistence of the survivor count, though this is impossible for our UAV to monitor since it can not see the relevant area all at once. If one of the survivors leaves, then the plan revision process will take the resulting body count discrepancy into account when it is discovered.

Suppose however that due to the large distance and hostile geography of the area (or some other unknown error) the radio communication stops functioning while the UAV is scanning the area, before reporting the results. The UAV perceives that the fluent (radio uav ccc) was *not* persistent and the truth maintenance system successively removes incompatible assumptions and sub-goals until a revised plan is found:

(informRef mob ’(value 13:00 (survivors (cell 2 3))))
(request mob ’(Occurs mob (*b e*]
(informRef ccc ’(value 13:00 (survivors (cell 2 3))))))

The new plan involves requesting help from another mobile agent (mob). By communicating the survivor count to this “middle man”, and requesting it to pass on the information to the CCC, the UAV ensures that the CCC gets the requested information.

Another set of assumptions now require monitoring, namely (radio mob ccc) and (radio uav mob). While the UAV cannot monitor the other agent’s radio communication, it will be monitored if that agent is also running our agent architecture. At this point let us assume that no further failures ensue so that the knowledge gathering assignment is completed successfully within this paper’s page limit.

[3] http://jade.tilab.com/

REFERENCES

[1] Patrick Doherty and Jonas Kvarnström, ‘Temporal action logics’, in *Handbook of Knowledge Representation*, eds., Vladimir Lifschitz, Frank van Harmelen, and Bruce Porter, Elsevier, (2007).

[2] Robert Moore, ‘Reasoning about knowledge and action’, Technical Report 191, AI Center, SRI International, Menlo Park, CA, (1980).

[3] Leora Morgenstern, *Foundations of a logic of knowledge, action, and communication*, Ph.D. dissertation, New York, NY, USA, 1988.

[4] Raymond C. Perrault, James F. Allen, and Philip R. Cohen, ‘Speech acts as a basis for understanding dialogue coherence’, in *TINLAP’78*, pp. 125–132, (1978).

[5] John L. Pollock, ‘Natural deduction’, Technical report, Department of Philosophy, University of Arizona, (1999).

[6] Thierry Vidal and Hélène Fargier, ‘Handling contingency in temporal constraint networks: From consistency to controllabilities’, *JETAI*, **11**(1), 23–45, (1999).