ELECTRONIC NEWSLETTER ON
REASONING ABOUT ACTIONS AND CHANGE

Issue 00005 Editor: Erik Sandewall 26.7.2000

Today

Congratulations today to Camilla Schwind for the acceptance of her article "Causality in Action Theories" for publication in the ETAI. Both referees recommended acceptance; one of them suggested minor changes to the article.

The relationship between Actions and Change research on one hand, and Planning and Scheduling research on the other may be much closer in the future than it has been so far. The recent Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000) featured a planning systems competition where logic-based systems obtained much better results than what was previously thought to be possible. Today's issue of this Newsletter contains a report by Fahiem Bacchus, who organized the competition, on the competition results and what conclusions can be drawn from them. The report follows in plain text. More easy-to-read versions in HTML and in Latex-based postscript are available as usual on the ETAI web page.

New Trends in Research

The Role of Logic in Planning

by Fahiem Bacchus

Introduction

Recently I was involved with organizing and running the second biannual planning competition that was held in conjunction with the Fifth International conference on Artificial Intelligence Planning and Scheduling (AIPS2000). This years competition was much larger in scale that the previous competition, held in 1998, and like the previous competition it told us a lot about the potential of various approaches to planning.

The topic of this paper is to examine what the competition tells us about logic and its role in planning. Logic and logical approaches have long been criticized as been computationally ineffective, yet some of the best performing planners in the competition were based on logic. So it would seem that there is reason for researchers in logic-based AI to be optimistic.

This optimism needs to be guarded, however. Closer examination indicates that in these systems the scope of logical reasoning is severely restricted, and

I suspect that similar restrictions are at the heart of almost all computationally successful applications of logic. In particular the reasoning performed in these systems has more in common with model checking (i.e., checking whether or not a *single* model satisfies a formula [HV91]), than with what is classically viewed as logical reasoning, i.e., the application of deductive proof theories to produce conclusions true for an entire set of models.

I will explain how the results of the competition lead to these conclusions. These conclusions agree with views I have held and that have driven much of my own research for a number of years.

The problems addressed in the planning competition

The competition was aimed at solving classical planning problems. In particular, it consisted of planning problems where there is a fixed and known initial state, and a goal consisting of a finite conjunction of ground atomic formulas. Clearly, classical planning is only a subset of the types of planning problems that we need to be able to solve in AI applications. Nevertheless, classical planning of this sort has been a very computationally challenging problem since the very earliest days of AI. It is only very recently that real progress has been made on being able to solve reasonably large planning problems of this type.

For more information about the planning domains, the problem suites, the competitors, and the results of the competition please see the competition home page at www.cs.toronto.edu/aips2000. A special section of the AI-Magazine is in preparation. It will contain more in-depth information about the competition.

All of the planning domains were specified in a syntactic variant of ADL, called PDDL (planning domain description language [DM98]) . ADL (action description language) is a extension of STRIPS [FN71] due to Pednault [Ped89]. ADL extends STRIPS to allow for universally quantified and conditional effects, among other things. Pednault also proposed a method for updating functional fluents, but these are not currently part of PDDL, and so were not used in the competition. One interesting thing about ADL is that Pednault proposed it as a formalism that lies between the "non-logical" STRIPS representation and the purely logical situation calculus.[1] In particular, given a mechanism for solving the frame problem (e.g., [Rei91]), it is not difficult to translate a collection of ADL operators into a logical theory.

The key component of ADL (for this discussion) is that it can be viewed as specifying for each possible state of the world a fixed and finite set of fluent updates, very much like the original STRIPS representation. In particular, if we have a full specification of the current state of the world, i.e., a single model, we can use model checking to compute a finite set of ground atomic facts that must be negated (deleted in the STRIPS vernacular), a finite set of atomic facts that must be made true (added), and every other ground

[1]Although there are many different logical formalisms for specifying action theories, for concreteness, I will restrict my attention to logical theories specified in the situation calculus.

atomic fact is unchanged. Consider, e.g., the ADL action[2]

```
(adl-operator (move-briefcase Loc1)
  (pre (and (at briefcase Loc0)
            (location Loc1)
            (not (= Loc1 Loc0))))
  (add (at briefcase Loc1))
  (del (at briefcase Loc0))
  (forall (?z)
    (implies (in-briefcase ?z)
             (and (add (at ?z Loc1))
                  (del (at ?z Loc0))))))
```

For any state s, if

$s \not\models$ `(and (at briefcase Loc0) (location Loc1) (not (= Loc1 Loc0)))`

the action cannot be applied to s—s does not satisfy its precondition. On the other hand if s does satisfy the precondition, then the set of facts to be made false is `(at briefcase Loc0)` as well as `(at C Loc0)` for all objects `C` such that $s \models$ `(in-briefcase C)`. Similarly the set of facts to be made true includes `(at briefcase Loc1)` and `(at C Loc1)` for the same set of objects `C`. All other facts remain unchanged.

Three things are to be noted. First, for any fixed state s we can determine the complete set of atomic formulas whose truth value changes by checking whether or not s satisfies certain specific formulas. Second, since a model is entirely determined by the truth values it assigns to atomic facts we see that the ADL action generates a unique successor model (state). And third, if we have an explicit representation of the current model (state) (e.g., a database of all the true atomic facts that hold in the model, or a horn clause theory to which we apply completion) we can construct an explicit representation of the successor state.[3]

A full blown logically specified action theory need not satisfy any of these properties. For example it is quite possible that such a logical theory only partly specifies the successor state, so that even if we have full knowledge of the current state, we do not have full knowledge of the successor state.[4]

So we see that even before we examine the logical reasoning that is occurring inside of the planning systems, the problems being addressed fall into a restricted logical class.

Of course, just because the planning competition utilized only a restricted class of action theories is no reason to suppose that logic based techniques

[2]Normally ADL actions are specified as ADL operators. Operators contain unbound variables, and every instantiation of these variables generates a unique action. For simplicity we present a fully instantiated action.

[3]It should be noted that not all planning systems use these properties of ADL actions.

[4]This could be the case even if the action is deterministic. For example, one could specify that the `move-briefcase` either moves the briefcase to `Loc1` or to another nearby location, say `Loc2`. Thus even if the action deterministically transforms a situation to a unique new situation, we might not know some of the properties of that new situation.

could not be used on less restricted theories. However, I suspect that restrictions of this form are essential for computational performance. For example, the restriction that actions preserve complete knowledge is part of Reiter's [Rei91] approach to the frame problem. In particular, Reiter's solution involves using a completeness assumption to construct an successor state axiom for each fluent. This axiom specifies an *equivalence* between the status of each fluent in the current situation and the successor situation. Hence, if we have complete knowledge of the fluents in the current situation (e.g., if we have a single model for the current situation) we will have complete knowledge of the fluents in any successor situation.

State constraints are one seemingly necessary construct that move us out of the restricted class of action theories addressed in the planning competition. It is worth noting however, that most ADL specified planning domains deal with state constraints by adding them into the action descriptions. In the above `move-briefcase` action, the state constraint that an object can only be in one location at a time is embedded in the action specification: the action deletes the old location of each of the objects it moves. This approach to handling state constraints also shows up in purely logical approaches. For example, Lin and Reiter [LR94] present an approach where state constraints are embedded into the fluent successor state axioms. The end result is that one once again achieves an action theory that preserves complete knowledge.

The Successful Planning Systems

The planning competition was divided into two separate competition tracks. The first track was for fully automated systems: the planners were required to generate their plans given only a specification of the actions available in the domain. In the second track, the planning systems could be provided with additional "hand tailored" domain specific information to help them plan.

This difference leads to incomparable results between the two tracks. So the performance of the systems in each track was assessed separately, and separate prizes were awarded for each track. However, in both tracks the prize winning planning systems utilized the properties of the ADL operators explained in the previous section. In particular, from any completely specified state the actions can be used to generate a new set of complete specified successor states. Both of the prize winning planners searched for plans in this space: the space of completely specified states reachable from the initial state. Each sequence of actions generates a specific reachable state, and all that the planners need do is to search for a reachable state satisfying the goal: the plan is the sequence of actions used to generate that state.

This approach to planning is often called forward chaining, and it has been around for a very long time. For most of this time it was dismissed as being impractical. The original STRIPS system used means-ends analysis, which is a mixture of forward chaining and backwards regression of the goal. Various developments lead to partial order planning [MR91, PW92] which was at one time considered to be most promising approach by the planning community. More recently, planners based on the combinatorial reasoning approach of GraphPlan [BF97] and SatPlan [KS96] were shown to have considerably better performance than partial order planners and became a focus of much research. This approach to planning dominated the previous 1998 planning competition. In this year's competition, however, it was the forward chaining planners that gained ascendancy.[5]

Why has forward chaining been so successful in this years competition? I would argue that the prize winning planners have found successful ways of utilizing the fact that forward chaining provides one with a fully specified model of the current state of the world.

In 1994 when interest in partial order planning was near its height, a colleague (Froduald Kabanza) and I realized that just as a complete state can be utilized to efficiently check the satisfaction of a logical formula, sequences of complete states could be utilized to efficiently check the satisfaction of formulas of temporal logic. We then concluded that useful domain specific control knowledge could be expressed as formulas of temporal logic (we utilized a first order version of linear temporal logic [MP92]) and then checked against the sequences of states generated by a forward chaining engine. If the sequence of states violates the temporal formula that sequence could then be pruned, greatly reducing the size of the search space. This approach yielded the TlPlan planning system (www.cs.toronto.edu/~fbacchus/tlplan.html).[6] We first reported on the TlPlan system at the 1995 European Workshop on Planning [BK96], and recently some journal length descriptions of our approach have appeared [BK98, BK00].

The approach of using logical formulas to specify control formulas yielded tremendous improvements in performance, and for a number of years the TlPlan system was the most effective planning system available. This years planning competition has verified the validity of this approach as the prize winning planner in the hand tailored track is based precisely on the TlPlan approach.

This prize winning planner was a system called TALPlanner due to Patrick Doherty, Jonas Kvarnstrom, and Patrik Haslum of Linkoping University, Sweden. The basic operation of TALPlanner is exactly the same at that of TlPlan: it utilizes domain specific knowledge expressed as formulas of a temporal logic to prune sequences of states from the search space explored by forward chaining. TALPlanner also included some significant improvements over TlPlan[7] In particular, it utilized a more efficient way of checking whether or not the state sequences violated the temporal formulas, and it also utilized different data structures for representing the states that supported more efficient formula checking.[8] The important point here, however, is that TALPlanner used logic to represent knowledge to support sophisticated logical reasoning. The "logical reasoning" it performed was confined

[5] There has also been a parallel development of HTN (hierarchical task network) planners, e.g. [NCLMA99].

[6] TlPlan was not entered in the planning competition since that would have caused an uncomfortable conflict of interest.

[7] Thus even if Tlplan had been entered in the competition, I am certain that TALPlanner would have still been the faster planner.

[8] TALPlanner also employed a different temporal logic, but I don't think that this is a significant difference.

to model checking formulas against specific models.

The TlPlan approach is intrinsically dependent on domain-specific knowledge, and one of the things that we have demonstrated in our TlPlan work is that useful control knowledge is available for many different planning domains. The prize winning planner in the fully automated track was not able to utilize domain-specific knowledge to guide it search for a plan. Instead it utilized mechanisms for automatically computing heuristic guidance. Around the same time as my own work on TlPlan, McDermott was also trying to exploit the complete knowledge of the current state that forward chaining search provides. He developed a mechanism for computing a heuristic estimate of the distance between the current state and a goal state using an means-ends analysis [McD96]. This estimate was used to guide the search. An alternate method for computing such heuristics was later explored by Geffner et al. [BLG97]. The prize winning planner in the fully automated track was a system called FF (Fast Forward) developed by Joerg Hoffmann of Albert Ludwigs University. Hoffmann discovered a different and more successful way of generating heuristics estimates. His method used some of the ideas from GraphPlan, but he used these ideas to compute heuristics for forward chaining rather than for direct planning.

Logical Reasoning

TALPlanner was by far the fastest planner in the competition (although it is of course unfair to compare it directly with the fully automated planners). And it is a planning system that, like its predecessor TlPlan, makes heavy use of logic. So we return to the question of what this says about logic and its role in planning.

As I indicated in the previous section, logic is used in a very restricted manner in TALPlanner. The main use of logic in this planner is for knowledge representation, there is very little traditional logical-reasoning performed in the system. Rather, models are constructed during search and logic is used to express formulas used to query those models.[9]

This is not to belittle the role of logic in this system. Rather, as I have argued in my work on TlPlan, logic is essential for this process. By writing the domain specific knowledge in a formal logic, one obtains a clear and formal semantics for this knowledge. Furthermore, logical languages, especially those with quantification, are tremendously expressive. This expressiveness is vital in allowing us to represent the various pieces of commonsense domain knowledge we seem to possess about almost every domain.

The bottom line is that unbridled logical reasoning has an exponential appetite for computational resources. We can never satisfy such an appetite—there has to be a systematic way of restricting the amount of logical reasoning our systems perform. Sometimes specific types of reasoning can be supported with syntactically restricted computationally tractable logics, e.g., terminological reasoning. But in general I am skeptical of syntactic

[9]Expressing queries of a model as logical formulas is, of course, also foundation of database technology. Formal treatments of query languages generally treat queries as logical expressions and databases as models.

restrictions, as it seems to me that expressiveness is the key feature that makes logical languages so powerful. In the planning domain, and in search problems in general, being able to reduce the problem down to a specific set of models and thus reduce logical reasoning to query evaluation on these models, is an approach that yields tractable reasoning with no penalty in expressiveness. As the TlPlan and TALplanner systems demonstrate this is seems to be a very fruitful approach to solving planning problems.

Is there any role for more complex "open-ended" logical reasoning in planning systems? Yes I believe that to a certain extent there is. Consider the fully automated planner FF. FF displayed very impressive performance in the competition while computing the control knowledge it required on the fly, as it solved the planning problem. TALPlanner on the other hand was given its control knowledge ahead of time, and it needed only to do formula evaluation to apply that knowledge as it solved the planning problem. There would seem to be much room for the middle ground. In particular, I believe that "off-line" logical reasoning could be used to precompute much of the control knowledge TALPlanner utilizes. Much of this knowledge is a logical consequence of fact that the specified domain operators are the only mechanisms of change in the domain. By doing this reasoning off-line, prior to planning, one would be able to devote more computational resources to support more complex logical reasoning.

Nevertheless, even off-line reasoning must be restricted or guided in various ways. It is only by understanding the particular structural features of the problem, or class of problems, that we can achieve the insights required to restrict the logical inferences we need to perform. Thus I am pessimistic about the approach of converting our problems into a logical theory and then doing general logical reasoning with that theory to draw the conclusions we desire. Rather I see more promise in the approach of understanding the specific structural properties of the problems we are addressing, and then using those (mostly semantic) insights to develop reasoning mechanisms tailored to those problems. For classical planning problems, search in the forward chaining space seems to a powerful way of restricting the scope of logical reasoning required.

It is true that the specific reasoning mechanisms we might develop for particular problems can often be mimicked by an appropriate theorem proving strategy. But my argument is that the insights required to find an effective theorem proving strategy (reasoning strategy) are most effectively obtained by thinking about the problem domain rather than by thinking about the logical theory the problem domain gives rise to. For example, we could duplicate forward chaining search in Green's generic approach to planning as theorem proving [Gre69]. All that would be required would be realize that the critical choice points in a search for a proof are the action choices. However, it is unlikely one would arrive at that intuition if one simply examined the situation calculus theory. Rather it is only by understanding the semantic properties of the domain that one can realize that the actions play such a distinguish role.

In conclusion, I think that the planning competition demonstrated that logical approaches to AI display many of the advantages long claimed of them. In particular, logic allows us to represent in a semantically clear way large amounts of common sense knowledge. In general, logical approaches are extremely promising when we are able to place strong restrictions on the amount of logical reasoning we need to do. Most interestingly, in the domain of planning it is extra knowledge represented *in logic* that allows us to achieve these restrictions.

References

[BF97] Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

[BK96] Fahiem Bacchus and Froduald Kabanza. Using temporal logic to control search in a forward chaining planner. In M. Ghallab and A. Milani, editors, *New Directions in AI Planning*, pages 141–153. ISO Press, Amsterdam, 1996.

[BK98] Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. *Annuals of Mathematics and Artificial Intelligence*, 22:5–27, 1998.

[BK00] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116:123–191, 2000.

[BLG97] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *Proceedings of the AAAI National Conference*, pages 714–719, 1997.

[DM98] et al. Drew McDermott. *PDDL—The Planning Domain Definition Language*. Yale University, 1998.

[FN71] Richard Fikes and Nils Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Gre69] Cordell Green. Application of theorem proving to problem solving. In *Procceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, pages 219–239, 1969.

[HV91] J. Y. Halpern and M. Y. Vardi. Model checking vs. theorem proving: a manifesto. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 325–334. Morgan Kaufmann, San Mateo, California, San Mateo, CA, 1991.

[KS96] Henry Kautz and Bart Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the AAAI National Conference*, pages 1194–1201, 1996.

[LR94] F. Lin and R. Reiter. State constraints revisited. *Journal Of Logic And Computation*, 4(5):655–678, 1994.

[McD96] D. McDermott. A heuristic estimator for means-end analysis in planning. In *Proceedings of the Third International Conference on A.I. Planning Systems*, 1996.

[MP92] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems: Specication.* Springer-Verlag, New York, 1992.

[MR91] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings of the AAAI National Conference*, pages 634–639, 1991.

[NCLMA99] Dana S. Nau, Yue Cao, Amnon Lotem, and Hector Munoz-Avila. Shop: Simple hierarchical ordered planner. In *Procceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, pages 968–975, 1999.

[Ped89] E. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 324–332, 1989.

[PW92] J.S. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for adl. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 103–114, 1992.

[Rei91] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Vladmir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

This newsletter is issued whenever there is new news, and is sent by automatic E-mail and without charge to a list of subscribers, in plaintext format. The present postscript issue is a bona fide rendering of the same information in alternative typography. Instruction for contributors and other additional information is found at the following URL:

`http://www.etaij.org/rac/`