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

MIPS
The Model-Checking Integrated Planning System

Stefan Edelkamp and Malte Helmert

■ MIPS is a planning system that applies binary decision diagrams (BDDs) to compactly represent world states in a planning problem and efficiently explore the underlying state space. It was the first general planning system based on model-checking methods. It can handle the STRIPS subset of the PDDL language and some additional features from ADL, namely, negative preconditions and (universal) conditional effects. At the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00), MIPS was one of five planning systems to be awarded for distinguished performance in the fully automated track. This article gives a brief introduction to, and explains the basic planning algorithm used by, MIPS, using a simple logistics problem as an example.

With the model-checking integrated planning system (MIPS), model checking has eventually approached classical AI planning. It was the first planning system based on formal verification techniques that turned out to be competitive with the various GRAPHPLAN- or SAT-based approaches on a broad spectrum of domains.

MIPS uses binary decision diagrams (BDDs), introduced by Bryant (1986) to compactly store and operate on sets of states. More precisely, it applies reduced ordered BDDs, which we refer to simply as BDDs for the rest of this article.

Its main strength compared to other, similar approaches lies in its precompiling phase, which infers a concise state representation by exhibiting knowledge that is implicit in the description of the planning domain (Edelkamp and Helmert 1999). This representation is then used to carry out an accurate reachability analysis without necessarily encountering exponential explosion of the representation.

The original version of MIPS, presented at the Fifth European Conference on Planning (ECP'99), was capable of handling the STRIPS subset of PDDL. It was later extended to handle some important features of ADL, namely, domain constants, types, negative preconditions, and universally quantified conditional effects.

Other extensions include two additional search engines based on heuristics, one incorporating a single-state hill-climbing technique similar to Hoffmann's FF, the other one making use of BDD techniques, thus combining heuristic search with symbolic representations. However, because the first search engine does not contribute many new ideas, its merits mainly lying in the combination of Hoffmann's heuristic estimate with the preprocessing techniques of MIPS, we won't dwell on it.

Neither do we say much about the symbolic heuristic search techniques included in MIPS, namely, the BDDA* and PURE BDDA* algorithms. These techniques were disabled in the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'00) planning competition in favor of the original MIPS planning algorithm, partly because it turned out to perform better on some domains and partly because it always yields optimal (sequential) plans, which we consider an important property of the planner that counterbalances some of its weaknesses in performance compared to other current planning systems such as FF. Readers interested in these parts of the MIPS planning system are referred to Edelkamp and Helmert (2000).

Thus, in the following sections, we cover the core of mips, illustrating its basic techniques with a simple example.

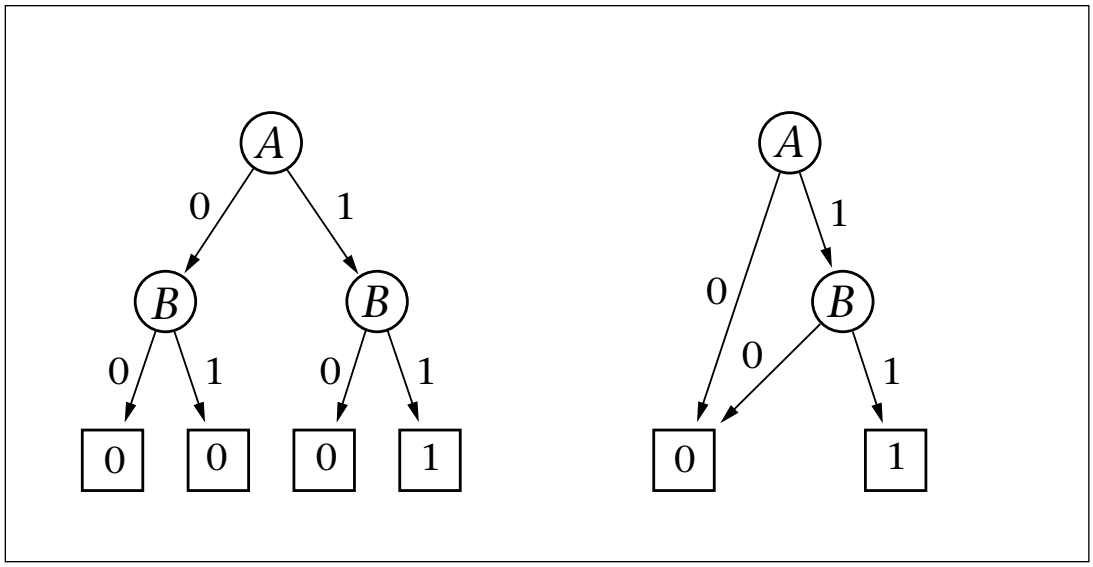

Figure 1. Two Equivalent Binary Decision Diagrams, a Nonreduced One and a Reduced One.

The 1 sink can only be reached by following the edges labeled 1 from *A* and *B*; thus, the represented Boolean function $\psi(A, B)$ evaluates to true if and only if *A* and *B* are true.

Binary Decision Diagrams: Why and for What?

MIPS is based on satisfiability checking, which is indeed not a new idea. However, MIPS was the first SAT-based planning system to make use of binary decision diagrams to avoid (or at least lessen) the costs associated with the exponential blowup of the Boolean formulas involved as problem sizes get bigger. Since the early days of MIPS, other planning systems based on BDDs have emerged, most notably Fourman's PROPPLAN and Störr's BDDPLAN. We believe that the key advantage of MIPS compared to those systems lies in its preprocessing algorithms.

Thus, it looks like BDDs are currently considered an interesting topic in AI planning. Why is that? There is no doubt about the usefulness of this data structure. Nowadays, BDDs are a fundamental tool in various research areas, such as model checking and the synthesis and verification of hardware circuits. In AI planning, they are mainly useful because of their ability to efficiently represent huge sets of states commonly encountered in state-space search.

Without going into too much detail, a *BDD* is a data structure for efficiently representing Boolean functions, mapping bit strings of a fixed length to either *true* or *false*. A BDD is a directed acyclic graph with a single root node and two sinks, labeled 1 and 0, respectively. For evaluating the represented function for a given input, a path is traced from the root node to one of the sinks, quite similar to the way decision trees are used. What distinguishes BDDs from decision trees is the use of certain reductions, detecting unnecessary variable tests and isomorphisms in subgraphs, leading to a unique representation that is polynomial in the length of the bit strings for many interesting functions. Figure 1 provides an example.

Among the operations supported by current BDD packages are all usual Boolean connectors such as *and* and *or* as well as constant time satisfiability and equality checking. MIPS uses the BUDDY package by Jørn Lind-Nielsen, which we considered particularly useful for our purposes because of its ability to form groups of several Boolean variables to easily encode finite-domain integers.

In MIPS, BDDs are used for two purposes: (1) representing sets of states and (2) representing state transitions.

Binary Decision Diagrams for Representing Sets of States

Given a fixed-length binary code for the state space of a planning problem, BDDs can be used to represent the characteristic function of a set of states (which evaluates to true for a given bit string, that is, state, if and only if it is a member of this set). The characteristic function can be identified with the set itself.

Unfortunately, there are many different pos-

sibilities to come up with an encoding of states in a planning problem, and the more obvious ones seem to waste a lot of space, which often leads to bad performance of BDD algorithms. It seems worthwhile to spend some effort on finding a "good" encoding, which is where the preprocessing of MIPS enters the stage.

Let us consider a very simple example of a planning problem where a truck is supposed to deliver a package from Los Angeles to San Francisco. The initial situation in PDDL notation is given by (PACKAGE package), (TRUCK truck), (LOCATION los-angeles), (LOCATION san-francisco), (AT package los-angeles), and (AT truck los-angeles). Goal states have to satisfy the condition (AT package san-francisco). The domain provides three action schemata named LOAD to load a truck with a certain package at a certain location, the inverse operation UNLOAD, and DRIVE to move a truck from one location to another.

The first preprocessing step of MIPS will detect that only the AT (denoting the presence of a given truck or package at a certain location) and IN predicates (denoting that a package is loaded in a certain truck) are fluents and, thus, need to be encoded. The labeling predicates PACKAGE, TRUCK, and LOCATION are not affected by any operator and, thus, do not need to be specified in a state encoding.

In a next step, some mutual exclusion constraints are discovered. In our case, we detect that a given object will always be at or in at most one other object, so propositions such as (AT package los-angeles) and (IN package truck) are mutually exclusive.

This result is complemented by what we call fact-space exploration: Ignoring negative (delete) effects of operators, we exhaustively enumerate all propositions that can be satisfied by any legal sequence of actions applied to the initial state, thus ruling out illegal propositions such as (IN los-angeles package), (AT package package), or (IN truck san-francisco).

Now all the information that is needed to devise an efficient state-encoding schema for this particular problem is at the planner's hands. MIPS discovers that three Boolean variables *A, B,* and *C* are needed. The first one is required for encoding the current city of the truck, where *A* is set if (AT truck san-francisco) holds true, and *A* is cleared otherwise, that is, if (AT truck los-angeles) holds true. The other two variables *B* and C encode the status of the package: Both are cleared if it is at Los Angeles, *C* but not *B* is set if it is at San Francisco, and *B* but not *C* is set if it is inside the truck.

We can now rephrase initial state and goal test as Boolean formulas, which can, in turn, be represented as BDDs: $\neg A \wedge \neg B \wedge \neg C$ denotes the initial situation, and the goal is reached in every state where $\neg B \wedge C$ holds true. The corresponding BDDs are illustrated in figure 2.

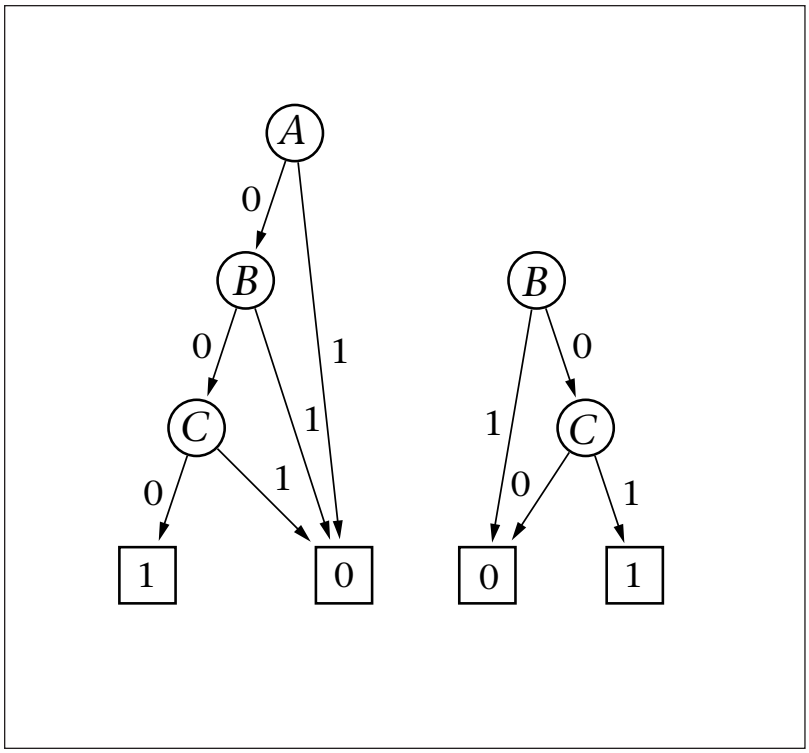

Figure 2. Binary Decision Diagrams for the Characteristic Functions of the Initial State, init(A, B, C) = $\neg A \wedge \neg B \wedge \neg C$, and of Goal States, goal(A, B, C) = $\neg B \wedge C$.

Binary Decision Diagrams for Representing State Transitions

What have we achieved so far? We were able to reformulate the initial and final situations as BDDs. As an end in itself, this representation does not help too much. We are interested in a sequence of actions (or transitions) that transforms an initial state into one that satisfies the goal condition.

Transitions are formalized as relations, that is, as sets of tuples of predecessor and successor states or, alternatively, as the characteristic function of such sets, Boolean formulas using variables *A, B, C* for the old situation and *A′, B′, C′* for the new situation. For example, the action (DRIVE truck los-angeles san-francisco), which is applicable if and only if the truck currently is in Los Angeles and has as its effect a change of location of the truck, not altering the status of the package, can be formalized using the Boolean formula $\neg A \wedge A' \wedge (B \leftrightarrow B') \wedge (C \leftrightarrow C')$.

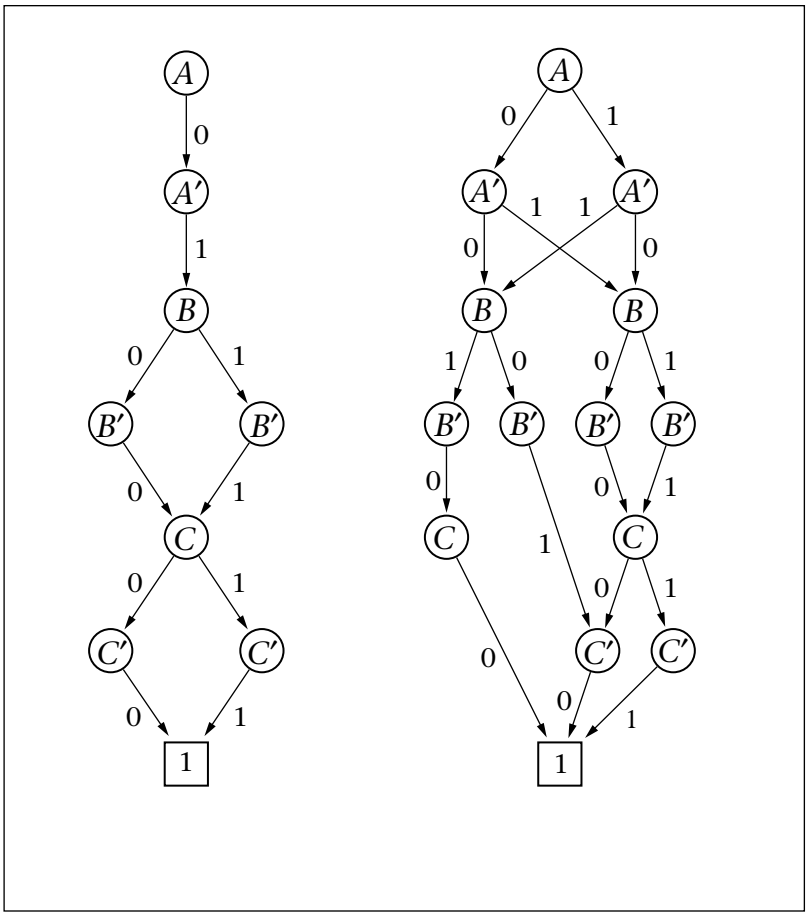

Figure 3. The Left Binary Decision Diagram Represents the Single Action (DRIVE truck los-angeles san-francisco), the Right One the Disjunctive of All Possible Actions and, Thus, the Complete Transition Relation.

The 0 sink and edges leading to it have been omitted for aesthetic reasons.

By conjoining this formula with any formula describing a set of states using variables *A, B,* and *C* introduced before and querying the BDD engine for the possible instantiations of (*A′, B′, C′*), we can calculate all states that can be reached by driving the truck to San Francisco in some state from the input set. This query for reachable states, put shortly, is the relational product operator that is used at the core of MIPS to calculate a set of successor states from a set of predecessor states and a transition relation. Of course, we have more than one action at our disposal (otherwise, planning would not be all that interesting), so rather than using the transition formula denoted earlier, we build one such formula for each feasible action (adding a no-op action for technical reasons) and calculate the disjunction of those, illustrated in figure 3.

Calculating the relational product for the operators in our example, starting from the set containing only the initial state, we get a set of three states—(1) the initial state, (2) one state where the truck has moved, and (3) one where the package was picked up—represented by a BDD with three internal nodes. Repeating this process, this time starting from the state set just calculated, we get a set of four states represented by a BDD with a single internal node, and a third iteration finally yields a state where the goal has been reached (figure 4). The goal condition can be tested by building the conjunction of the current state set and goal state BDDs and testing for satisfiability.

By keeping the intermediary BDDs, a legal sequence of states linking the initial state to a goal state can then easily be extracted, which, in turn, can be used to find a corresponding sequence of actions.

Evaluation of the MIPS Algorithm

It is not hard to see that given enough computational and memory resources, MIPS will find a correct plan if one exists. As it performs a breadth-first search in the state space, the first solution found will consist of a minimal number of steps. The algorithm will also terminate on unsolvable problem instances; the breadth-first search will eventually reach a fix point, which can easily be detected by comparing the successor BDD to the predecessor BDD after calculating the relational product. Thus, the algorithm is complete and optimal.

However, it is not blindingly fast, so various efforts were made to speed it up, mostly well-known standard techniques in symbolic search such as forward-set simplification. A bigger gain in efficiency was achieved by using bidirectional search, which can be incorporated into the algorithm in a straightforward fashion. One problem that arises in this context is that in some planning domains, backward iterations are far more expensive than forward iterations, and it is not trivial to decide when to perform which. We tried three different metrics to decide on the direction of the next exploration step: (1) BDD size, (2) number of states encoded, and (3) time spent on the last exploration step in this direction. In our experiments, the last metric turned out to be most effective.

Outlook

As for the basic exploration algorithm, big improvements leading to a dramatically better performance are not to be expected for the near future, with the possible exception of transition function splitting, which still needs to be incorporated into the system.

From the algorithmic repertoire of MIPS, the heuristic symbolic search engine, which until now has produced promising results but is still lacking in some domains, is getting the most attention at the moment (Edelkamp 2001). It might also be worthwhile to investigate the issue of optimal parallel plans, building on the work done by Haslum and Geffner (2000) for HSP.

Another research aim is the development of precomputed informative and admissible estimates for explicit and symbolic search based on heuristic pattern databases.

The single most important area of interest, however, is certainly the extension of MIPS to more general flavors of planning, such as conformant or strong cyclic planning where the strengths of symbolic methods are much more apparent than in the classical scenario (Cimatti and Roveri 1999; Daniele, Traverso, and Vardi 1999).

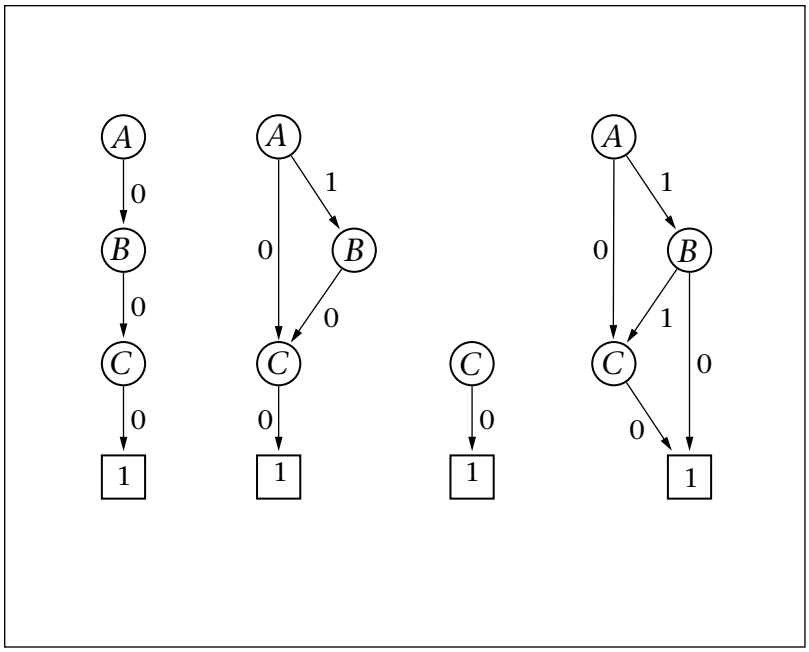

Figure 4. Binary Decision Diagrams (BDDs) Representing the Set of Reachable States after Zero, One, Two, and Three Iterations of the Exploration Algorithm.

Note that the size (number of internal nodes) of a BDD does not necessarily grow with the number of states represented. Edges leading to the 0 sink have been omitted for aesthetic reasons.

Acknowledgment

We thank F. Reffel for his cooperation concerning this research. S. Edelkamp's work is supported by DFG in a project entitled "Heuristic Search and Its Application to Protocol Validation."

References

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

Cimatti, A., and Roveri, M. 1999. Conformant Planning via Model Checking. In *Recent Advances in AI Planning, Volume 1809,* eds. M. Fox and S. Biundo, 21–34. Lecture Notes in Artificial Intelligence. New York: Springer-Verlag.

Daniele, M.; Traverso, P.; and Vardi, M. Y. 1999. Strong Cyclic Planning Revisited. In *Recent Advances in AI Planning, Volume 1809,* eds. M. Fox and S. Biundo, 35–40. Lecture Notes in Artificial Intelligence. New York: Springer-Verlag.

Edelkamp, S. 2001. Directed Symbolic Exploration and Its Application to AI Planning. Paper presented at the Spring Symposium on Model-Based Validation of Intelligence, 26–28 March, Stanford, California.

Edelkamp, S., and Helmert, M. 2000. On the Implementation of MIPS. Paper presented at the Fifth International Conference on Artificial Intelligence Planning and Scheduling, Workshop on Model-Theoretic Approaches to Planning, 14 April, Breckenridge, Colorado.

Edelkamp, S., and Helmert, M. 1999. Exhibiting Knowledge in Planning Problems to Minimize State Encoding Length. In *Recent Advances in AI Planning, Volume 1809,* eds. M. Fox and S. Biundo, 135–147. Lecture Notes in Artificial Intelligence. New York: Springer-Verlag.

Haslum, P., and Geffner, H. 2000. Admissible Heuristics for Optimal Planning. In Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling, 140–149. Menlo Park, Calif.: American Association for Artificial Intelligence.

Stefan Edelkamp is a research assistant for the Algorithms and Data Structures Group at the Institute for Computer Science at the University of Freiburg. He studied computer science and mathematics in Dortmund and Dublin and received his Ph.D. in computer science from the University of Freiburg on the subject of data structures and learning algorithms in state space search. His current research interests include sequential sorting, heuristic search, AI planning, model checking, and graph algorithms. His e-mail address is edelkamp@informatik.uni-freiburg.de.

Malte Helmert is currently working on his Ph.D. thesis with the Artificial Intelligence Group of the Institute for Computer Science at the University of Freiburg. He studied computer science in Freiburg and Durham. His current research interests include AI planning, complexity theory, games, and computational geometry. His e-mail address is helmert@informatik.uni-freiburg.de.

Call for Papers and Participation

Eighteenth National Conference on Artificial Intelligence

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

Sponsored by the American Association for Artificial Intelligence

AAAI-02 is the Eighteenth National Conference on Artificial Intelligence (AI). The purpose of this conference is to promote research in AI and scientific interchange among AI researchers, practitioners, and scientists and engineers in related disciplines.

The conference provides a forum for a broad range of topics, including (but not limited to) knowledge representation, machine learning, autonomous agents, planning, robotics and machine perception, expert systems, theorem proving, commonsense reasoning, probabilistic inference, constraint satisfaction, game playing, automated diagnosis, data mining, natural language processing, neural networks, reinforcement learning, and cognitive modeling.

As the national conference for all of AI, AAAI-02 is intended to play a centralizing and unifying role that complements more specialized conferences. To encourage papers that speak to the whole AI community, this year we will avoid multiple parallel sessions. Instead, all accepted papers will be presented as posters, and a subset will also be presented orally in a single plenary track. All papers will appear identically in the proceedings, and all will be subject to the same rigorous review process.

Timetable for Authors

- *January 21, 2002:* Electronic submission of title page, abstract and paper
- *January 22, 2002:* Submission of two (2) paper copies to AAAI office
- *March 15, 2002:* Notification of acceptance or rejection
- *April 9, 2002:* Camera-ready copy due at AAAI office

Electronic Title Page

Because of the blind reviewing process, we require that authors who plan to submit a full paper submit a separate electronic title page by the electronic paper submission deadline, January 21, 2002. This is done by using a web browser to fill out the form linked to the Authors Web site:

- www.aaai.org/Conferences/National/2002/Author/titlepage.html

The form will include the abstract, keywords, and other information as indicated. After submitting the form, the page will display a unique tracking number. Authors should print two (2) hard copies of this electronic title page (unstapled) and submit them with the hard copy of their papers as described below. Information for authors without access to a forms-capable browser will appear on the web site.

Paper Submission

Electronic paper submission is required. Instructions about how to do this will be available at the URL above. Authors should also submit two (2) hard copies of the paper. If an author has been excused by the program cochairs from submitting the electronic version of a paper, the author is required to submit six (6) hard copies of the paper. Both hard copy and electronic submissions may be no longer than 6 pages long including references, and formatted in AAAI two-column camera ready style. We cannot accept submissions by e-mail or fax. Please send papers to:

AAAI-02
American Association for Artificial Intelligence
445 Burgess Drive
Menlo Park, CA 94025-3442
Telephone: 650-328-3123

Reviewing for AAAI-02 will be blind to the identities of the authors. Details on formatting and preparing the paper for blind review can be found at the authors' web site noted above.

Submissions to Other Conferences or Journals

Papers submitted to this conference must not have been accepted for publication elsewhere or be under review for another AI conference. However, to encourage interdisciplinary contributions, we may consider work that has been submitted or presented in part to a forum outside of AI. The guidelines of the AAAI policy on multiple submissions are fully detailed at the authors' web site noted above and must be carefully followed.

Review Process

Program committee members will identify papers they are qualified to review based on the information electronically submitted (the paper's title, content areas, and abstract). Their reviewing will be done blind to the identities of the authors and their institutions. The program committee's reviews will make recommendations to the senior program committee, which in turn will make recommendations to the program cochairs. Although the program cochairs will formally make all final decisions, in practice almost all will be made earlier in the process.

Publication

Accepted papers will be allocated six (6) pages in the conference proceedings. Up to two (2) additional pages may be used at a cost to the authors of $275 per page. Papers exceeding eight (8) pages and those violating the instructions to authors will not be included in the proceedings. *Authors will be required to transfer copyright of their paper to AAAI.*

Questions and Suggestions

Concerning author instructions and conference registration, write to: ncai@aaai.org.
Concerning suggestions for the conference and other inquiries, write to the Program Cochairs:

Rina Dechter, University of California, Irvine (dechter@ics.uci.edu)

Richard S. Sutton, AT&T Shannon Laboratory (sutton@research.att.com)