USING SATISFIABILITY IN TEMPORAL PLANNING

KAMAL-ALDIN GHIATHI
Computer Science Department
Sharif University of Technology
IRAN
ghiathi@mehr.sharif.edu
http://ce.sharif.edu/~ghiathi

GHOLAM-REZA GHASEM-SANI
Computer Science Department
Sharif University of Technology
IRAN
sani@sharif.edu
http://sharif.edu/~sani

Abstract: Most traditional planners ignore the notion of time. Of those that have addressed this issue, Allen's temporal planning is the most expressive approach so far. However, due to its high complexity, it is useless for tackling real world problems. On the other hand, new highly efficient planning methods based on satisfiability, do not still address the time appropriately. In this paper, using the idea of satisfiability planning, a new method of temporal planning is introduced, which outperforms the Allen's approach.

Key-Words: temporal planning, satisfiability planning, interval algebra, point algebra

1 Introduction

Before introducing the notion of satisfiability planning by Kautz and Selman [10], the planning was seen as a deductive process. As a side effect of this view, planners were usually lifted and solved problems in first order logic. In last decade, two new planners, SatPlan[10] and GraphPlan[2], returned back to propositional planning and gained a great speed-up. Since then, some planners have been constructed that combined the ideas of these planners with different aspects of planning. SatPlan, which is discussed in section 2, is the pioneer in this regard.

An important aspect that humans consider in real world planning is time. But automatic planners usually rely on simplifying assumptions about the time. For instance it is often assumed that actions are instantaneous. As a result of this assumption, temporal relations between actions are reduced to just "before", "after", or "synchronous". But, in the real world, actions are not instantaneous and may have complex relations (e.g. consider relations between micro-instructions in a microprocessor).

Some old planners tried to incorporate the notion of time in planning [1,16]. Although considering the time increases the expressiveness of planners, but not all planners have exploited all of this expressiveness. In some planners time is expressed numerically[17] and others allow it to be expressed solely as relations between actions and propositions[1]. On the other hand, some planners work with discrete time corresponding to natural numbers[10], while others work with real numbers[15]. In some planning systems, duration of actions is fixed but some allow variable length actions. Allen has introduced the most expressive method so far in a planner called TIMELOGIC[1]. Using a model for planning based on a temporal logic, Allen introduced a way of incorporating time in planning. However, the main problem with his method is its high complexity, which renders it useless in tackling real problems. This model is discussed in section 3.

There exist much similarity between TIMELOGIC and SatPlan. For instance both need more information than what can be expressed in the STRIPS[7] format. In this paper, we introduce a new method of temporal planning that solves planning problems using constraint satisfaction method. The method has been implemented in a planner we named SATEMP[1]. SATEMP selects a combination of actions and, after a series of simple tests, if it is determined to have the chance of being a solution, translates it to an equivalent CNF[2] formula. If this CNF formula is satisfiable, then the combination of actions would be a solution to the problem; and one can obtain the solution to the planning problem by using values of variables in the satisfied CNF formula. SATEMP is discussed in section 4.

There are different ways of translating a planning problem to the corresponding CNF formula. For instance, one can translate the planning problem into a CNF formula based on *interval algebra*[1] or *point algebra*[5]. These methods have different efficiencies. In section 5 different methods of encoding planning problems into CNF formulas and their efficiency are discussed. Section 6 contains an example of SATEMP Planning.

[1] SAtisfiability TEMporal Planner.
[2] Conjunctive Normal Form.

2 Overview of SatPlan

Planning problems can be considered as constraint satisfaction problems [15]. Furthermore, satisfiability as a special representation of constraint satisfaction problems has drawn attention of many researchers [9]. In 1992, in order to show effectiveness of a satisfiability tester called GSAT[9], Kautz introduced the notion of satisfiability planning and expanded the domain of solvable planning problems. This success has continued till now and still the fastest planners use satisfiability test in at least one part of their problem solving process [11]. Part of this success is due to excessive work done on the SAT problem itself. In this section, we give a brief review of SatPlan.

SatPlan gets as input a planning problem (in propositional form) and makes an equivalent CNF formula (assuming the solution size is not greater than a predetermined limit). By equivalent CNF, we mean that if the CNF is satisfiable then the planning problem is solvable; and if the CNF is unsatisfiable then the planning problem is not solvable. Furthermore, it is possible to find the solution of the planning problem in polynomial time, given the solution of the CNF, and vise versa. The above statement needs a bit of elaboration. SatPlan does this transformation assuming that the size of the solution plan is known. To solve a problem, it first assumes that a zero-size solution exists, and does the above transformation. If the corresponding CNF is not satisfiable, it then assumes that the size of solution is one, and again tries to find the solution. This process is repeated until either the solution is found or a predetermined limit on the number of iterations is reached.

There are many different methods of encoding planning problems as CNF formulas[6]. Here we explain a simple method that was used in [10]. Assume the size of the solution of a planning problem is d. Then there are d actions with a total ordering, and we can bind each action to its execution step. Similarly, at each time step some propositions are true and some are false. Each variable in the CNF formula corresponds to the execution of an action or truth of a proposition in one of the d time steps. Now that CNF formula's variables are known, one can translate these constraints into a CNF formula:

1. If action p is done at time step t, then its preconditions are true at time step t-1 and its effects are true at time step t.
2. Exactly one action takes place at each time step.
3. Initial conditions are true at time step zero.
4. Goal conditions are true at time step d.
5. If a proposition is true at time t-1 and the action performed at time t does not delete it, it continues to be true at time step t.
6. Domain constraints. For instance in blocks world, propositions "on(a,b)" and "on(a,c)" can't be true at the same time step.

These constraints can be expressed in propositional form. For example, to state that at each step at least one action takes place, we can write:

$$(a_1(1) \vee a_2(1) \vee \ldots \vee a_n(1)) \wedge (a_1(2) \vee a_2(2) \vee \ldots \vee a_n(2)) \wedge \ldots \wedge (a_1(d) \vee a_2(d) \vee \ldots \vee a_n(d))$$

Where n is the number of actions, and each $a_i(j)$ is a variable.

After constructing the CNF formula, it is given to a satisfiability checker program such as GSAT[10], WalkSat [14], or Tableau [4] to determine whether it is satisfiable or not; and, if it is, which propositions are true and which are false. Having determined the solution of a CNF formula, the solution to the corresponding planning problem can easily be found by listing actions that are true at each time step.

3 Temporal world model of Allen

Allen has introduced a model of planning that is solely based on the notion of time. In his model, each action and each proposition is associated with a time interval. Each action (proposition) takes place (is true) at its corresponding time interval. Table 1 shows 13 possible relations between any two intervals.

Relation	Symbol	Symbol for inverse	Pictorial example
X before Y	<	>	XXX YYY
X equal Y	=	=	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYYY

Table 1: The thirteen possible relations

In this method, when defining an operator, the relations between intervals of propositions and that of the operator itself must be defined. For example,

consider operators MOVE and MOVTBL in the blocks world domain:

MOVE (x,y,z)
Preconditions: CLEAR(x), CLEAR(z), ON(x,y)
Effects: ON(x,z), CLEAR(y), ¬CLEAR(z) , ¬ON(x,y)

MOVTBL[1] (x,y)
Preconditions: CLEAR(x), ON(x,y)
Effects: ON(x,table) , CLEAR(y), ¬ON(x,y)

In the temporal world model these operators are expressed as:

If MOVE (x,y,z) occurs over time Sxyz then
CLEAR(x) holds over time Cx1,
and Cx1 *meets* Sxyz, and
CLEAR(z) holds over time Cz ,
and Cz *finishes* Sxyz, and
ON(x,y) holds over time Oxy,
and Oxy *(o s d)* Sxyz, and
CLEAR(y) holds over time Cy ,
and Sxyz *(o fi di)* Cy, and
CLEAR(x) holds over time Cx2,
and Sxyz *meets* Cx2, and
ON(x,z) holds over time Oxz,
and Sxyz *overlaps* Oxz.

If MOVTBL (x,y) occurs over time Sxyt then
CLEAR(x) holds over time Cx1,
and Cx1 *meets* Sxyt, and
ON(x,y) holds over time Oxy,
and Oxy *(o s d)* Sxyt, and
CLEAR(y) holds over time Cy,
and Sxyt *(o fi di)* Cy, and
CLEAR(x) holds over time Cx2,
and Sxyt *meets* Cx2, and
ON(x,table) holds over time Oxt,
and Sxyt *overlaps* Oxt.

Let's define some terms. C*ollapsing* two intervals means to limit the relation between them to equal. If an interval of a goal or action's precondition is not yet collapsed with an interval of an initial state proposition or action's effect, then a *causal gap* exists, and that goal (or precondition) has no *causal explanation (i.e. it is unexplained)*.
Now we can explain Allen's planning process. There are two special intervals: I (initial state) and G (goal state), such that initial state intervals (intervals associated with initial state propositions) *contain*[2] I, and goal state intervals (intervals associated with goal state propositions) *contain* G. The problem solver continually repeats the process of finding causal gaps and eliminating them with new proposed actions or collapsing two intervals. As a result of causal gap elimination process, possible relations between the two intervals involved in the causal gap are restricted to equal. This can propagate and constrain other relations. If in some step, the set of possible relations between two intervals becomes empty, system backtracks. When no causal gap is left, the problem is regarded as solved and the process stops.
In propagating constraints, two other types of constraints other than interval algebra's constraints, are used: the proposition and domain constraints.

Proposition constraint:
Two intervals associated with the same proposition are either equal, or one is strictly before the other. For instance, if both intervals I1 and I2 are associated with proposition P, then I1(< = >) I2.

Domain constraint:
In each domain, some constraints exist that are not used in deduction. These constraints are a consequence of initial state and actions' definitions[2]. For example, in the blocks world, if we start with an initial state that no block is both CLEAR and have some block on it, then we can't reach a state in which CLEAR(a) and ON(b,a) hold at the same time. This additional information is not necessary when we use a deductive approach. But it is necessary in the satisfiability (constraint satisfaction) approach.

It is interesting that although TIMELOGIC solves the planning problem by using deduction, since it uses constraint satisfaction, it also needs domain constraints.

4 Temporal planning as satisfiability

In this section, a new planning method, which aggregates some aspects of SatPlan and TIMELOGIC is introduced. From previous discussion, we can see the following similarities between SatPlan and TIMELOGIC:

1. Both define actions as a conditional proposition, where the execution of an action implies the truth of its preconditions and effects, respectively, before and after the execution.
2. Both do not discriminate between preconditions

[1] Move to table
[2] Interval A *contains* interval B iff A(di si fi)B.

and effects. Albeit, in TIMELOGIC when adding a new action to the plan, preconditions are assumed to be unexplained, while effects are assumed to be explained.
3. Both need more information than what is given in the STRIPS format. For example, SatPlan needs a complete initial state (i.e. no proposition is allowed to be unknown) and both need domain constraints.

On the other hand, the following differences between these systems are noticeable:

1. TIMELOGIC solves the planning problem in first order logic, whereas SatPlan does this in propositional logic.
2. In TIMELOGIC, planning problem is solved using deduction, but in SatPlan it is solved using satisfiability.

Here, we introduce a new planner called SATEMP, which is similar to TIMELOGIC from the possible temporal relations point of view, but it solves the planning problem using satisfiability. The main goal is to improve the efficiency of temporal planning process while preserving its expressiveness of TIMELOGIC.
To find a solution, we first perform a breadth first search, i.e, first a zero-length solution is searched for, and if failed the length of solution is assumed to be two, etc. This process is continued until either a solution is found or the length of the plan being searched for exceeds a predetermined limit. For each fixed length of solution, all possible combinations (order is not important) of actions are generated. Then for each combination of actions a CNF formula is generated. Each CNF formula is constructed using the following constraints:

1. Interval I is before or meets all other intervals; and interval G is after or is met by all other intervals.
2. Interval I meets the intervals of initial state propositions and intervals of goals state propositions meet G.
3. Constraints between actions' intervals given in the actions' definitions.
4. Domain constraints.
5. Propositional constraints.
6. Goal state intervals and intervals associated with preconditions of actions must have causal explanation.
7. Every two propositions have at least one of 13 possible relations.
8. Every two propositions have at most one of 13 possible relations.
9. Propagating constraints (by interval algebra), e.g. if a<b and b<c we infer a<c.

It is necessary to elaborate on item 8. This item ensures that a linear plan (total order plan) is produced. This is in contrast with TIMELOGIC in which a partial order plan is generated. Allen has claimed that after constraint propagation if the set of possible relations between no two intervals is empty, then at least one linear plan exists. Even if this claim is true, partial order plan generated by TIMELOGIC is not worthy enough, because the solution is general with respect to the specific actions used in generating the plan. This means that in a real problem with some numerical constraints on the length of intervals, it is possible that while a solution exists, the solution produced by TIMELOGIC would be unacceptable.
After generating a CNF formula, tableau[1] is used to check its satisfiability. Before making a CNF formula, some preliminary tests are performed, the most important of which is assuring that for each causal gap at least one explanation exists. This test is very fast and avoids construction of CNF formulas except in a few cases.

5 A comparison among different encoding methods

There are many different methods to transform a planning problem to a CNF formula. Each of these is called an encoding method. Counting the number of clauses generated with each of constraints mentioned in section 4, one can verify that, here, the dominant factor is the size of the CNF formula and therefore the main source of hardness of the problem, is the 9th constraint[2].
The number of "propagating constraints" is I^3R^2, where I is the number of intervals in the proposed solution (the combination of actions translated to CNF), and R is the number of possible relations between intervals. In the implementation of SATEMP based on interval algebra[3] R is 13. Since each constraint is a conditional proposition with a disjunctive consequent part and

$$a \wedge b \rightarrow c_1 \vee \ldots \vee c_k \Leftrightarrow \neg a \vee \neg b \vee c_1 \vee \ldots \vee c_k ,$$

[1] To overcome the huge size of our CNF formulas, we changed tableau a bit.
[2] If all clauses have only two literals, the problem is solvable in polynomial time[3]. So it is reasonable to say that clauses with 3 or more literals are the main source of the hardness of the problem.
[3] SATEMP is implemented both based on interval algebra and improved natural encodings.

each propagating constraint is translated into a clause. As it was said, these constraints have the most influence on the hardness and size of the CNF formula, thus we base comparison among different encodings on the number of propagating constraints and the number of literals in each clause.

5.1 TIMELOGIC encoding

In this encoding R is 13. Renaming I to n, we have $169n^3$ propagating constraints. We can omit those clauses that don't expose any restriction, i.e. those that restrict the relations between two intervals to all 13 possible relations. By doing so, propagating constraints are reduced to $166n^3$. In this method, number of literals in each clause can be as many as 14 (2 for the antecedent part and 12 for the consequence part of the propagating constraint).

5.2 Natural encoding

Humans usually work with only three relations < = >. If we break n intervals into their corresponding endpoints, we will have 2n number of endpoints. A few other constraints are needed to ensure that left endpoints are before right endpoints (these new constraints are very useful, since they constitute clauses with only one variable, which are very useful in fast simplification of the CNF formula). Using this method, number of clauses is reduced to $3^2(2n)^3$ or $76n^3$. If we omit non-restricting clauses (those with a consequent part stating that the two endpoints can have all possible 3 relations), number of clauses is reduced to $7(2n)^3$ or $56n^3$. Another advantage of this encoding is the reduced number of literals in each clause. In TIMELOGIC encoding, we had clauses with 14 literals. Using new encoding, number of literals in each clause is reduced to 3 (2 for 2 antecedents of "if aR1b and bR2c then …" constraint and 1 for the single possible relation between a and c). This reduction in the number of literals in each clause is very important. Because although no solution for SAT problem, better than the obvious $O(2^n)$ where n is number of variables in CNF formula, is found[8], for 3-SAT solutions with better time do exist[13]. Also, there are evidences indicating that if P≠NP, then k-SAT (SAT problem with no clauses having more than k literals) is easier for smaller k [8].

5.3 Improved natural encoding

It is possible to express < = > relations with a single ≤ relation. For instance, a=b can be expressed as ($a \le b \wedge b \le a$). This leads to a reduction of the number of propagating constraints down to $(2n)^3$ or $8n^3$. Another benefit of this encoding is the augmentation of unit clauses, since = is now represented as a conjunction of ≤ relations. The CNF formulas produced by this encoding are at least 20 times smaller than those produced by TIMELOGIC encoding. Since check of satisfiability of a CNF formula is exponential in size of CNF formula, reduction of the size of CNF formula, results in an exponential increase of speed.

6 An example

In this section we illustrate, as an example, that how SATEMP solves the Sussman anomaly. Sussman anomaly is depicted in Figure 1. The version of SATEMP based on TIMELOGIC encoding, solves this problem in 35 seconds on a 950MHz computer with 256MB memory. Three CNF formulas are made before finding the solution and each CNF formula consists of 7.8 million clauses. The version of SATEMP based on improved natural encoding, solves this problem in 3 seconds on the same computer. It makes only one CNF formula (which correspond to the solution) and the CNF formula consists of 390,000 clauses. Figure 2 shows the solution generated by SATEMP. The source of SATEMP is available from http://ce.sharif.edu/~ghiathi/SATEMP.

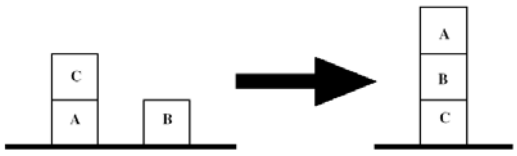

Figure 1: Sussman anomaly problem.

7 Conclusion

In this paper a new planner has been introduced that aggregates efficiency of SatPlan and expressiveness of TIMELOGIC. Although a practical comparison between SATEMP and TIMELOGIC was not possible for the lack of access to the source of TIMELOGIC, but in section 5 it was explained why SATEMP is much faster. Main reasons for improvement of SATEMP over TIMELOGIC are:

1. SATEMP can use heuristics like fail first principle and solve the problem very fast. It is so, because SATEMP first constructs a CNF formula in which all constraints are at hand. In fact, efficiency of tableau is a result of this sort of heuristics.
2. By transforming the planning problem into a

CNF formula, it is possible to use encodings that are 20 times smaller than encoding based on the interval algebra. Note that a linear reduction in size of CNF formula results in an exponential speed up. It was also shown that TIMELOGIC could be further improved if, instead of the interval algebra, the point algebra is used.

Although SATEMP has gained a great speed up compared to TIMELOGIC, but it is still slow and can't be used for the real world problems. In future we want to exchange some simplifying assumptions with an acceptable speed up. The idea is that preconditions and effects of the real life actions are usually before and after action interval and at most may have some overlap. But it is unusual for a precondition to last after the action interval, and for an effect to be seen before the action is executed. Using this assumption, we can rule out cyclic causal gap explanation that in our experiments was the main source of wasted time.

8 Acknowledgement

The authors would like to thank James Crawford for his publicly available code of tableau, and Bart Selman and Henry Kautz for that of GSAT.

References:

[1] James F. Allen and Johannes A. Koomen. Planning using a temporal world model. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence. 741-747. 1983.

[2] Avrim L. Blum and Merrick L. Furst. Fast Planning Through Graph Analysis. Artificial Intelligence, 90:281-300, 1997.

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms (second edition). MIT press, page 1003. 2001.

[4] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in satisfiability problems. Proc. AAAI-93, Washington, DC.1993.

[5] T. Dean and M. Boddy. Incremental causal reasoning. AAAI87. 196-201. 1987.

[6] Michael D. Ernst, Todd D. Millstein and Daniel S. Weld. Automatic SAT-Compilation of Planning Problems. In proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI- 97), Nagoya, Aichi, Japan, August 23-29, 1997.

[7] R. E. Fikes and N. J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3-4):189-208. 1971.

[8] Russel Impagliazzo and Ramamohan Paturi. Complexity of k-SAT. IEEE Conference on Computational Complexity 1999.

[9] J. Gu, P. W. Purdom, J. Franco, and B. W. Wah. Satisfiability Problem: Theory and Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, pp. 19-152, 1997.

[10] Henry Kautz and Bart Selman. Planning as Satisfiability. In proceedings of the 10th European Conference on Artificial Intelligence (ECAI 92), Vienna, Austria, August 1992.

[11] Henry Kautz and Bart Selman. Pushing the Envelope: Planning, Propositional Logic, And Stochastic Search. In proceedings of the 13th National Conference on Artificial Intelligence (AAAI- 96), Porland, OR, 1996.

[12] Henry Kautz and Bart Selman. Unifying sat-based and graph-based planning. In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99), pages 318–325. Morgan Kaufman, 1999.

[13] Robert Rodosek. A new approach to soling 3-satisfiability. In Proceedings of the 3rd International Conference on Artificial Intelligence and Symbolic Mathematical Computation, pages 197-212. Springer, LNCS 1138, 1996.

[14] Bart Selman, Henry Kautz and B. Cohen. Local search strategies for satisfiability testing. Dimacs Series in Discrete Mathematics and Theoretical Computer Science. 1996.

[15] David E. Smith and Daniel S. Weld. Temporal Planning With Mutual Exclusion Reasoning. IJCAI. 1999.

[16] Matk Stefik. Planning and Meta-Planning (MOLGEN: Part 2). Artificial Intelligence 16 141-170. 1981.

[17] Steven A. Vere. Planning in Time: Windows and Durations for Activities and Goals. IEEE. 1983.

[18] Daniel S. Weld. An Introduction to Least Commitment Planning. AI Magazine, Summer/ Fall 1994.

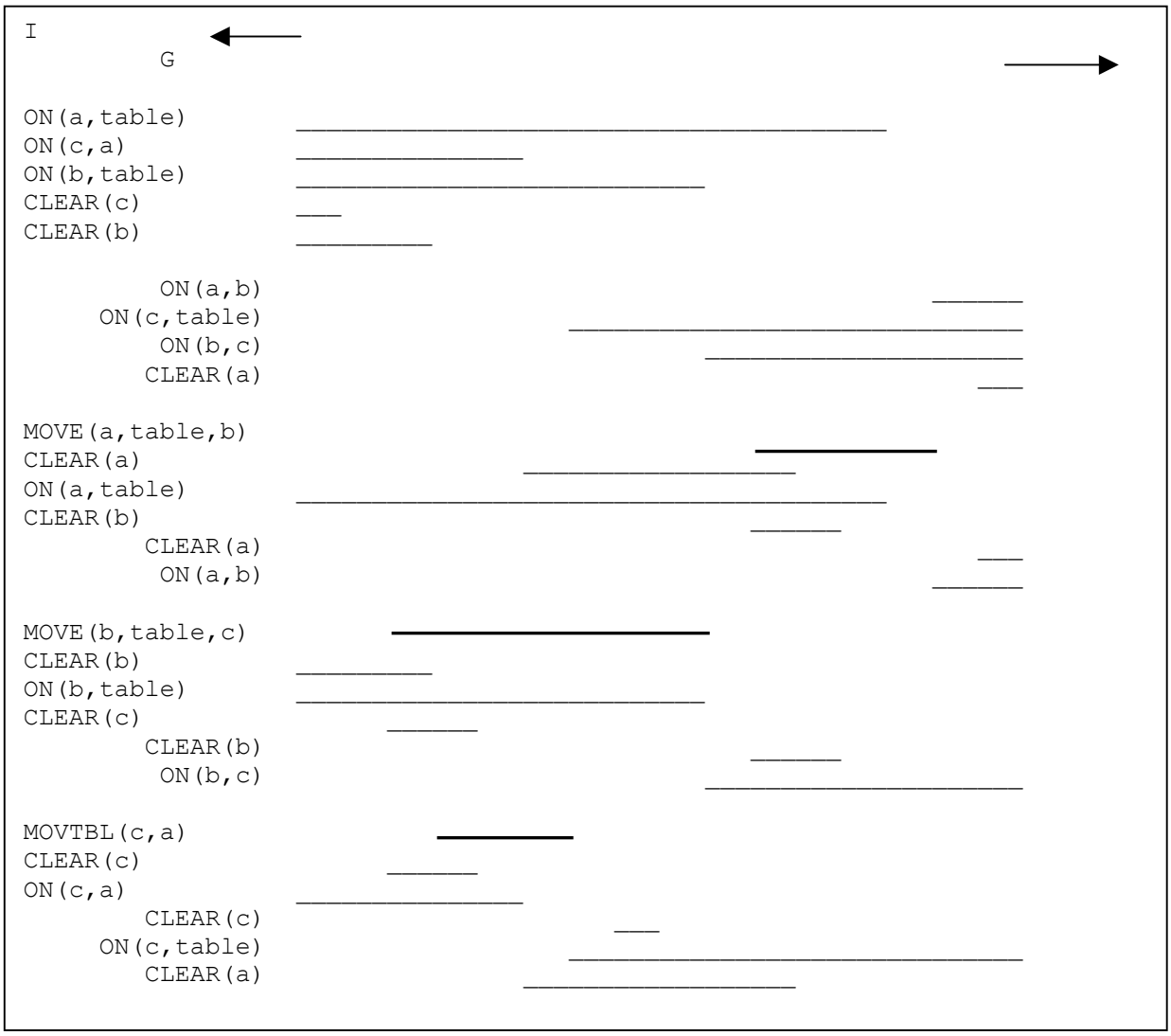

Figure 2: solution generated by SATEMP for the Sussman anomaly.