ON THE SEMANTICS OF STRIPS

Vladimir Lifschitz
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

STRIPS is a problem solver which operates with world models represented by sets of formulas of first-order logic. A STRIPS system describes the effect of an action by a rule which defines how the current world model should be changed when the action is performed. The explanations of the meaning of these descriptions in the literature are very informal, and it is not obvious how to make them more precise. Moreover, it has been observed that minor and seemingly harmless modifications in standard examples of STRIPS systems cause STRIPS to produce incorrect results. In this paper we study the difficulties with interpreting STRIPS operator descriptions and define a semantics which draws a clear line between "good" and "bad" uses of the language of STRIPS.

1. Introduction

STRIPS (Fikes and Nilsson 1971) is a problem solver which operates with *world models*, represented by sets of formulas of first order-logic. A STRIPS system is defined by an *initial* world model, which describes the initial state of the world, and by a set of *operators*, which correspond to actions changing the current state. Using means-ends analysis, STRIPS attempts to find a sequence of operators transforming the initial world model into a model which satisfies a given *goal formula*.

The *description* of each operator consists of its *precondition* (the applicability condition, expressed by a first-order formula), its *add list* (the list of formulas that must be added to the current world model), and its *delete list* (the list of formulas that may no longer be true and therefore must be deleted). A resolution theorem prover is used for the verification of operator preconditions, for establishing the validity of the goal formula in the last world model, and also for directing the search.

The explanation of the meaning of operator descriptions in (Fikes and Nilsson 1971) is very brief and is almost completely reproduced in the parenthesized comments above. It is not immediately clear how to make this explanation more precise; more specifically, it turns out to be a non-trivial task to define under what conditions the delete list of an operator may be considered sufficiently complete. Moreover, some minor and seemingly harmless modifications in the main example of (Fikes and Nilsson 1971) cause STRIPS to produce incorrect results (see Sections 4 and 5 below). Alan Bundy observes that the AI literature "abounds with plausible looking formalisms, without a proper semantics. As soon as you depart from the toy examples illustrated in the paper, it becomes impossible to decide how to represent information in the formalism or whether the processes described are reasonable or what these processes are actually doing" (Bundy 1983). Is STRIPS a formalism of this sort?

In this paper we do the additional theoretical work needed to make sure that this is not the case. We study the difficulties with interpreting STRIPS operator descriptions and define a semantics which draws a clear line between "good" and "bad" uses of the language of STRIPS.

2. Operators and Plans

We start with an arbitrary first-order language L. A *world model* is any set of sentences of L. An *operator description* is a triple (P, D, A), where P is a sentence of L (the *precondition*), and D and A are sets of sentences of L (the *delete list* and the *add list*).

Consider an example from Section 3.2 of (Fikes and Nilsson 1971). In this example, the language contains some object constants and two predicate symbols,

unary ATR and binary AT. Intuitively, the language is designed for describing the locations of a robot and of other objects. We think of the universe of the intended model as consisting of these objects and their possible locations. $ATR(x)$ means that the robot is at location x. $AT(x, y)$ means that the object x is at location y. Let now k, m, n be object constants of this language. The operator $push(k, m, n)$ for pushing object k from m to n is described by the triple:

Precondition: $ATR(m) \wedge AT(k, m)$.

Delete list: $\{ATR(m), AT(k, m)\}$.

Add list: $\{ATR(n), AT(k, n)\}$.

What we have defined here is a family of operator descriptions, one for each triple of constants k, m, n. The precondition shows that the corresponding action is possible whenever the robot and object k are both at location m. The delete list tells us that these two facts should be removed from the current world model when the operator $push(k, m, n)$ is applied. The add list requires that the information about the new location of the robot and of object k, represented by the formulas $ATR(n)$ and $AT(k, n)$, be added to the model.

A *STRIPS system* Σ consists of an *initial* world model M_0, a set Op of symbols called *operators*, and a family of operator descriptions $\{(P_\alpha, D_\alpha, A_\alpha)\}_{\alpha \in Op}$.

Section 4 of (Fikes and Nilsson 1971) introduces a STRIPS system which represents a world consisting of a corridor with several rooms and doorways, a robot and a few boxes and lightswitches. The language contains, in addition to ATR and AT, some other predicate symbols, for instance:

$TYPE(x, y)$: x is an object of type y,

$CONNECTS(x, y, z)$: door x connects room y with room z,

$NEXTTO(x, y)$: object x is next to object y,

$INROOM(x, y)$: object x is in room y,

$STATUS(x, y)$: the status of lightswitch x is y.

The initial world model in this example consists mostly of ground atoms, such as

$$TYPE(DOOR1, DOOR),$$
$$CONNECTS(DOOR1, ROOM1, ROOM5),$$
$$INROOM(ROBOT, ROOM1),$$
$$STATUS(LIGHTSWITCH1, OFF).$$

It contains also one universally quantified formula,

$$\forall xyz(CONNECTS(x, y, z) \supset CONNECTS(x, z, y)). \qquad (1)$$

Among the operators we find:

$goto1(m)$: robot goes to location m,

$goto2(m)$: robot goes next to item m,

$pushto(m, n)$: robot pushes object m next to object n,

$gothrudoor(k, l, m)$: robot goes through door k from room l to room m,

$turnonlight(m)$: robot turns on lightswitch m,

and a few others.

This system will be subsequently referred to as the "main example".

Given a STRIPS system Σ, we define a *plan* to be any finite sequence of its operators. Each plan $\overline{\alpha} = (\alpha_1, \ldots, \alpha_N)$ defines a sequence of world models $M_0, M_1, \ldots, M_N$, where M_0 is the initial world model and

$$M_i = (M_{i-1} \setminus D_{\alpha_i}) \cup A_{\alpha_i} \qquad (i = 1, \ldots, N). \tag{2}$$

We say that $\overline{\alpha}$ is *accepted* by the system if

$$M_{i-1} \vdash P_{\alpha_i} \qquad (i = 1, \ldots, N). \tag{3}$$

In this case we call M_N the *result* of executing $\overline{\alpha}$ and denote it by $R(\overline{\alpha})$.

In what terms do we want to describe the semantics of STRIPS?

We think of the world described by the language L as being, at any instant of time, in a certain *state*; we assume that one of the states, s_0, is selected as *initial*. We assume that it is defined for each state s which sentences of L are (known to be) *satisfied* in this state, and that the set of sentences satisfied in state s is closed under predicate logic. An *action* is a partial function f from states to states. If $f(s)$ is defined then we say that f is *applicable* in state s, and that $f(s)$ is the *result* of the action. We assume that an action f_α is associated with each operator α. A STRIPS system along with this additional information will be called an *interpreted* STRIPS system.

A world model M of an interpreted STRIPS system Σ is *satisfied* in a state s if every element of M is satisfied in s. For each plan $\overline{\alpha} = (\alpha_1, \ldots, \alpha_N)$ of Σ, we define $f_{\overline{\alpha}}$ to be the composite action $f_{\alpha_N} \ldots f_{\alpha_1}$.

3. Semantics: A First Attempt

Consider a fixed interpreted STRIPS system $\Sigma = (M_0, \{(P_\alpha, D_\alpha, A_\alpha)\}_{\alpha \in Op})$. Our goal is to define under what conditions Σ can be considered *sound*. We start with the most straightforward formalization of the intuition behind operator descriptions.

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 $f(s)$.

Σ is *sound* if M_0 is satisfied in the initial state s_0, and each operator description $(P_\alpha, D_\alpha, A_\alpha)$ is sound relative to f_α.

Soundness Theorem. *If Σ is sound, and a plan $\overline{\alpha}$ is accepted by Σ, then the action $f_{\overline{\alpha}}$ is applicable in the initial state s_0, and the world model $R(\overline{\alpha})$ is satisfied in the state $f_{\overline{\alpha}}(s_0)$.*

Proof. Let $\overline{\alpha} = (\alpha_1, \ldots, \alpha_N)$ be a plan accepted by Σ. Let us prove that for every $i = 0, \ldots, N$ action $f_{\alpha_i} \ldots f_{\alpha_1}$ is applicable in s_0, and M_i defined by (2) is satisfied in state $f_{\alpha_i} \ldots f_{\alpha_1}(s_0)$. The proof is by induction on i. The basis is obvious. Assume that M_{i-1} is satisfied in $f_{\alpha_{i-1}} \ldots f_{\alpha_1}(s_0)$. By (3), it follows that P_{α_i} is satisfied in this state too. Since $(P_{\alpha_i}, D_{\alpha_i}, A_{\alpha_i})$ is sound relative to f_{α_i}, we can conclude that $f_{\alpha_i} f_{\alpha_{i-1}} \ldots f_{\alpha_1}(s_0)$ is defined, and that both $M_{i-1} \setminus D_{\alpha_i}$ and A_{α_i} are satisfied in this state. By (2), it follows then that M_i is satisfied in this state too.

There is a serious problem, however, with Definition A: it eliminates all usual STRIPS systems as "unsound". Consider, for instance, the description of $push(k, m, n)$ given in Section 2. The two atoms included in its delete list are obviously not the only sentences which may become false when the corresponding action is performed. Their conjunction is another such sentence, as well as their disjunction or, say, any sentence of the form $ATR(m) \wedge F$, where F is provable in predicate logic. To make the delete list complete in the sense of Definition A, we would have to include all such sentences in it. The delete list will become infinite and perhaps even non-recursive!

The designer of a STRIPS system cannot possibly include in a delete list all arbitrarily complex formulas that may become false after the corresponding action is performed. In our main example, the delete lists of all operator descriptions contain only atomic formulas. The same can be usually found in other examples of STRIPS systems. When describing an operator, we can try to make the delete list complete in the weaker sense that all *atoms* which may become false are included. More precisely, we may be able to guarantee condition (ii) for *atomic* sentences, but it is not realistic to expect that it will hold for all sentences in the language.

It would be a mistake, however, to restrict (ii) to atoms in Definition A and make no other changes, because that would make the assertion of the Soundness Theorem false. World models may include non-atomic sentences, and the weaker

form of (ii) does not guarantee that such sentences are deleted when they become false. What is the right way to exploit this "atomic completeness" of delete lists?

One possible solution is to change the definition of a world model and require that it include atomic sentences only. In this case we should also allow only atomic formulas in add lists (otherwise $R(\overline{\alpha})$ will generally include non-atomic formulas and thus will not be a world model), and there will be no need to allow anything other than atoms in delete lists. (In this "atomic STRIPS", logical connectives and quantifiers would be still allowed in preconditions and in the goal formula).

This somewhat restrictive approach gives a satisfactory interpretation of many simple STRIPS systems. In fact, the description of STRIPS in (Nilsson 1980), for ease of exposition, allows only conjunctions of ground literals in world models, which is almost equally restrictive. But let us remember that our main example contains a non-atomic formula, (1). Why does that system appear to function correctly? This question is addressed in the next section.

4. Non-Atomic Formulas in World Models

Consider the description of the operator *turnonlight*($LIGHTSWITCH1$) in the main example. Its delete list is $\{STATUS(LIGHTSWITCH1, OFF)\}$. When the operator is applied, the atomic sentence $STATUS(LIGHTSWITCH1, OFF)$ (which is a part of the initial world model) will be deleted. Now let us change the example slightly and replace this atomic sentence in the initial world model by the stronger assumption that *all* lightswitches are originally turned off:

$$\forall x(TYPE(x, LIGHTSWITCH) \supset STATUS(x, OFF)). \qquad (4)$$

This formula will not be deleted when *turnonlight*($LIGHTSWITCH1$) is applied, which will cause STRIPS to malfunction.

Sentences (1) and (4) have the same logical complexity, and they are assumed to be both satisfied in the initial state of the world. What is wrong about including (4) in the initial world model? This example seems to confirm that "the frontier between "acceptable" and "ridiculous" (STRIPS-like) axiomatizations of the world is a very tenuous one" (Siklóssy and Roach 1975).

There is, however, an obvious difference between (1) and (4): the former is satisfied not only in the initial state, but in *every* state of the world. This difference is crucial. It is true that in the main example non-atomic formulas are never deleted from world models; but there can be no need to delete (1). This is why it is safe to include (1) in M_0.

A similar precaution should be taken with regard to including non-atomic formulas in add lists. We can extend the main example, for instance, by the operator *turnoffalllights*, with the add list consisting of one formula (4). If *turnonlight*(m)

is applied after this operator, we will have a difficulty similar to the one discussed above. A non-atomic formula may be included in an add list only if it is satisfied in every state of the world. (Of course, it can be included then in the initial world model as well.)

This discussion suggests the following modification of Definition A.

Definition B. 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 atomic sentence which is satisfied in s and does not belong to D is satisfied in $f(s)$,
(iii) A is satisfied in $f(s)$,
(iv) every non-atomic sentence in A is satisfied in all states of the world.

Σ is *sound* if

(v) M_0 is satisfied in the initial state s_0,
(vi) every non-atomic sentence in M_0 is satisfied in all states of the world,
(vii) every operator description $(P_\alpha, D_\alpha, A_\alpha)$ is sound relative to f_α.

The Soundness Theorem remains valid for the new definition.

Proof. Let $\overline{\alpha} = (\alpha_1, \ldots, \alpha_N)$ be a plan accepted by Σ. Let us prove that for every $i = 0, \ldots, N$ action $f_{\alpha_i} \ldots f_{\alpha_1}$ is applicable in s_0, M_i is satisfied in state $f_{\alpha_i} \ldots f_{\alpha_1}(s_0)$, and every non-atomic formula in M_i is satisfied in all states. The proof is by induction on i. The basis is obvious. Assume that M_{i-1} is satisfied in $f_{\alpha_{i-1}} \ldots f_{\alpha_1}(s_0)$, and all non-atomic formulas in M_{i-1} are satisfied in all states. It follows from (3) that P_{α_i} is satisfied in state $f_{\alpha_{i-1}} \ldots f_{\alpha_1}(s_0)$. Since $(P_{\alpha_i}, D_{\alpha_i}, A_{\alpha_i})$ is sound relative to f_{α_i}, we can conclude that $f_{\alpha_i} f_{\alpha_{i-1}} \ldots f_{\alpha_1}(s_0)$ is defined, that every non-atomic formula in $M_{i-1} \setminus D_{\alpha_i}$ or in A_{α_i} is satisfied in this state, and that every atomic formula in any of these two sets is satisfied in all states. By (2), it follows then that M_i is satisfied in state $f_{\alpha_i} \ldots f_{\alpha_1}(s_0)$, and that every non-atomic formula in M_i is satisfied in all states.

5. The General Semantics of STRIPS

Our work has not come to an end yet. A careful examination of the main example reveals a small detail which shows that, in spite of all our efforts, that system is *not* sound in the sense of Definition B.

This peculiarity, pointed out in (Siklóssy and Roach 1975), is connected with the delete lists of some operators which change the position of the robot: $goto1(m)$, $goto2(m)$, $pushto(m, n)$ and $gothrudoor(k, l, m)$. As can be expected, the delete lists of these operators contain ground atoms which describe the robot's current

position. They include all atoms of the form $ATROBOT(\$)$, where \$ is any object constant. They also include the atoms $NEXTTO(ROBOT, \$)$. However, they do *not* include $NEXTTO(\$, ROBOT)$.

This asymmetry is somewhat counterintuitive, because the authors apparently interpret *NEXTTO* as a symmetric predicate. For instance, the delete list of $pushto(m, n)$ (the robot pushes object m next to object n) contains both $NEXTTO(\$, m)$ and $NEXTTO(m, \$)$, and its add list contains both $NEXTTO(m, n)$ and $NEXTTO(n, m)$. One may get the impression that the non-symmetric treatment of *NEXTTO* with *ROBOT* as one of the arguments is an oversight.

However, this is not an oversight, but rather a trick carefully planned by the authors in the process of designing the main example. They make sure that assertions of the form $NEXTTO(\$, ROBOT)$ never become elements of world models in the process of operation of the system: there are no atoms of this form in M_0 or on the add lists of any operators. For example, the add list of $goto2(m)$ contains $NEXTTO(ROBOT, m)$, but not $NEXTTO(m, ROBOT)$, even though these atomic sentences both become true and, from the point of view of Definition B, nothing prevents us from adding both of them to the current world model.

The purpose of this is obvious: storing information on the objects next to the robot in both forms would have made the operator descriptions longer and would have led to computational inefficiency. In principle, it is possible to go even further in this direction and, for instance, store facts of the form $NEXTTO(BOXi, BOXj)$ only when $i < j$.

It is easy to accomodate the systems which use tricks of this kind by slightly generalizing Definition B. Let E be a set of sentences; the formulas from E will be called *essential*. Definition B corresponds to the case when E is the set of ground atoms.

Definition C. 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 essential sentence which is satisfied in s and does not belong to D is satisfied in $f(s)$,
(iii) A is satisfied in $f(s)$,
(iv) every sentence in A which is not essential is satisfied in all states of the world.

Σ is *sound* if

(v) M_0 is satisfied in the initial state s_0,
(vi) every sentence in M_0 which is not essential is satisfied in all states of the world,
(vii) every operator description $(P_\alpha, D_\alpha, A_\alpha)$ is sound relative to f_α.

It should be emphasized that Definition C defines the soundness of operator descriptions and STRIPS systems only with respect to a given class of sentences

that are considered essential. The choice of this class E is an integral part of the design of a STRIPS system, along with its language, its initial model, and the set of its operators with their descriptions. When a STRIPS system is introduced, it is advisable to make the choice of E explicit; the description of the main example, for instance, will be more complete if we specify that a sentence is considered essential in this system if it is an atom and does not have the form $NEXTTO(\$, ROBOT)$. This information will help the user to avoid mistakes when the initial model is modified to reflect different assumptions about the initial state of the world, or when new operators are added to the system.

The treatment of $NEXTTO$ in the main example shows that it may be advantageous to make E a proper subset of the set of ground atoms. Sometimes it may be convenient to include some non-atomic formulas in E. For instance, we may wish to update negative information by means of adding and deleting negative literals; then E would be the set of ground literals, both positive and negative.

The proof of the Soundness Theorem given in the previous section can be easily generalized to soundness in the sense of Definition C.

Acknowledgements

This research was partially supported by DARPA under Contract N0039-82-C-0250. I would like to thank John McCarthy and Nils Nilsson for useful discussions, and Michael Gelfond for comments on an earlier draft of this paper.

References

A. Bundy, *The Computer Modelling of Mathematical Reasoning*, Academic Press, 1983.

R. E. Fikes and N. J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence* **2** (1971), 189-208.

N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Company, Palo Alto, Califormia, 1980.

L. Siklóssy and J. Roach, Model verification and improvement using DISPROVER, *Artificial Intelligence* **6** (1975), 41-52.