AI Magazine Volume 22 Number 3 (2001) (© AAAI)

Planning in the Fluent Calculus Using Binary Decision Diagrams

Hans–Peter Störr

■ BDDPLAN was created to perform certain reasoning processes in the fluent calculus, a flexible framework for reasoning about action and change based on first-order logic with equality (plus some second-order extensions in some cases). The reasoning is done by mapping the problems into propositional logic, which, in turn, can be implemented as operations on binary decision diagrams (BDDs).

As a part of my group's activities on the extension of the fluent calculus (Hölldobler and Schneeberger 1990; Thielscher 1998), I've been working to provide an efficient inference engine for the fluent calculus by using the efficiency of binary decision diagrams (BDDs) (Bryant 1986). In the past, BDDs have significantly improved the performance of algorithms and enabled the solution of new classes of problems in areas such as formal verification and logic synthesis (see, for example, Burch et al. [1992]). Surprisingly, BDDs have only recently been introduced to implement the solution of planning problems (Cimatti et al. 1997). The goal of our project was to investigate whether BDDs might also help to increase the efficiency of algorithms solving problems in the field of reasoning about action and change. For a start, I have implemented the solution of fluent calculus planning problems restricted to deterministic actions and propositional fluents (Hölldobler and Störr 2000; Störr 2001).

Binary Decision Diagrams

The idea of BDDs is similar to decision trees: A Boolean function is represented as a rooted acyclic-directed graph. Figure 1 shows a BDD for the function $(a \wedge b) \vee (c \wedge d)$. For a given valuation of the propositional variables occurring in the function, the value of the Boolean function represented by the BDD is obtained by traversing the diagram starting from the root and taking at each node the edge labeled with the value of the variable occurring in the node. The label of the terminal node defines the value of the function under the current valuation. For example, $<a \rightarrow \perp, b \rightarrow \perp, c \rightarrow \top, d \rightarrow \perp>$ leads to a node labeled 0, that is, the value of the formula is 0 with respect to this valuation.

The difference between BDDs and decision trees is that there is a fixed order of the occurrences of variables in each branch of the diagram and that isomorphic substructures of the diagram are represented only once. The use of BDDs can lead to exponential savings in space in comparison to representations such as decision trees or disjunctive or conjunctive normal form. In practical applications, logical operations on a BDD representation are often much more efficient than in these other representations. Thus, BDDs are an efficient implementation tool for algorithms semantically based on Boolean functions.

Fluent Calculus Planning and Binary Decision Diagrams

The crucial idea of the fluent calculus is to represent the states of the world at term level by combining all fluents that hold in the state with an associative and commutative function symbol $\circ$. For example, if the world is warm and cozy, its state is represented as a term *warm* $\circ$ *cozy*. In the fluent calculus axiomatization, such a term denotes effectively a finite multiset of fluents, that is, in my example {*warm*, *cozy*}. Thus, modeling change is

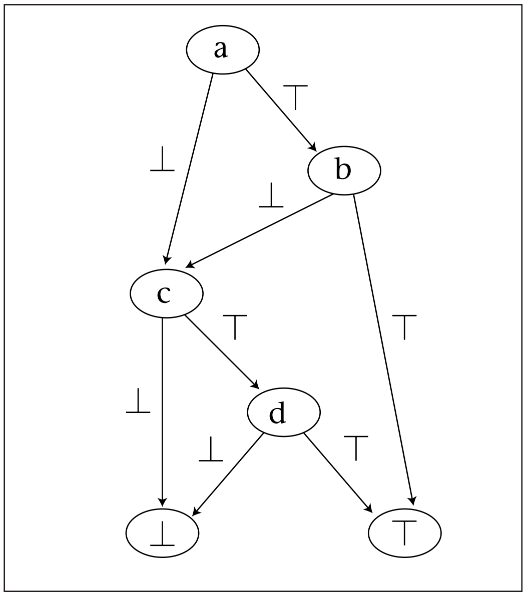


Figure 1. A Binary Decision Diagram for $(a \wedge b) \vee (c \wedge d)$.

The current implementation as a symbolic breadth-first search has the advantage that it always finds the shortest plan and is able to deal with exponentially long plans.

mapped to a rewriting of terms, which proves to be a flexible axiomatization technique (Thielscher 1998) and has computational advantages.

A planning problem in the fluent calculus amounts to a proof of a logical consequence:

$$\Delta \models t = state(s) \tag{1}$$

where Δ is a fluent calculus axiomatization of the considered planning; *t* is a term representing a state that fulfills the goal of the problem; *s* is a situation, that is (roughly), a term that encodes an action sequence; and *state*(*s*) denotes the state holding in a situation *s*. The semantics of *state* is determined by Δ. Please note that equation 1 declares that state *t* is reachable in the planning problem because it can be reached by the action sequence encoded in *s*.

The process applied in my system is currently a straightforward breadth-first search over the set of reachable states, which amounts to generating all terms *t* occurring in consequences of the form (equation 1). The calculation process is stepwise: The sets of states reachable from the initial state after execution of *n* actions are consecutively generated.

The crucial point in my application of BDDs is that one can represent sets of terms representing states as a BDD whose size is in many practical cases much smaller than the set itself. The basic idea of this encoding is that each term can be represented uniquely by a valuation of a set of propositional variables, such that for each fluent *f* the propositional variable z_f is true iff *f* occurs in the term. For example, the term warm ∘ cozy is represented by a valuation

$$\{z_{warm} \to \top,\ z_{cozy} \to \top\}$$

Thus, a set of terms can be represented by a Boolean function that is true for a valuation iff the corresponding term is in the set. For example, the set

{*warm* ∘ *cozy, warm*}

is represented by the Boolean function

$$(z_{\mathrm{warm}} \wedge z_{cozy}) \vee (z_{\mathrm{warm}} \wedge \neg z_{cozy}),$$

which can, in turn, be represented by a BDD. The encoding is discussed in full detail in Hölldobler and Störr (2000).

It is well known that the manipulation of sets given by their characteristic function can be performed by logical operations on this function. Thus, the manipulation of term sets encoded by Boolean functions, as discussed, can be implemented by BDD operations. Furthermore, I devised a BDD-based way to implement the rewriting of terms taking place in our domain specification Δ to describe actions. Thus, I have been able to map the planning process onto the manipulation of Boolean functions in BDD representation. The algorithm is closely related to model-checking algorithms (Burch et al. 1992) that perform symbolic breadth-first search in the state space.

Implementation Remarks

To enable comparison of my planning algorithm, I developed a fluent calculus semantic for the ADL fragment of PDDL (Störr 2001), which is the base for my planner BDDPLAN. The implementation was done in C++ using the University of California at Berkeley CAL BDD package.

An important factor for the efficiency of BDD algorithms is the order in which the variables occur in the BDD because good variable orderings can lead to exponentially smaller BDDs. For our purposes, I developed an ordering principle called *sort order* that leads to good results on many problems. We are still in the process of investigating how optimization techniques well known in the area of model checking using BDDs can be adapted such that they increase the efficiency of the implementation. Disjunctive partitioning of the transition relation and frontier simplification have successfully been applied to date.

Discussion

I have started to investigate how reasoning problems in the fluent calculus can be supported by a BDD reasoning engine. A general problem with BDD algorithms is their tendency to blow up: The size of BDDs can be exponential in the number of propositional variables of the encoded Boolean function, so they can get unmanageable on more complex problems. There are quite a number of techniques to alleviate this problem, but it is limiting the use of BDDs. Still, we received promising results for some planning problems. The current implementation as a symbolic breadth-first search has the advantage that it always finds the shortest plan and is able to deal with exponentially long plans (such as in the towers of Hanoi problem).

Thus, it seems that BDDs are a valuable tool to use in planning algorithms, but they have their limitations. I envision the appropriate use of BDDs in a toolbox of algorithms from which, for each planning problem, the appropriate one can be chosen to solve the problem or which can work concurrently on the problem until a solution is found.

References

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers* 8(C-35): 677–691.

Burch, J.; Clarke, E.; McMillan, K.; and Dill, D. 1992. Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computation* 98(2): 142–170.

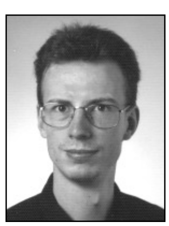
Cimatti, A.; Giunchiglia, E.; Giunchiglia, F.; and Traverso, P. 1997. Planning via Model Checking: A Decision Procedure for AR. In *Proceedings of the Fourth European Conference on Planning (ECP97)*, eds. S. Steel and R. Alami, 130–142. Lecture Notes in Artificial Intelligence 1348. New York: Springer-Verlag.

Hölldobler, S., and Schneeberger, J. 1990. A New Deductive Approach to Planning. *New Generation Computing* 8(3): 225–244.

Hölldobler, S., and Störr, H.-P. 2000. Solving the Entailment Problem in the Fluent Calculus Using Binary Decision Diagrams. In *Proceedings of the First International Conference on Computational Logic (CL),* eds. J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, and P. J. Stuckey, 747–760. Lecture Notes in Computer Science 1861. Berlin: Springer-Verlag.

Störr, H.-P. 2001. Planen mit binären Entscheidungsdiagrammen (Planning with Binary Decision Diagrams). Ph.D. dissertation, Department of Computer Science, Dresden University of Technology. Forthcoming.

Thielscher, M. 1998. Introduction to the Fluent Calculus. *Electronic Transactions on Artificial Intelligence* 2(3–4): 179–192. Available at www.ep.liu.se/ea/cis/1998/014/ .



Hans-Peter Störr is a research assistant at Dresden University of Technology investigating applications of AI in electronic commerce. His current research interests include binary decision diagrams, nonmonotonic logic, machine learning, fuzzy logic, and personalization issues. His e-mail address is hans-peter@inf.tu-dresden.de.

Call for Proposals

AAAI-02 Tutorial Forum

Eighteenth National Conference on Artificial Intelligence

July 28–August 1, Shaw Convention Center, Edmonton, Alberta, Canada

Sponsored by the American Association for Artificial Intelligence

The AAAI-02 Program Committee invites proposals for the Tutorial Forum of the Eighteenth National Conference on Artificial Intelligence (AAAI-02). The Tutorial Forum will be held July 28-29, 2002 in Edmonton, Alberta, Canada. Anyone interested in presenting a tutorial at AAAI-02 should submit a proposal to Michael Littman, 2002 Tutorial Chair, at the address below.

What Is the Tutorial Forum?

The Tutorial Forum provides an opportunity for junior and senior researchers to spend two days each year freely exploring exciting advances in disciplines outside their normal focus. We believe this type of forum is essential for the cross fertilization, cohesiveness, and vitality of the AI field. We all have a lot to learn from each other; the Tutorial Forum promotes the continuing education of each member of the AAAI. Once again this year, attendance at the tutorials will be included in the AAAI-02 technical registration fee.

Topics

AAAI is interested in proposals for advanced tutorials at the leading edge of AI. We especially encourage tutorials taught by a strong team of two established researchers, providing a balanced perspective on a core research topic. We are interested in tutorials that summarize recent technical advances in active core areas of AI. We are also interested in tutorials that educate the AI community about emerging opportunities, technologies and problem areas that are in their adolescence, such as intelligent real-time systems, education, AI and entertainment, knowledge acquisition and information gathering on the web, and particularly those topics we didn't imagine to mention. We are equally looking for tutorials about core methods from other computational and scientific disciplines that promise a strong synergy with AI methods, such as traditional computer science, operations research, cognitive psychology, etc. We are very interested in exploring innovative proposals for educational approaches that go beyond the traditional format of four-hour tutorials, exploiting the flexibility that a one-fee program offers.

AAAI-2000's forum included tutorials on: probabilistic robotics; practical tools for knowledge representation and nonmonotonic reasoning; new frontiers in statistical natural language processing; foundations of electronic market; approximation techniques for automated reasoning; text mining; solving and programming with soft constraints; vision-based interaction and control; recent advances in AI planning; text summarization; empirical methods for artificial intelligence and computer science; conceptual modeling and ontological analysis; and user modeling and adaptive interfaces. This list serves merely as an example. We are looking for continued innovation in the forum's program that incorporates novel and under-represented topic areas.

Submission Requirements

We need two kinds of information in the proposals: information that will be used for selecting proposals and information that will appear in the tutorial description brochure. The proposal should provide sufficient information to evaluate the quality of the technical content being taught, the quality of the educational material being used, and the speakers' skill at presenting this material. Each proposal should include at least the following:

- *Goal of the tutorial:* Who is the target audience? What will the audience walk away with? What makes the topic innovative?
- *Content:* Detailed outline and list of additional materials, augmented with samples, such as past tutorial slides and survey articles, whenever possible. Be as complete as possible.
- *Tutorial description:* A short paragraph summarizing the tutorial outline.
- *Prerequisite knowledge:* What knowledge is assumed.

Please also submit the following information about the team of presenters: name, mailing address, phone number, e-mail address; background in the tutorial area, including a list of publications and/or presentations; any available examples of work in the area (ideally, a published tutorial-level article or presentation materials on the subject); evidence of teaching experience (courses taught or references); and evidence of scholarship in AI or computer science.

Submission Deadline

Proposals must be received by October 15, 2001. Decisions about the tutorial program will be made by November 30, 2001. Speakers should be prepared to submit completed course materials by May 24, 2002.

Please e-mail proposal material to the tutorial chair at the following address. Hard copy submissions will also be accepted:

Michael Littman
AT&T Labs Research
Shannon Laboratory, Room A275
180 Park Avenue
Florham Park, NJ 07932-0971
Telephone: (973) 360-8312
Fax: (973) 360-8970
E-mail: mlittman@research.att.com