STRIPS as an Action Description Language

STRIPS, as defined originally by Fikes and Nilsson, is not an action description language in the sense of the paper *Action Languages* by Gelfond and Lifschitz, because STRIPS operators represent operations on "world models"—not on states of a transition system. But we can turn STRIPS into an action description language if we modify its definition as follows.

Consider a fixed propositional action signature. A *formula* is a propositional combination of fluent names. A *literal* is a formula of the form P or $\neg P$, where P is a fluent name. A *STRIPS operator* is a pair $\langle F, X\rangle$, where F is a formula (the *precondition*), and X is a consistent set of literals (the *effects*). A fluent name P is in the *add list* of the operator if P belongs to X, and in the *delete list* if $\neg P$ belongs to X. In STRIPS, an *action description* is a function from action names to STRIPS operators.

Example 1. The fluent names are P and Q; the only action name is A. The STRIPS operator corresponding to A has precondition Q, and its only effect is P. More concisely, the action description is

A :
Precondition: Q
Effects: P.

Example 2. A building has N doors $d_1, \dots, d_N$. Each door can be locked or unlocked; this is represented by fluent names $\mathit{locked}(D)$, where D is one of $d_1, \dots, d_N$. Action name $\mathit{lock}(D)$ represents locking door D; this action is nonexecutable if D is already locked. Action name $\mathit{lockAll}$ represents locking all doors that are currently unlocked; this action is always executable. This domain can be formalized in STRIPS as follows:

$\mathit{lock}(D) \quad (D \in \{d_1, \dots, d_N\})$:
Precondition: $\neg \mathit{locked}(D)$
Effects: $\mathit{locked}(D)$.

$\mathit{lockAll}$:
Precondition: *True*
Effects: $\mathit{locked}(D) \quad (D \in \{d_1, \dots, d_N\})$.

Figure 1: Transition system described by Example 1.

In the following definition, $\mathbf{F}$ is the set of all fluent names of the underlying action signature, and $\mathbf{A}$ is the set of its action names. Let D be an action description in STRIPS. The transition system $\langle S, V, R\rangle$ *described* by D is defined as follows:

(i) S is the set of all interpretations of $\mathbf{F}$,

(ii) $V(P,s) = s(P)$ for every $P \in \mathbf{F}$ and every $s \in S$,

(iii) R is the set of all triples $\langle s, A, s'\rangle$, where $s, s' \in S$ and $A \in \mathbf{A}$, such that s satisifes the precondition of $D(A)$, and s' is defined by

$$s'(P) = \begin{cases} \mathsf{t}, & \text{if } P \text{ is an effect of } D(A); \\ \mathsf{f}, & \text{if } \neg P \text{ is an effect of } D(A); \\ s(P), & \text{elsewhere.} \end{cases}$$

Any transition system described in STRIPS is deterministic. (Why?)

Problem 2. Show that Example 1 describes the transition system depicted in Figure 1.

Problem 3. Draw the transition system described in Example 2 for $N = 2$.

In Figure 2 (copied from p. 12 of *Action Languages*), action A operates in a "state-dependent" way: it makes P true when Q is true, but it does

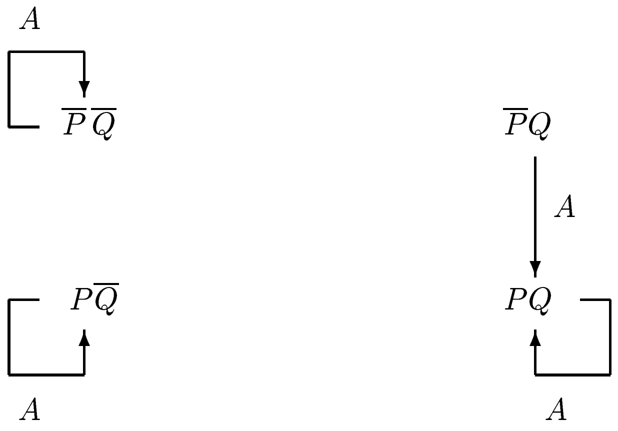

Figure 2: An action with state-dependent effects.

not affect P when Q is false. This transition system cannot be described in STRIPS.

Problem 4. Prove this assertion.

Problem 5. Action *Toggle* is always executable and changes the value of fluent *On* to the opposite. Draw the corresponding transition system and show that it cannot be described in STRIPS.

In a transition system represented in STRIPS, all interpretations of the set of fluent names are counted as states; STRIPS provides no mechanism for expressing that some combinations of values are physically impossible. Language $\mathcal{A}$ has this limitation also.

In view of this fact, many domains involving actions do not have fully adequate formalizations in STRIPS or in $\mathcal{A}$. But such a domain may have a formalization that is "semi-adequate" in the sense that

- the corresponding transition system, restricted to a subset of "good" states, represents the domain correctly, and

- the "bad" states of are not reachable from the "good" states.

For the purpose of specifying a planning problem, it is not necessary to provide a fully adequate formalization of the domain; semi-adequate formalizations are good enough.

Example 3. A monkey is assumed to be always at one of 3 locations. The position of the monkey can be described by fluent names $at(monkey, L)$, where L is l_1, l_2 or l_3. Any transition system described in STRIPS using these fluent names has 8 states, so that it can provide, at best, a semi-adequate formalization, with 3 "good" states and 5 "bad" states.

Problem 6. Consider the domain consisting of the monkey from Example 3 and the actions of walking to a location L, represented by action names $walk(L)$. Give a semi-adequate formalization of this domain in STRIPS and draw the "good" part of the corresponding transition system.

Problem 7. Consider the domain consisting of the monkey from Example 3, a box that is always at one of the given locations also, and the actions of pushing the box to a location L. Give a semi-adequate formalization of this domain in STRIPS and draw the "good" part of the corresponding transition system. How many "bad" states are there in this transition system?

Problem 8. The "monkey-and-banana" domain includes the fluents and actions mentioned in Problems 6 and 7, and also a banana and the actions of climbing the box and grasping the banana. Give a semi-adequate formalization of this domain in STRIPS. Show that the solution produced by CCALC and discussed at the beginning of the semester corresponds to a path in the corresponding transition system.

Problem 9. There is a robot with two grippers. It can carry a ball in each. The available actions are picking up and dropping balls and moving between rooms. Describe this domain in STRIPS.