AIAA '87

AIAA-87-2526-CP

Air Traffic Controller Aids for Planning of Arrival Traffic: An AI Approach

Ronald L. Chrisley and Leonard Tobias

NASA Ames Research Center
Moffett Field, California

Guidance, Navigation and Control Conference
August 17-19, 1987/Monterey, California

AIR TRAFFIC CONTROLLER AIDS FOR PLANNING OF ARRIVAL TRAFFIC: AN AI APPROACH

Ron Chrisley and Leonard Tobias***
NASA Ames Research Center
Moffett Field, California.

ABSTRACT

This paper considers air traffic management as an AI planning problem, and develops a conceptual framework for planning in the Air Traffic Control (ATC) domain. In particular, if ATC is viewed as a planning problem, it becomes desirable to design the planner to be an efficient searcher. An Assumption-based Truth Maintenance System (ATMS) provides for efficient search in monotonic domains, i.e., in domains where the number of statements known to be true is strictly increasing over time, but there has been no discussion of the use of the ATMS in a nonmonotonic planning domain such as ATC. This paper examines a modification of the ATMS to include the nonmonotonicities inherent in the ATC domain, noting especially under what circumstances the advantages of the ATMS in standard problem-solving domains carry over to planning. The *noninteracting* actions of the conceptualization of the ATC domain as presented are contrasted with the (problematic) interacting actions of other domains. These planning concepts will be incorporated in an automation aid for enroute arrival controllers, under development at NASA Ames.

INTRODUCTION

An Artificial Intelligence (AI) approach to solving a realtime problem, such as ATC, may be preferred to a traditional, algorithmic approach for several reasons. First, it may be the case that the only way an acceptable solution can be generated in realtime in a complex domain is to employ a nondeterministic, or heuristic, search of the solution space. An exhaustive search of the possibilities would most certainly be impractical in any real-world domain. Also, it is arguable that a declarative programming approach to the problem (where elements of the program are not machine or algorithm instructions, but logical representations about the domain itself) is more desirable since it facilitates research and development, evaluation, and future modification of the system in a way that conventional algorithmic solutions cannot. In the case where the AI system is assisting in solving a problem that previously only a human expert could solve, the declarative approach facilitates knowledge engineering: the translation of human expertise into code that will enable the computer to perform similarly. For any sufficiently complex domain, the declarative approach seems to be the only practical alternative.

Another factor in designing ATC advisory aids in particular points toward an AI approach: the high importance of controller confidence in the ATC advisory aid. In any situation where a human controller and an advisory-generating system interact, the controller must be convinced that the system is generating realistic solutions to the ATC problems. This can be accommodated, at least in the early stages of the introduction of the system, by an explanation capability, with which a controller can ask for the reasoning behind a recommended advisory and evaluate its logic and sensitivity to the intricacies of the situation. Such a capability is much easier to develop for a system that already represents the ATC domain in terms of the relations, objects, and rules the controller understands and uses.

---

* Research Scientist

** Research Scientist, Member AIAA

ATC AS PLANNING

An earlier paper claimed that air traffic control (ATC) in the near-terminal area could be conceptualized as an AI planning problem.[1] That paper designs the ATC Advisor as a forward planner, translating each proposed solution into parameters for a procedurally attached 4D descent advisor algorithm. The 4D advisor is then invoked, providing the necessary data for the simulation of the proposed plan. The simulation is then evaluated for safety and efficiency. The results of this evaluation are used to order the set of proposed plans.

The necessary existence of the 4D trajectory advisor precludes an exclusively backward planning approach from goals to advisories, since there are no rules that have as consequents the facts derived by the 4D advisor. Furthermore, the goals in ATC are much less well-defined than the options available to a controller in any given situation. For a planner in the chess domain, it would be unreasonable to generate a plan by choosing a particular winning state and reason backwards until the current state is reached; this is also the case in the ATC domain. Unfortunately, much of the research in planning has eschewed forward planners, in favor of general, domain-independent backward planners. However, if a domain-specific planner is desired, then the generality of the backward approach will be of little use and may have serious limitations, since there are some problems which cannot be solved efficiently without domain knowledge. For example, one disadvantage of general backward planning is its inability to handle some planning situations that involve conjunctive goals. Sussman's Anomaly is an example of such a problem in the blocks world. Consider the initial situation of block **A** on block **B**, block **C** on the table, with the goal being to have **A** on **C** on **B**. A backward planner will see that it cannot attain the **C** on **B** goal immediately, so it will solve the **A** on **C** goal. Of course, this prevents it from ever attaining the **C** on **B** goal without undoing what has already been "accomplished." A forward planner, equipped with the right heuristics, could solve this problem more efficiently. This is not to say that a backward planner with heuristics could not solve it; but if you have to use domain knowledge, backward planning seems to have no special advantage over forward planning. In fact there might be reasons to believe that forward planning is better, if people find it easier to deal with forward heuristics.

However, the ability to handle Sussman's anomaly in particular, or conjunctive goals in general, cannot be a significant justification for forward planning in our current conceptualization of the ATC domain, since there are very few, if any instances where the effects of one action that is being undertaken to reach the goal remove the preconditions necessary for another action that is necessary to reach the goal. Specifically, the actions available to the ATC Advisor, when applied to one object, do not affect the preconditions required to apply any other action to any other object. This is an important particular fact about the ATC domain to which will be referred to below.

Although it may be that a general, backward approach to planning is the best, when it is possible, there are nevertheless domains which require forward planning, as does the ATC domain because of the nonlinear descent advisor algorithm. Very little has been said about which forward planning systems are the best and how they may be improved.

THE ATMS AND PLANNING

This section will examine how the ATMS may be used in forward planning. Since forward planning requires search, any attempts to develop an ATC advisory system as a forward planner must be sensitive to efficient control of knowledge-based search. The ATMS is an efficient means of searching the type of space a planner is confronted with. The ATMS, like the TMS, avoids futile backtracking, rediscovering contradictions and inferences, and the disadvantages of inefficient orderings of alternatives in normal problem-solving situations.[3,4] However, the ATMS also allows an explicit comparison of the various hypothetical

states, provide an efficient means of switching contexts from state to state, allow some valuable reasoning with inconsistency that the TMS prohibits, and is more efficient because it does not need to provide for dependency-directed backtracking.[4]

The ATMS does this by factoring the assumption space. All of the consequences of an assumption are saved and associated with that assumption. The set of sentences known to be true in a hypothetical world formed by assuming conjunctions of assumptions is then easily calculated: it is the union of the sentences known to be true in either of the "parent" worlds, the worlds corresponding to the individual assumptions. Thus, there is no rederivation of facts. Once a fact is derived, it is stored away with a label so that it can be easily determined whether the fact holds in a particular hypothetical world, or set of assumptions.

There are several potential uses for the ATMS in planning, but perhaps the most important aspect of this is how actions and their effects can be represented in an ATMS. Specifically, how can the ATMS handle the frame problem?

The method of application is trivial if the situation calculus[5] is used. In the situation calculus, states that result from applying planning operators are explicitly represented, and facts are conceptualized as holding in a particular state. But there are several drawbacks to this approach, an important one being the necessity of frame axioms or copy rules that explicitly inform the problem solver what facts do not change from state to state. Thus much computation is spent on facts that do not change, as well as facts that do. Furthermore, it is not clear that any benefits would be derived from the ATMS, since there is no explicit relation between states.

What is desired is a planning structure that will handle the frame problem, but one that will also allow efficient, ATMS-like search of the plan space. The STRIPS planning system[6] is one of the best known planners that handles the frame problem adequately, although several possible limitations of it have been discussed.[7,8] STRIPS conceptualizes actions as operators on sets of sentences. An example of he structure of a STRIPS operator is given by **ACTION = {P,A,D}**, where **P** (the *precondition*) is a set of precondition formulas, **A** (the *add list*) is a set of formulas to be added by the operator, and **D** (the *delete list*) is a set of formulas to be deleted by the operator. If **ACTION** is to be applied to a particular state of the world, every formula in the precondition set **P** must be satisfied. If they are, then every formula in the old state is copied into the new state except for those that are in the delete list **D**, as instantiated by the variable assignment **V** used to satisfy **P**. Also, every formula in the add list **A**, instantiated by **V**, is added to the new state. Thus, only the facts that changed are computed: the frame axioms required by the situation calculus are not needed.

There are two possible ways one can use STRIPS operators as assumptions. The context independent method is to immediately assume all the operator instantiations that will be needed. As discussed previously, this is inappropriate for the ATC domain. If assumptions correspond merely to the presence of a particular action in a plan, as opposed to a particular ordering of an action in the plan, then this method also has the disadvantage of not being able to readily deal with plans that may require more than one instantiation of the same operator. This is not an insurmountable problem, but it must be addressed if this approach is taken.

The other method is the context dependent assumption method. The design of such a planner that will be considered here is one in which there exists a set of control rules that hypothesize some subset of all the actions whose precondition lists are satisfied in the current context. For simplicity, assume a very simple (but inefficient) control in which all such possible actions are hypothesized. Then the STRIPS implementation of an action **ACTION = {P,A,D}** in such an ATMS is:

IF CONTROL and P => ASSUME that ACTION is being undertaken with

the variable bindings B that satisfied P.

IF ACTION is being undertaken with bindings B => ASSERT every expression in A and retract every expression in D (with free variables bound in accordance with B).

In graph-theoretical terms, if the first rule matches in an environment, or hypothetical world corresponding to a particular set of assumptions, called **C1**, it creates a new daughter context **C2** corresponding to the hypothetical world just like **C1**, but in which the assumption that **ACTION** is undertaken is also present. Every fact in **C1** is "inherited" by **C2** -- the STRIPS assumption. The second rule is a STRIPS-like simulation of **ACTION**, modifying which facts are known to be true in the new context.

The context-dependent method is more desirable for the ATC domain, since there are so many operator instantiations; thus, it is desirable to assume only those that are promising in a given situation. It also promises to be a solution to the problem of multiple occurrences of the same action in a plan vs. the nonordered nature of action assumptions. This potential should be the subject of further research.

Although the TMS and the ATMS both allow a limited form of nonmonotonic inference[4], it is not immediately apparent how they can provide for the type of nonmonotonicity that a STRIPS-like planning system can require. To deal with the frame problem in STRIPS, certain facts must be *retracted* when an action is performed. However, an apparent incompatibility between the monotonicity of de Kleer's ATMS and using STRIPS operators as assumptions derives from the following: If fact **F** is true in the state corresponding to an assumption set **S**, then **F** must be true in any super-set of **S**. This is a direct consequence of the definition of a "merge" of assumption sets, the union operation mentioned above. Or as de Kleer puts it:

> "... problem solvers act, changing the world, and this cannot be modeled in a pure ATMS in which there is no way to prevent the inheritance of a fact into a daughter context."[9]

This is clearly unacceptable for the a STRIPS system if assumptions are to correspond to operators. For example, **LIGHT-ON** may be a fact true in a world corresponding to an assumption set **S** of operators, but it is not necessarily true in any world corresponding to a super-set of **S**; i.e., it is not true in **S U {TURN-OFF-LIGHT}**, where **LIGHT-ON** in a member of **TURN-OFF-LIGHT**'s delete list.

Thus, in an ATMS that uses assumptions to represent actions, facts must be retracted. A description of how the ATMS apparatus may be used to implement retraction is given by Morris and Nado.[10] The basic idea of their implementation is to make every added fact dependent on an additional non-deletion assumption. Then, if an action is to delete a fact, the fact corresponding to that action is given a justifier that makes it inconsistent with the non-deletion assumption. It is constructive to point out that deleting a fact in STRIPS is equivalent, in a TMS, to changing its status from **IN**, or known to be true, to **OUT**, a status where the fact is neither known to be true nor false. Thus it will at least be needed to reintroduce the notions of retraction and **IN/OUT** that the ATMS shed from its TMS roots.

The retraction issue is important, since an important test of whether an ATMS helps in searching the plan space is to determine whether it prevents rederiving inferences, even with retractions involved. The nominal ATMS approach to avoiding fact rederivation is to have a simple function that will compute the facts true in a context from the facts true in the parents of the context. One view is that the normal merge function, set union, can be used in a planning domain if you restrict the domain to having only mutually commutative operators. In general, two actions are said to be commutative if the order in which they are executed makes no effective difference. The particular definition of this in terms of STRIPS is that two operators **A** and **B** are said to be commutative if the set of sentences known to be

true after first applying operator **A** and then **B** to the initial world model is always identical to the set of sentences known to be true after applying operator **B** first, then **A**, to the initial world model.

This view is seemingly common. For example, Morris and Nado, who seem sensitive to many of the details of the ATMS in planning nevertheless introduce the merge problem as being a consequence of non-commutative actions:

> "More generally, a difficulty arises in that the effect of changes may depend on the order in which they are applied, resulting in an ambiguous merge." (Ref. 10, p 15).

But this is misleading, and to be fair, the treatments by Morris and Nado, as well as by Kalme,[11] indicate that they realize that interaction, as opposed to commutativity is what is crucial. To show this, consider a blocks world domain where the action of sliding a block is allowed. Let **A** and **C** be blocks on the left and right ends, respectively, of a row of blocks on the table. Then certainly the result of pushing **A** two locations to the right and then **C** three locations to the left will be the same as performing the same actions in an inverted order. But this does not mean that the worlds resulting from the composition of actions are directly computable, via some simple merge function, from the states that result after performing these actionas on **A** and **C** independently. Whether you wish to view the conclusion of this example as "commutativity is not enough to insure the applicability of a simple union merge function", or as "the above definition of commutativity was inadequate, and should instead require non-interaction between the actions" is a matter of choice.

Morris and Nado offer the following definition of when a particular merge operation is acceptable:

> Theorem 3: A sufficient condition for a merge to be unambiguous is that the ancestor subgraph may not contain two worlds, one of which deletes a fact and the other of which adds it, such that neither is an ancestor of the other. (Ref. 10, p 16).

Intuitively, it seems that the only domains that will allow a simple merge operator to meet the above requirements will be artificial, or uninteresting, or both. However, in the ensuing discussion of primary and secondary facts, it will be seen how a particular conceptualization of a non-trivial domain like ATC will allow the simple union merge to satisfy the requirements of Morris and Nado's theorem. That is, it will be shown how one can use STRIPS and the ATMS on a non-interacting basis of a domain in order to cheaply calculate the effect of actions in the entire domain.

The following example illustrates the computational savings that noninteracting actions allow, and demonstrates a context in which the ATC domain can be considered non-interacting. Suppose that an aircraft, **UA114**, which is near touchdown, decides to execute a missed approach, and a revised plan for this aircraft is generated. Note that this initial replanning is done independently of other aircraft which may be present in the terminal area. However, rescheduling **UA114** along the desired routing results in a conflict situation between **UA114** and another aircraft **CO718**; i.e., the separation time between them when they are intrail is 30 seconds less than that which is allowed. For each flight, the planning alternatives are **SPEED-UP**, **SLOW-DOWN**, and **HOLD**, the latter option being the least desirable. Given that **CO718** will be ahead of **UA114** when they are intrail, a hypothesized **SLOW-DOWN** for **CO718** and a hypothesized **SPEED-UP** for **UA114** can be heuristically ruled out as options, since a conflict will probably still exist even if these advisories are followed, while **HOLD** should be considered for either flight only if necessary. Therefore, **SLOW-DOWN (UA114)** and **SPEED-UP (CO718)** are hypothesized. In each hypothetical world, suppose it is discovered that there is still a conflict of 12 seconds between the aircraft when they are intrail. But the world that results from taking both actions can be computed directly, automatically, from the parent worlds, and it is discovered without

further inference that this world will avoid conflict by roughly 6 seconds.

This example demonstrates another important difference between using the ATMS in planning and in other problem-solving domains: in planning it is not always desirable to tag non-solutions as contradictions. Otherwise the worlds that were the parents of the eventual solution would have been tagged as contradictory worlds (no-goods), and there would have been no further merging or reasoning involving them, thus missing the most efficient plan.

Attempting to use the ATMS directly in interacting planning domains remains a problem. To accommodate the necessary ability to retract facts, the merge operation for computing conjunctions of assumption sets in interacting domains must be redefined. Morris and Nado suggest several merge algorithms. An adaptation of one of these, the "pessimistic merge" algoritm, is used in ART*, which defines the facts true in the merge of two assumption sets to be the union of the facts in each of the merging worlds minus any facts that are **OUT** in either parent.[12] However, there is no indication that a general merge exists for non-interacting domains.

The mixing of commutative and non-commutative operators in a domain does not seem to be a problem; however, it will not generally be possible to avoid rederiving inferences in a world formed by two or more interacting action assumptions.

Because the ATMS must handle retract and **IN/OUT** relationships, several of the ATMS advantages de Kleer mentions are lost. For instance, the outing problem, dealt with effectively by the ATMS in monotonic domains, must be dealt with again. And unlike the ATMS (which does not allow retraction) but like the TMS (which uses retraction as a means to get some of the effects an ATMS does) a system like the one being described will need some kind of dependency-directed updating scheme. If **P** lets you derive **Q** from **P => Q**, and **P** is later retracted, it is often desirable to change **Q**'s status from **IN** to **OUT**.

PRIMARY AND SECONDARY RELATIONS

The need for an efficient dependency-directed updating scheme that an ATMS in a planning domain requires can be satisfied by conceptualizing the domain into primary and secondary relations[13]. The idea behind primary and secondary relations is simply this: that there exists some subset, a *basis set* , of the predicates in a domain such that all the other predicates can be expressed completely in terms of the members of the basis set. In a trivial sense, this notion is domain independent: there is always the trivial basis set of all the predicates. But the notion seems to be domain independent in a broader sense as well, because many domains, including ATC, have an underlying set of primary qualities that form a proper subset of all the relations.

A formal definition of this notion of primary relations is desired, but there seem to be some fundamental problems in giving such a definition. One approach is to draw upon the metaphor of linear algebra. One can view the space of sentences for a given model as an analog of a linear vector space. In linear algebra it is straight forward to define the notion of a basis for a space, a minimal set of vectors which still spans the space, in terms of the linear independence of a set of vectors: a set of vectors is linearly independent if they span the **0** vector uniquely. Define a set of predicates **B** to be logically independent in a model **S** iff for every predicate **P** in **B** there does not exist a sentence comprised entirely of the other predicates in **B** that is logically equivalent to **P**. Note that this sentence cannot involve constant symbols either, or else every predicate could be expressed as a disjunction of equalities. Thus there can be no sentence of the form $\forall$ **x [P(x) iff B(x)]** that is true in the model, where **x** is the vector of all free variables of **P**, the predicate being defined, and where **B(x)** is a sentence com-

* ART™ is an automated reasoning tool developed by Inference Corporation; the ATC Advisor uses ART™.

posed entirely of other predicates in **B** with just those variables in **x** free.

Perhaps logical independence can be more clearly seen to be an analog of the standard linear algebraic definition of linear independence when it is defined as follows: a set of predicates **B** is logically independent if the only contradictory sentences (the analog of the **0** vector) formed from them, when written in disjunctive normal form, involve, for at least one predicate **P** in the set, **P** and its negation, ¬**P**.

Both of these definitions are problematic, since they are extensional-based definitions in a particular model. What relations are defined as logically independent will depend on the particular model. For instance, in the blocks world, if the model one is using to make the primary/secondary distinction assigns the same set of objects to the predicate **RED** as it does to **ON-TABLE**, these two predicates will not be independent, since they are co-extensional. Intuitively, it is desirable to rely on the intention of the predicates so that models in which the extensions of the two predicates differ can be anticipated. This is the inherent problem of relying on one model to construct the basis. Solutions to this might include: 1) using sets of models, or systematically varying models to get at the intention; 2) using domain knowledge about the predicates to differentiate intentionally distinct but extensionally identical predicates.

Given some (perhaps nonformal) definition of logical independence, however, define a set of predicates **B** to be a basis for a model **S** if 1) **B** is logically independent with respect to **S**; and 2) for every predicate **P** in **S**, there is a sentence composed entirely of the predicates in **B** that is logically equivalent to **P**. Note that since every predicate is logically equivalent to some sentence of **B** predicates, every sentence **s** in the model is logically equivalent to some other sentence of **B** predicates. Define such a sentence to be a basic sentence for **s** in **S** with respect to **B**. Also note that there may be many such sentences for a particular model, sentence, and basis.

The standard idea for the use of the primary/secondary distinction is to have the planner update only the basis facts, and then derive secondary facts as needed from the (consistent) primary model. If a primary fact is deleted, all secondary facts derived from it are deleted as well (changed from **IN** to **OUT** status). Thus, one can see why it does not seem that unlikely that a non-trivial domain such as ATC can have basis with non-interacting actions. The basis relations of the ATC domain depend only on one flight, so that changing a flight **F1**'s plan is the only way to change primary facts about **F1**: replanning another flight **F2** cannot influence or be influenced by the primary relations involved in replanning **F1**. Morris and Nado's criterion is met.

This straightforward application of the primary/secondary distinction has been criticized by Waldinger[7]. Waldinger did admit that the primary/secondary distinction does: 1) simplify action descriptions; 2) makes model updating more efficient; and 3) allows the system designer to introduce new relationships without needing to modify the actions' descriptions, which are all valid points. Nevertheless, it was argued: 1) keeping an updated model of just the primary facts is still expensive; 2) not all relations that are derived from **P** need to be deleted when **P** is deleted, thus necessitating unnecessary rederivation; 3) it may be impossible to define some lesser relationships in terms of the important ones. The first point is not a substantial criticism since Waldinger has already admitted that his approach is less efficient. The second point is the most telling of the three, although its solution lies in a sophisticated logical-dependency scheme, if such a solution exists. The last criticism should be kept in mind, but it does not seem very likely (cf the comments above about the trivial basis of all predicates). If fact, it seems very likely that Waldinger did not mean this literally, given the context of the remark.

However, another idea for integrating primary facts, planning, and the ATMS has been suggested: use the ATMS to generate the state space for the planner. This would require a conceptualization of the domain

into primary and derived relations. The system would then specify in advance all the possible primary facts it may encounter, and use them as assumptions to generate the state space. The ATMS would then do the necessary bookkeeping for the secondary or derived facts. It would be the planner's responsibility to be able to determine which hypothetical world corresponds to which planning state, but once found, no rederivation would be necessary. Unfortunately, one of the requirements for this approach is that there be a small bound on the number of operator instantiations in the domain, which is not a characteristic of our conceptualization of the ATC domain. For example, there are potentially an infinite number of instantiations of the **HOLD** operator: when and how long a controller may want to hold an aircraft at a particular waypoint are both continuous variables. Thus the number of necessary assumptions would be virtually infinite.

However, one could envision an incremental version of this system, where the initial assumptions in the ATMS are only the basis facts known in the initial state. Then the planner would use domain knowledge to select an operator, use the 4D algorithm to determine what new basis facts are true in that world, and add these as assumptions in the ATMS. As above, the planner would be responsible for control and the maintenance of primary facts, while the ATMS would prevent any recomputation of derived, secondary data, but it would not be necessary to specify the state space in advance. This idea of using an ATMS to efficiently organize the state space is a promising one and deserves closer evaluation.

THE SEMANTICS OF STRIPS

Lifschitz[8] has demonstrated that some intuitive soundness conditions for STRIPS operators actually cause STRIPS to be unsound. He proposes some modifications to the basic STRIPS structure that would make STRIPS sound according to the definition of soundness he proposes, but that would limit the expressiveness of STRIPS. However, if the conceptualization of the domain is sensitive to the notion of what will be called the primary/secondary relation distinction, then the soundness condition can be restated so that it still matches our intuitions about what such a soundness condition should be, and also provides for a sound, expressive variant of STRIPS that exploits the primary/secondary relation distinction.

The definition of soundness that Lifschitz initially offers seems intuitive enough:

> **Definition A: An operator description (P,D,A) is sound relative to an action f if, for every state s such that P is satisfied in s,**
>
> **(i) f is applicable in state s;**
> **(ii) every sentence which is satisfied in s and does not belong to D is satisfied in f(s);**
> **(iii) A is satisfied in f(s).**

A STRIPS system Σ is sound if the initial model is satisfied in the initial state, and each operator description $(P_\beta, D_\beta, A_\beta)$ is sound relative to f_β. Define a sequence of operators (a plan) to be accepted by Σ if the model produced by applying the ith operator proves the precondition of the i+1th operator. Lifschitz then proves a soundness theorem, which claims that if Σ is sound, and a plan p is accepted by Σ, then the action corresponding to the plan f_p is applicable in s_0, and the final model produced by executing p is satisfied by $f_p(s_0)$. Therefore, if the individual STRIPS operators can be shown to be sound, then a STRIPS planning system will not yield models of the world that are unsatisfied, i.e. impossible.

The problem that Lifschitz points out is that according to this definition of soundness, no usual STRIPS system is sound. Consider a schematic of the operator description corresponding to the action of turning on a light:

```
TURN-ON-LIGHT =
P = LIGHT-OFF;
A = {ROOM-LIT, LIGHT-ON};
D = {ROOM-DARK, LIGHT-OFF}.
```

This can be safely taken as an example of a typical operator description in the usual use of STRIPS. Lifschitz points out the obvious unsoundness of this operator: consider any sentence of the form **(LIGHT-OFF OR X)**, where **X** is any sentence not satisfied in **f(s)**. Clearly, if this operator is applied to a model of a state **s**, then these sentences must be satisfied in the state **s**. They are also not in the delete list. So if the operator is to be sound they must be satisfied in **f(s)**. But they are not. If the operator is to be sound, the delete list must be infinite and, as Lifschitz suggests, perhaps non-recursive.

Lifschitz's proposals for solving this problem are not satisfying. He initially suggests that one could simply require STRIPS to be sound on the ground atoms. But this would require that the only non-ground atoms you can allow are those that are true in all worlds, thus preventing STRIPS from being a powerful formalism.

His second suggestion of using essential formulas as a means of providing a soundness condition for special STRIPS formulations is a step in the right direction, but it is still limiting. The idea is to allow STRIPS to operate on particular non-ground atoms, which must be specified in advance. The awkwardness of this approach is a direct consequence of the same element in his third definition of soundness that caused problems in the first two approaches: membership in the delete list determines what is deleted.

As long as one is willing to redefine STRIPS (or a least create a new variant of it), the following modification to the STRIPS operator application procedure is offered as a partial solution to the soundness problem: not only are the sentences in **D** deleted from the model to which the operator is applied, but so are any sentences provable from **D**. To define the soundness of this variant of STRIPS, define an operator to be sound in the same way as definition **A** above, except:

> **ii) every sentence which is satisfied in s and which is not *provable* from D is satisfied in f(s).**

This new STRIPS and definition of soundness is immune to criticisms like the above involving turning on the light. But it is not yet clear if there are other reasons why such operators might not be acceptable. Consider the case where **P() => Q()** and **P()** are in in the initial model, and **Q()** is in the delete list. Then the new STRIPS variant defined above would cause the implication to be deleted and the formula **P** to remain. Call this the "sentential **P => Q** problem" (note that this does not happen in the non-sentential case; see below). There certainly might be instances where one would want STRIPS to do just the opposite. How can this distinction be made?

Some formulas (usually implications) can be given some immunity status, by being included in a basis set **B** (This set is not to be confused with idea of basis relations, although a connection will be proposed below). Then STRIPS would never be allowed to delete formulas in **B**. These formulas are therefore ones that are true in all states of the world, similar to the notion that Lifschitz was trying to reach via essential formulas. But there still remains the **P => Q** problem for protected sententials and non-sententials: If the sentence **P => Q** does not participate in the deletion process, the cause of **Q**'s presence in the model, **P**, will not be removed and **Q** will just be rededuced. So the basis formulas must participate in the deletion process.

The simple way to do this would be to admit only formulas that are not proved by **D U B**. However, the delete lists, **D**, have a negative bias, indicating what is no longer true, while the basis formulas, **B**, have a positive bias, expressing what is true in all worlds. This causes problems like the following: **B = {P => Q}**, **D = {Q}**, but **[D U B] |- P**, so **P** is not deleted as desired.

One way to counteract this is to give the sets **B** and **D** the same bias. Thus a formula is admitted only if its negation is not proved by **B U ¬D**. A straight forward interpretation of the notion of a set of formulas' negation is simply the set of the negations of each of the formulas, although alternative definitions are possible. But if formulas

of **D** are non-atomic, this method might be incoherent. As an example, consider the delete list **D = {P => Q, ¬Q}**. This seems like an intelligible thing for an action to do, but **¬D = {P and ¬Q, Q}**, from which anything is provable, so nothing is admitted. This does not seem to be an intended consequence of deleting **P => Q** and **¬Q**. This is also a motivation for not making the bias of **B** and **D** the same by changing **B** to not **¬B**.since the elements of **B** will almost certainly be complex, and probably of the form of implications.

One solution is to require that a formula be admitted iff its negation is not provable from any set consisting of an element of D and the members of B. Now consider the following: **B = {Q}, D = {P => Q}**. Then **B U ¬D** is a contradiction, which may seem inappropriate; but if **Q** is always true, then **P => Q** must always be true. This approach will suffice for now, although alternative mechanisms can be imagined, with interesting differences; consider requiring that a formula be admitted iff its negation is not provable from the cross-product of **B U ¬D**.

Although it looks as if a sound, yet powerful version of STRIPS has been described, it is not yet clear how the operators in such a formalism can be derived and made efficient and complete. For these reasons, the notion of primary and secondary relations is again used, and one can require the members of the delete lists to only be constructed out of primary relations. Since all secondary relations can be derived from the primary relations, one could simply include a basic sentence for all of the secondary relations in the protected set **B**. The idea is superficially the same as Lifschitz's essential sentences, except that by referring to essential relations only, the primary/secondary relations approach allows much more flexibility. It was seen above that the formalism had to be complexified if one wished to include complex formulas in **D**. It remains a conjecture that with basis relations there would be no need to delete complex formulas. How will this restriction address the three issues of conceptualization, efficiency, and completeness?

Since one of the criteria with which a planning system can be evaluated is the ability (of a knowledge engineer) to actually provide operator descriptions that meet these soundness requirements, some formal notion of the ability to produce operator descriptions needs to be introduced to serve as a bridge between theory and practice. Thus the Action Conceptualization Hypothesis (ACH) is introduced:

> **For any action it is possible to conceptualize the effects that action has on the primary qualities of the domain.**

Note that it is not as important to claim that this is always true as it is to claim that if this is not true for a particular domain, then there is little hope of a successful planning system using any known Artificial Intelligence approach. In this respect, the ACH seems to be rather uncontentious. Of course, one cannot formalize "primary qualities", for reasons similar to the extension vs. intention problem in formalizing logical independence, above.

Define an operator description **O** to be *basically sound* relative to an action **f** if it is sound in the model identical to **S** except that only the primary relations exist. Of course, it must be defined for such a model, so it must only use primary relations. Then by application of our ACH, it can be claimed that for any domain since there always exists a basis set for a model of that domain, with the members of the basis set corresponding to the primary qualities of the domain, if **A** is an action in the domain, a corresponding operator description O_A that is basically sound can be produced.

From this, it can be claimed that sound operator descriptions can always be produced, since basically sound operators can be produced. Thus it has been made explicit what conditions must hold about a domain in order for STRIPS to be sound in that domain.

Also, the STRIPS formalisms mentioned above require an efficient means of determining whether a fact is implied by **B U ¬D**. Since every sentence has an equivalent ba-

sis sentence, the use of delete and add lists using only primary relations can facilitate this provability check. The structure of **B** and **D** will dictate the degree to which this can be done; thus, it is not surprising that representing **D** in terms of a minimal basis will minimize computation, although this has not been proven.

Completeness issues are also relevant to the primary/secondary distinction. It is easy to imagine a completeness analog of the ACH, which would ensure general completeness. Of course, true completeness is not desirable or possible, but some notion of basic completeness might be introduced using primary relations and some artificial criteria to limit the scope of required completeness. For example, an operator could be said to be complete if it designates as **IN** in the post-action model all sentences that are 1) basic; 2) satisfied in the corresponding post-action world; and 3) were previously known to be **IN** in the pre-action model. In such a scheme, the set of sentences whose **IN /OUT** statuses are updated would only increase via add lists.

CONCLUDING REMARKS

The planning formalisms discussed herein demonstrate that the noninteracting nature of the ATC domain can be exploited to provide an efficient planner that avoids the frame problem. Further specification of particular planning operators and primary/secondary relation distinctions is necessary, as is more research in system display and controller-machine interaction issues in order to completely integrate these formalisms. An analysis of a multi-level ATMS (such as ART™) is a natural extension to this work.

ACKNOWLEDGMENTS

We would like to thank the many people who gave constructive comments on our ideas and this paper, and would especially like to thank Johan de Kleer, Robert Filman, Michael Georgeff, Matthew Ginsberg, Benjamin Grosof, Charles Kalme, Vladimir Lifschitz, L. M. Pereira, Nils Nilsson, Paul Rosenbloom, Devika Subramanian, and Chuck Williams for their assistance and/or encouragement.

REFERENCES

[1]Tobias, L., and Scoggins, J. L., "Time-Based Air Traffic Management Using Expert Systems", *IEEE Control Systems Magazine*, Vol. 7, No. 2, April, 1987.

[2]Nilsson, Nils J., *Principles of Artificial Intelligence*, Palo Alto: Tioga Publishing Corporation, 275-360.

[3]Doyle, Jon, "A Truth Maintenance System" in *Artificial Intelligence*, 12:231-272, 1979.

[4]de Kleer, Johan, "An Assumption-based TMS", *Artificial Intelligence*, 28:127-162, 1986.

[5]McCarthy, John, "Situations, Actions, and Causal Laws," Stanford University Artificial Intelligence Project Memo no. 2; reprinted in *Semantic Information Processing*, Marvin Minsky (editor), pp 410-418, Cambridge: MIT Press, 1968.

[6]Fikes, Richard E. and Nilsson, Nils, "STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving", *Artificial Intelligence*, 2:189-208, 1971.

[7]Waldinger, Richard, "Achieving Several Goals Simultaneously", *Machine Intelligence*, 8:94 -136, 1977.

[8]Lifschitz, Vladimir, "On the Semantics of STRIPS", *Reasoning about Action and Plans: Proceedings of the 1986 Workshop.* Timberline, Oregon, 1986.

[9]de Kleer, Johan, "Problem Solving with the ATMS", *Artificial Intelligence*, 28:197-224, 1986.

[10]Morris, Paul H. and Nado, Robert A., "Representing Actions with an Assumption-based Truth Maintenace System", *AAAI-86 Proceedings*, 13-17, 1986.

[11]Kalme, C., "The Structure of the ART Viewpoint Graph and the Dynamics of Rule Firing Across the Graph," Inference Corporation, 1984.

[12]Williams, Chuck, "Managing Search in a Knowledge-based System", Inference Corporation, Los Angeles, CA, 1984.

[13]Fahlman, Scott E., "A Planning System For Robot Construction Tasks", *Artificial Intelligence*, 5:1-49, 1974.