Logical Agents that Plan, Execute, and Monitor Communication*

Martin Magnusson and **David Landén** and **Patrick Doherty**
Department of Computer and Information Science, Linköping University, Sweden
{marma,davla,patdo}@ida.liu.se

Abstract

Real and simulated worlds that involve cooperation between several autonomous agents create highly dynamic conditions. An individual agent can not hope to rely on fixed behaviors, but needs to plan both physical actions and communication actions that enlist other agents to help achieve the goal. We present a logical agent that plans regular actions and speech acts, monitors their execution, and recovers from unexpected failures. The architecture is based on automated reasoning in a formal logic and is grounded in a physical robot platform in the form of an autonomous helicopter. We apply the system to a knowledge gathering goal and illustrate how it deals with the complexities of the task.

Introduction

Imagine the chaotic aftermath of a natural disaster. Teams of rescue workers search the afflicted 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 unmanned aerial vehicles (UAVs) that can be requested to carry out tasks autonomously. The UAVs help quickly locate 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.

Complex tasks like these occur frequently in real world situations as well as in training simulations and computer games. Achieving their goals requires the cooperation of many independent agents. Each agent faces an environment that can change in response to actions by other agents in addition to its own. It is unrealistic to assume that such an agent could be made autonomous by equipping it with fixed behaviours for all possible situations that might arise. Instead, the agent must automatically construct plans of action adapted to the situation at hand. These multi-agent plans involve other agents' mental states and communicative acts to affect them. In addition, assumptions made during planning must be monitored during execution so that the agent can autonomously recover from failures, which are prone to happen in any dynamic environment. These are significant challenges for any proposed planning formalism.

*This work is supported in part by the Swedish Foundation for Strategic Research (SSF) Strategic Research Center MOVIII, the Swedish Research Council Linnaeus Center CADICS, and CENIIT, the Center for Industrial Information Technology.

We extend Temporal Action Logic (Doherty and Kvarnström 2007), a first-order language with a well-developed methodology for representing actions that has previously been used for planning (Magnusson 2007), with syntactic operators that express the modalities of belief and commitment. This makes it possible to formalize *inform* and *request* speech acts, and to plan both physical and communicative actions in a multi-agent setting. Automation is provided by a natural deduction theorem prover that integrates planning, execution, monitoring, and plan revision. The resulting physical actions are executed (in simulation) by a delegation framework (Doherty and Meyer 2007), built using the Java agent development framework[1] (JADE), and communicative actions are realized as standardized FIPA ACL (Foundation for Intelligent Physical Agents 2002) speech acts. We describe this agent architecture and its use in solving the above scenario.

Related Work

Early planning research tended to ignore any processes that occur before or after the planning phase, and many classical planning algorithms depended crucially on a global closed world assumption. Agent programming languages provide interesting alternatives that often incorporate execution and monitoring, but move planning to a less prominent role or eliminate planning altogether, which might result in some sacrifice of flexibility in unforeseen situations.

Modern planning research tries to relax the classical restrictions while keeping the planning process in focus. Shanahan's agent architecture (Shanahan 2000) based on the Event Calculus seamlessly integrates planning and plan revision. Though it has not been applied in multi-agent settings, and its implementation as an abductive logic program is not sufficiently expressive to represent communicative acts conveniently.

Searle's *speech acts* (1969) characterize natural language utterances as actions with conditions upon their execution and effects on the mental states of others. Speech acts form a basis for expressing communicative acts as planning operators, as is done by Perrault, Allen, and Cohen (1978). They generate plans that involve both regular actions and speech acts, but the implementation uses limited versions of speech acts in the form of STRIPS-like planning operators.

Speech acts have also been adopted by research on software agents (Genesereth and Ketchpel 1994). This body of

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

work depends fundamentally on agent communication languages (ACL), which are standardized sets of speech acts that ensure interoperability in agent to agent communication. The widely used FIPA ACL is based on speech act theory and has a logical semantics defined using multi-modal BDI logic. But there is no integration of speech acts within a more general framework of action and change that would facilitate planning of speech acts together with other actions to achieve goals. This is reflected by architectures that use the FIPA ACL-based JADE framework in robotic applications. Their focus is not planning but rather the interoperability that is gained from using a standardized ACL and the ease of developing agents using JADE.

In contrast, Morgenstern (1988) offers an integrated theory of both types of actions using a *syntactic* first-order logic that includes quotation. Davis and Morgenstern (2005) provide an alternative integration using regular first-order logic. The two theories' semantics cover both speech acts and their content, however their use has so far been limited to a STRIPS-like planner in the case of the former theory, and as a specification for future implementations in the case of the latter.

Temporal Action Logic

Reasoning about both physical and communicative actions is only possible in a highly expressive formalism. We have chosen one such formalism as a foundation, the Temporal Action Logic (TAL), and extended it with a representation of belief and commitment.

The origins of TAL are found in Sandewall's model-theoretic Features and Fluents framework (1994). Doherty (1994) selected important concepts, such as an explicit time line and the use of occlusion (discussed below), to form TAL and gave it a proof-theoretic first-order characterization. Doherty and Kvarnström (2007) provide a detailed account of the logic, but the version presented below includes further extensions that make TAL suitable for applications in multi-agent planning and reasoning.

TAL uses *fluents* to model properties and relations that may change over time. A fluent f is a function of time, and its value at a time point t is given by (value t f). An agent carrying out action a during time interval $(t_1\ t_2]$ is specified by a predicate (Occurs *agent* $(t_1\ t_2]$ a). But the most important feature of TAL is its *occlusion* concept. A fluent that is persistent by default is permitted to change its value when occluded, but must retain its value during time intervals when not occluded. The following formula (with free variables implicitly universally quantified and in prefix form to make the representation of the quoted formulas introduced below more convenient) relates a fluent f's value at the start and end time points of a time interval:

$$(\rightarrow (\neg \text{ (Occlude } (t_1\ t_2]\ f)) (= \text{ (value } t_1\ f) \text{ (value } t_2\ f)))$$

By assuming that fluents are not occluded unless otherwise specified, one is in effect making the frame assumption that things usually do not change. Exceptions are made explicit by occluding affected fluents in action specifications and dependency constraints. E.g., if the UAV flies between two locations, its location fluent (location uav) would be occluded during any interval with a non-empty intersection with the movement interval. By exercising fine-grained control over occlusion one gains a flexible tool for dealing with important aspects and generalizations of the frame problem.

Syntactic Belief

Previous accounts of TAL lack a representation of agents' mental states and beliefs. Introducing a *syntactic* belief operator that takes a *quoted* formula as one of its arguments provides a simple and intuitive notion of beliefs. E.g., the fact that the UAV believes, at noon, that there are five survivors in grid cell 2,3 of the area map can be expressed by the following formula (where 12:00 is syntactic sugar for a Unix time integer):

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

We use the quotation notation from KIF (Genesereth and Fikes 1992), which is a formal variant of Lisp's. An expression preceded by a quote is a regular first-order term that serves as a *name* of that expression. Alternatively one may use a back quote, in which case sub-expressions can be *unquoted* by preceding them with a comma. This facilitates *quantifying-in* by exposing chosen variables inside a back quoted expression for binding by quantifiers. E.g., we can use quantifying-in to say that there is some number that the UAV believes to be the number of survivors:

$$(\exists\ n \text{ (Believes uav 12:00 } \text{`(= (value 12:00 (survivors (cell 2 3))) ',}n))) \qquad (1)$$

Note that it is not the existentially quantified number itself, but the *name* of the number, that should occur as part of the quoted third argument of Believes. The quote preceding the comma ensures that whatever value n is bound to is quoted to produce that value's name.

However, if expressed as above, Formula 1 would be satisfied if the UAV believes all tautologies of the form $x = x$. To see this, simply replace n by the term (value 12:00 (survivors (cell 2 3))) and apply existential generalization. We need to add the requirement that the UAV's belief is not merely a tautology, but that it really *identifies* the number of survivors.

Moore (1980) suggests the use of *rigid designators* and Morgenstern (1987) modifies this suggestion slightly in her requirement that *standard identifiers* are known. We follow Morgenstern and use the syntactic predicate (Id x) to single out the name x as a standard identifier, adding the background knowledge that integers are standard identifiers. It is convenient to introduce a two argument predicate asserting that the second argument is a standard identifier for the first:

$$(\leftrightarrow \text{ (Id ``,}x\text{ ``,}y) (\wedge (= x\ y) \text{ (Id ``,}y)))$$

Note the interaction between back quote and quote in ``,$x$ and ``,y to make sure that the arguments of Id are *names* of the expressions. The initial back quote turns the following quote into the name of a quote, leaving the variables x and y free for binding. The resulting expression denotes the quoted version of whatever the variables are bound to rather than quoted variables that can not be bound at all. The use

of quoted expressions as arguments to the Id predicate prevents substitution of identicals, as is required by opaque belief contexts.

If the UAV can *identify* the number of survivors we say:

(∃ *n* (Believes uav 12:00
'(Id '(value 12:00 (survivors (cell 2 3))) ",*n*)))

Here, the occurrence of the Id predicate is nested in a belief predicate, necessitating *double* quotes on the variable *n*.

Speech Acts

Speech acts can be used to communicate beliefs to, and to incur commitment in, other agents. The extensions introduced above make it possible to reformulate Allen's speech acts (1988) in TAL using syntactic belief and commitment predicates. The type of information we will be interested in is beliefs about what a particular value is. This is straightforwardly communicated by standard identifiers. E.g., if the UAV wishes to inform the CCC that it is in the map's grid cell 2,3 at noon, it may plan an action of the following form:

(inform ccc '(Id '(value 12:00 (location uav)) '(cell 2 3))) (2)

However, this is complicated when the CCC wishes to *ask* the UAV what its location is. Hintikka (1978), and many others, suggests viewing questions as requests for information. The CCC should thus request that the UAV perform the inform action in Formula 2. But since the CCC does not know where the UAV is, which is presumably the reason why it is asking, it can not know what action to request.

Again we follow Allen's directions and introduce an informRef action designed to facilitate questions of this type. The informRef action does not mention the value that is unknown to the CCC, which instead performs the following request:

(request uav '(Occurs uav (*b e*]
(informRef ccc (value 12:00 (location uav)))))

The informRef preconditions require that the informing agent holds a belief about what the value that is being informed about is and the effects assert the existence of a standard name that the hearer believes the value has. Note that an agent that commits to *executing* the action schedules an *inform* procedure call, plugging in the sought value. In contrast, an agent that only *reasons* about the effects of the informRef action, as in the question example above, knows that it *will* come to hold a belief about the value, but need not yet have such a belief.

Automated Natural Deduction

The theory presented so far needs to be complemented with an automated reasoner. Earlier work with TAL made deductive planning possible through a compilation of TAL formulas into Prolog programs (Magnusson 2007). But Prolog's limited expressivity makes it inadequate for our present purposes. Instead, our current work utilizes a theorem prover based on *natural deduction*, inspired by similar systems by Rips (1994) and Pollock (1999).

Natural deduction is an interesting alternative to the widely used resolution method. A natural deduction prover works with the formulas of an agent's knowledge base in their "natural form" directly, rather than first compiling them into clause form. The set of proof rules is extensible and easily accommodates special purpose rules that make reasoning more efficient. E.g., we incorporate specialized inference rules for reasoning with quoted expressions and beliefs. This works well, though the quoted expressions in our examples are simple atomic formulas. Whether the approach will scale to an effective method of reasoning with more general uses of quotation is an open question.

Rules are divided into *forward* and *backward* rules. Forward rules are triggered whenever possible and are designed to converge on a stable set of conclusions so as not to continue generating new inferences forever. Backward rules, in contrast, are used to search backwards from the current proof goal and thus exhibits goal direction. Combined, the result is a bi-directional search for proofs.

Nonmonotonic reasoning and planning is made possible through an assumption-based argumentation system. The set of *abducibles* consists of negated occlusion, action occurrences, temporal constraints, and positive or negative holds formulas, depending on the current reasoning task. These are allowed to be assumed rather than proven, as long as they are not counter-explained or inconsistent.

For some restrictions on the input theory we are able to guarantee completeness of the nonmonotonic reasoning (Magnusson, Kvarnström, and Doherty 2009). But in the general case, when one cannot guarantee completeness of the consistency checking, we might conceivably fail to discover that one of the assumptions is unreasonable. However, this would still not be a cause of *unsoundness*, since we are using the sound system of natural deduction that keeps track of all assumptions and the formulas that follow from them. But it might result in plans and conclusions that rest on impossible assumptions. A conclusion Φ depending on an inconsistent assumption would in effect have the logical form $\bot \rightarrow \Phi$, and thus be tautological and void. This is to be expected though, due to the uncomputability of consistency checking. The most one can hope for is for the agent to continually evaluate the consistency of its assumptions, improving the chances of them being correct over time, while regarding conclusions as tentative (Pollock 1995).

Agent Architecture

Solving the scenario in the introduction requires a system with a tight coupling between planning, execution, and monitoring. Our architecture achieves this through logic-based planning that results in abductive frame assumptions (about the persistence of certain parts of the world) that are monitored during plan execution, as described below.

Planning

Planning is 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, 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 specifications of an action.

The belief precondition is expressed by a single axiom:

(→ (∧ (Primitive *name*) (Id arg_1 id_1) ⋯ (Id arg_n id_n))
(ActionId '(,*name* ,arg_1 ⋯ ,arg_n) '(,*name* ,id_1 ⋯ ,id_n)))

This captures Moore's (1980) insight that knowing identifiers for the arguments of a primitive action is knowing that action. This condition prevents e.g. a stock market agent from planning to get rich by buying "the stock that will increase in value." While theoretically correct, the plan is of no practical value unless the agent can identify some particular stock that will increase in value.

The time point at which an action is executed is also critically important. One would not want the agent to generate a plan to buy a particular stock "when it is lowest" and sell it "when it is highest." Without additional information about when these events occur this plan is equally useless. 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 (1999) 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 execution schedule (described below), or it uses the request speech act to delegate the action to another agent, thereby ensuring commitment.

Execution

Scheduled actions are tied to the STNU through the explicit time points in TAL's Occurs predicate. An STNU *execution* algorithm propagates time windows during which these time points need to occur (Morris and Muscettola 2000). Time points that are *live* and *enabled* with respect to the time windows are executed, i.e. they 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 self (*b e*] *id*))
(Dispatch self (*b e*] *action*))

The ProcedureCall predicate 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. Note that it is the action's identifier that is used in the procedure call. This guarantees that only integers, strings, and other standard identifiers are passed as arguments to procedures, and is necessary since the action's arguments might depend on information gathering actions whose results were not available at the time of planning. Invoking automated reasoning on the above axiom, rather than simply performing the procedure call, allows the full power of theorem proving to be applied in finding standard identifiers for the procedure call arguments. This could be necessary in order to apply background knowledge to convert the arguments into the standard format, or if the arguments are only implicitly represented as a deductive consequence of explicit knowledge.

Monitoring

Executing the plan will satisfy the goal as long as fluent persistence 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 all assumptions that are possible to monitor.

When a persistence assumption (in the form of a non-occlusion formula) fails it produces an occlusion percept that is added to the agent's knowledge base. A truth maintenance system removes assumptions that are contradicted by observations and unchecks goals that were previously checked off as completed but that include a failed assumptions among their dependencies. This immediately gives rise to a plan revision and failure recovery process 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 not 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.

UASTech Delegation Framework

Procedure calls are carried out by the UASTech Delegation Framework (Doherty and Meyer 2007). 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 grid cell using 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 (as described in (Doherty and Rudol 2007)). The assumption is that, while not a primitive action in the low-level system, 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 are coordinated by specialized modules, in this case the scan coordinator.

The Delegation Framework implementation uses the Java agent development framework (JADE) and encapsulates the agent so that all communication is channeled through a standardized interface as FIPA ACL speech acts. Human operators, like those in the CCC, communicate through an interface like any other agent but use a graphical user interface that displays a map subdivided into grid cells through which they can ask questions, position no-fly zones or other constraints, and delegate requests to other agents. The resulting multi-agent system can consist of widely differing agents that are able to interact through standardized speech acts to help each other achieve complex goals.

Scenario Solution

We have implemented this theorem proving based agent architecture and applied it to the scenario described in the introduction. If the UAV's knowledge base was initialized at 12:00 and the CCC requests having information of the survivor count in map grid cell 2,3 at 13:00 the UAV produces the following plan (in addition to an STNU that relates qualitative time points):

(Schedule uav (b_1 e_1] (fly (cell 2 3)))
(Schedule uav (b_2 e_2] (scan (cell 2 3)))
(Schedule uav (b_3 e_3]
  (informRef ccc '(value 13:00 (survivors (cell 2 3)))))

The success of the plan depends on two abductive assumptions that were made during planning and that can be monitored during execution:

(¬ (Occlude (12:00 b_3] (radio uav ccc)))
(¬ (Occlude (e_1 b_2] (location uav)))

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 runs off, then the plan will be modified to take the resulting body count discrepancy into account when it is discovered.

Suppose however that the large distance and mountainous terrain causes a radio communication break down while the UAV is scanning the area. The UAV perceives that the fluent (radio uav ccc) *was* occluded and the truth maintenance system successively removes incompatible assumptions and sub-goals until a revised plan suffix is found:

(Schedule uav (b_4 e_4]
  (informRef mob '(value 13:00 (survivors (cell 2 3)))))
(Schedule uav (b_5 e_5]
  (request mob
    '(Occurs mob (b_6 e_6]
       (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:

(¬ (Occlude (oc 12:00 b_6) (radio mob ccc)))
(¬ (Occlude (oc 12:00 b_4) (radio uav mob)))

While the UAV is incapable of monitoring the other agent's radio communication, it will be monitored if that agent is also running our agent architecture. Unless further failures ensue, this concludes the successful completion of the given knowledge gathering assignment.

Conclusions

We have described a scenario that involves planning communication between agents, plan execution with monitoring, and plan revision to recover from an unexpected communication failure. Our solution uses speech acts formalized in an extension of Temporal Action Logic that includes syntactic belief and commitment operators, which are made possible through the use of a quotation mechanism. Plan generation and revision is carried out using an automated natural deduction theorem prover. The subsequent execution uses an STNU execution algorithm to dispatch actions in accordance with the plan's temporal constraints. Finally, an action dispatch mechanism links the automated reasoning system and a delegation framework that coordinates the execution of primitive actions on the physical robot platform.

Our framework makes extensive use of logic to meet the challenges of dynamic environments. While the use of logic as a *theoretical* foundation is relatively commonplace, we have constructed a *practical* logical agent architecture that uses theorem proving technology to plan and execute actions in a multi-agent setting.

Much work remains before the technology is sufficiently efficient and robust for larger scale applications. But there is great potential for using logical agents in both real and simulated worlds. This paper has explored a robotic search and rescue scenario. Another paper uses the same technology in intelligent computer game characters (Magnusson and Doherty 2008). We believe these and similar opportunities make continued effort worthwhile.

References

Allen, J. 1988. *Natural Language Understanding*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.

Davis, E., and Morgenstern, L. 2005. A first-order theory of communication and multi-agent plans. *Journal of Logic and Computation* 15(5):701–749.

Doherty, P., and Kvarnström, J. 2007. Temporal action logics. In Lifschitz, V.; van Harmelen, F.; and Porter, B., eds., *Handbook of Knowledge Representation*. Elsevier.

Doherty, P., and Meyer, J.-J. C. 2007. Towards a delegation framework for aerial robotic mission scenarios. In *Cooperative Information Agents XI*, 5–26.

Doherty, P., and Rudol, P. 2007. A UAV search and rescue scenario with human body detection and geolocalization. In *Australian Conference on Artificial Intelligence*, volume 4830 of *Lecture Notes in Computer Science*, 1–13. Springer.

Doherty, P. 1994. Reasoning about action and change using occlusion. In *Proceedings of the 11th European Conference on Artificial Intelligence*, 401–405.

Foundation for Intelligent Physical Agents. 2002. FIPA communicative act library specification. `http://www.fipa.org/specs/fipa00037/`.

Genesereth, M. R., and Fikes, R. E. 1992. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University.

Genesereth, M. R., and Ketchpel, S. P. 1994. Software agents. *Communications of the ACM* 37(7):48–53.

Hintikka, J. 1978. Answers to questions. In Hiz, H., ed., *Questions*. D. Reidel Publishing Company. 279–300.

Magnusson, M., and Doherty, P. 2008. Logical agents for language and action. In *Proceedings of the 4th Artificial Intelligence and Interactive Digital Entertainment Conference*.

Magnusson, M.; Kvarnström, J.; and Doherty, P. 2009. Abductive reasoning with filtered circumscription. In *Proceedings of the 8th Workshop on Nonmonotonic Reasoning, Action and Change NRAC 2009*. UTSePress. Forthcoming.

Magnusson, M. 2007. *Deductive Planning and Composite Actions in Temporal Action Logic*. Licentiate thesis, Linköping University. `http://www.martinmagnusson.com/publications/magnusson-2007-lic.pdf`.

Moore, R. 1980. Reasoning about knowledge and action. Technical Report 191, AI Center, SRI International, Menlo Park, CA.

Morgenstern, L. 1987. Knowledge preconditions for actions and plans. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, 867–874.

Morgenstern, L. 1988. *Foundations of a logic of knowledge, action, and communication*. Ph.D. Dissertation, New York, NY, USA. Advisor: Ernest Davis.

Morris, P. H., and Muscettola, N. 2000. Execution of temporal plans with uncertainty. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, 491–496.

Perrault, C. R.; Allen, J. F.; and Cohen, P. R. 1978. Speech acts as a basis for understanding dialogue coherence. In *Proceedings of the 1978 workshop on Theoretical issues in natural language processing*, 125–132.

Pollock, J. L. 1995. *Cognitive Carpentry: A Blueprint for how to Build a Person*. Cambridge, MA, USA: MIT Press.

Pollock, J. 1999. Natural deduction. Technical report, Department of Philosophy, University of Arizona. `http://www.sambabike.org/ftp/OSCAR-web-page/PAPERS/Natural-Deduction.pdf`.

Rips, L. J. 1994. *The psychology of proof: deductive reasoning in human thinking*. Cambridge, MA, USA: MIT Press.

Sandewall, E. 1994. *Features and Fluents: The Representation of Knowledge about Dynamical Systems*, volume 1. Oxford University Press.

Searle, J. R. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

Shanahan, M. 2000. Reinventing Shakey. In *Logic-Based Artificial Intelligence*. Norwell, MA, USA: Kluwer Academic Publishers. 233–253.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: From consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence* 11(1):23–45.