1 Introduction

This article is a collection of definitions related to action languages. It does not provide a comprehensive discussion of the subject, and does not contain a complete bibliography, but it may be useful as a reference in future publications.

Action languages are formal models of parts of the natural language that are used for talking about the effects of actions. Central to this method of formalizing actions is the concept of a *transition system*, which is defined in Section 2. Then we consider action languages of two kinds. *Action description languages* serve for describing transition systems; examples of such languages are given in Sections 3–6. *Action query languages* serve for stating assertions about a transition system; after some preparations in Section 7, examples of such languages are given in Sections 8–10. Reference marks (like [99]) refer to the notes and examples in Section 11.

2 Transition Systems

Definition 1 An *action signature* consists of three nonempty sets: a set $\mathbf{V}$ of *value names*, a set $\mathbf{F}$ of *fluent names*, and a set $\mathbf{A}$ of *action names*.

Intuitively, any "fluent" (represented by a symbol from $\mathbf{F}$) has a specific "value" (represented by a symbol from $\mathbf{V}$) in any "state of the world"[1]. An "action" (represented by a symbol from $\mathbf{A}$), if executed in some state, leads to a "resulting" state.[2] Generally, the resulting state is not uniquely determined by the initial state and the action; in this sense, an action can be nondeterministic.[3]

The following definition makes this precise:

Definition 2 A *transition system* of an action signature $\langle \mathbf{V}, \mathbf{F}, \mathbf{A} \rangle$ consists of

(i) a set S,

(ii) a function V from $\mathbf{F} \times S$ into $\mathbf{V}$, and

(iii) a subset R of $S \times \mathbf{A} \times S$.

The elements of S are called *states*. We say that $V(P, s)$ is the *value* of P in s. The states s' such that $\langle s, A, s' \rangle \in R$ are the possible *results of the execution* of the action A in the state s. We say that A is *executable* in s if there is at least one state s' with this property, and that A is *deterministic* in s if there is at most one such s'.

A transition system can be thought of as a labeled directed graph. Every state s is represented by a vertex labeled with the function $P \mapsto V(P, s)$ from fluent names to value names. Every triple $\langle s, A, s' \rangle \in R$ is represented by an edge leading from s to s' and labeled A.[4]

To discuss the concurrent execution of actions in the transition system framework, we consider an action signature whose action names are truth-valued functions defined on a set $\mathbf{E}$ of "elementary action names":

$$\mathbf{A} = \{\mathsf{f}, \mathsf{t}\}^{\mathbf{E}}. \tag{1}$$

For such a signature, to execute A means, intuitively, to execute concurrently all elementary actions E such that $A(E) = \mathsf{t}$. (All elementary actions are assumed to have the same duration.) Accordingly, we will identify an elementary action name E with the action name that maps E to t and maps all other elementary action names to f. For any elementary action names $E_1, \ldots, E_n$ $(n \geq 2)$, $E_1 + \cdots + E_n$ stands for the action name that maps $E_1, \ldots, E_n$ to t and maps all other elementary action names to f. The action name that maps every elementary action name to f is denoted by 0 or by *Wait*.[5]

Definition 3 An action signature $\langle \mathbf{V}, \mathbf{F}, \mathbf{A} \rangle$ is *propositional* if its value names are the truth values of classical logic:

$$\mathbf{V} = \{\mathsf{f}, \mathsf{t}\}.$$

A *literal* of a propositional action signature is an expression of the form P or $\neg P$, where P is a fluent name. A transition system is *propositional* if its signature is propositional.

In the rest of this paper, we limit attention to propositional transition systems. That is to say, we only discuss the effects of actions on propositional fluents.[6]

3 STRIPS as an Action Description Language

The language of STRIPS operators, as defined in [Fikes and Nilsson, 1971], is not an action description language in the sense of this paper, because the objects that STRIPS operators operate on are "world models" rather than states of a transition system. The idea behind the modification of STRIPS described below is due to Pednault [1987].

Consider a fixed propositional action signature $\langle \{\mathsf{f}, \mathsf{t}\}, \mathbf{F}, \mathbf{A} \rangle$. The following two definitions characterize the syntax and semantics of STRIPS.

Definition 4 In STRIPS,

(i) an *operator* is a pair $\langle F, X \rangle$, where F is a propositional combination of fluent names (the *precondition*), and X is a consistent set of literals (the *effects*),

(ii) an *action description* is a function from action names to operators.

Recall that a (propositional) *interpretation* of a set of symbols is a function that maps this set into $\{\mathsf{f}, \mathsf{t}\}$.

Definition 5 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) $V(P, s) = s(P)$,

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

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

The last line of the definition of $s'(P)$ represents the STRIPS solution to the frame problem.

4 Action Description Language $\mathcal{A}$

Pednault [1987] observed that the expressive power of STRIPS can be enhanced by allowing the effects of an operator to be "conditional." This idea leads to the language $\mathcal{A}$, which is essentially the propositional fragment of Pednault's ADL.

Consider a fixed propositional action signature $\langle\{\mathsf{f},\mathsf{t}\},\mathbf{F},\mathbf{A}\rangle$.

Definition 6 In the language $\mathcal{A}$,

(i) a *proposition*[7] is an expression of the form

$$A \textbf{ causes } L \textbf{ if } F \tag{2}$$

where A is an action name, L a literal (called the *head*) and F a conjunction of literals (possibly empty),[8]

(ii) an *action description* is a set of propositions.

We will denote the empty conjunction by $True$. In (2), the part **if** F can be dropped if F is $True$.

In the following definition, we identify a propositional interpretation with the set of literals that satisfy this interpretation.

Definition 7 Let D be an action description in $\mathcal{A}$. 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) $V(P, s) = s(P)$,

(iii) R is the set of all triples $\langle s, A, s'\rangle$ such that

$$E(A, s) \subseteq s' \subseteq E(A, s) \cup s \tag{3}$$

where $E(A, s)$ stands for the set of the heads L of all propositions A **causes** L **if** F in D such that s satisfies F.

Intuitively, $E(A, s)$ lists the effects of the action A executed in the state s. The first inclusion in (3) requires that all these effects hold in the resulting state s'. The second inclusion says that otherwise the values of fluents in s' are the same as in s; this is how the semantics of $\mathcal{A}$ solves the frame problem.[9,10,11]

For any A and s, there may be at most one state s' satisfying (3), so that any action described in $\mathcal{A}$ is deterministic in every state.

5 Action Description Language $\mathcal{B}$

The language $\mathcal{B}$ is an extension of $\mathcal{A}$ in which one can describe actions with indirect effects. This is achieved by introducing propositions of a new kind, "static laws." A static law is understood in $\mathcal{B}$ as a rule for inferring literals.[12]

Given a fixed propositional action signature $\langle\{\mathsf{f},\mathsf{t}\},\mathbf{F},\mathbf{A}\rangle$, we define:

Definition 8 In the language $\mathcal{B}$,

(i) a *static law* is an expression of the form

$$L \textbf{ if } F \tag{4}$$

where L a literal (called the *head*) and F a conjunction of literals,

(ii) a *dynamic law* is an expression of the form

$$A \textbf{ causes } L \textbf{ if } F \tag{5}$$

where A is an action name, and L, F are as above,

(iii) an *action description* is a set of static and dynamic laws.

In (5), the part **if** F can be dropped if F is *True*.

The semantics of $\mathcal{B}$ (Definition 10 below) uses the following terminology and notation.

Definition 9 A set s of literals is *closed* under a set Z of static laws if s includes the head L of every static law L **if** F in Z such that s satisfies F. The set $Cn_Z(s)$ of *consequences* of s under Z is the smallest set of literals that contains s and is closed under Z.

Definition 10 Let D be an action description in $\mathcal{B}$. The transition system $\langle S, V, R\rangle$ *described* by D is defined as follows:

(i) S is the set of all interpretations of $\mathbf{F}$ that are closed under the static laws of D,

(ii) $V(P, s) = s(P)$,

(iii) R is the set of all triples $\langle s, A, s'\rangle$ such that

$$s' = Cn_Z(E(A, s) \cup (s \cap s')) \tag{6}$$

where Z is the set of all static laws of D, and $E(A, s)$ is the set of the heads L of all dynamic laws A **causes** L **if** F of D such that s satisfies F.[13]

The argument of Cn_Z in (6) is the union of the set $E(A, s)$ of the "direct effects" of A with the set $s \cap s'$ of facts that are "preserved by inertia." The application of Cn_Z adds the "indirect effects" of A to this union.[14]

6 Action Description Language $\mathcal{C}$

The action language $\mathcal{C}$ is similar to $\mathcal{B}$ in that its propositions are classified into static laws and dynamic laws.[15] In some ways, it is more expressive (although, strictly speaking, it is not a superset of $\mathcal{B}$). First, nondeterministic actions and the concurrent execution of actions can be conveniently described in $\mathcal{C}$. Second, inertia is not a "built-in feature" in the semantics of $\mathcal{C}$; we are free to decide for each fluent whether or not to postulate inertia for it.[16]

The view of causality adopted in $\mathcal{C}$ distinguishes between asserting that a certain fact "holds" and making the stronger assertion that it "is caused" (or "has a causal explanation").[17]

Consider an action signature of the form $\langle\{\mathsf{f},\mathsf{t}\}, \mathbf{F}, \{\mathsf{f},\mathsf{t}\}^{\mathbf{E}}\rangle$ whose set $\mathbf{E}$ of elementary action names is disjoint from $\mathbf{F}$.

Definition 11 In the language $\mathcal{C}$,

(i) a *state formula* is a propositional combination of fluent names,

(ii) a *formula* is a propositional combination of fluent names and elementary action names,

(iii) a *static law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \tag{7}$$

where F and G are state formulas,

(iv) a *dynamic law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \textbf{ after } U \tag{8}$$

where F and G are state formulas and U is a formula,

(v) an *action description* is a set of static and dynamic laws.

The formula F in a static law (7) or a dynamic law (8) is called its *head*. In (7) and (8), the part **if** G can be dropped if G is $True$.

Definition 12 Let D be an action description in $\mathcal{C}$. The transition system $\langle S, V, R\rangle$ *described* by D is defined as follows:

(i) S is the set of all interpretations s of $\mathbf{F}$ such that, for every static law **caused** F **if** G in D, s satisfies F if s satisfies G,

(ii) $V(P, s) = s(P)$,

(iii) R is the set of all triples $\langle s, A, s'\rangle$ such that s' is the only interpretation of $\mathbf{F}$ which satisfies the heads of all

- static laws **caused** F **if** G in D for which s' satisfies G, and
- dynamic laws **caused** F **if** G **after** U in D for which s' satisfies G and $s \cup A$ satisfies U.

Intuitively, the static and dynamic laws included in (iii) are those that are "applicable" to the transition from s to s' caused by executing A. According to (iii), the execution of A in a state s may lead to a state s' if s' is completely characterized by the applicable laws.[18]

Definition 13 In the language $\mathcal{C}$,

(i) an expression of the form

$$U \textbf{ causes } F \textbf{ if } G \tag{9}$$

where U is a propositional combination of elementary action names and F, G are state formulas, stands for the dynamic law

$$\textbf{caused } F \textbf{ if } True \textbf{ after } G \wedge U,$$

(ii) an expression of the form

$$\textbf{inertial } F$$

where F is a state formula, stands for the dynamic law

$$\textbf{caused } F \textbf{ if } F \textbf{ after } F,$$

(iii) an expression of the form

$$\textbf{inertial } F_1, \ldots, F_n$$

where $F_1, \ldots, F_n$ $(n > 1)$ are state formulas, stands for n dynamic laws **inertial** F_i $(i = 1, \ldots, n)$.[19]

In (9), the part **if** G can be dropped if G is *True*.

Definition 14 In the language $\mathcal{C}$,

(i) an expression of the form

$$\textbf{always } F$$

where F is a state formula, stands for the static law

$$\textbf{caused } \mathit{False} \textbf{ if } \neg F,$$

(ii) an expression of the form

$$\textbf{nonexecutable } U \textbf{ if } F \tag{10}$$

where U is a propositional combination of elementary action names and F is a state formula, stands for the dynamic law

$$\textbf{caused } \mathit{False} \textbf{ after } F \wedge U.$$

In (10), the part **if** F can be dropped if F is *True*.

The abbreviation **always** allows us to encode "qualification constraints" in the sense of [Lin and Reiter, 1994].

Other abbreviations:

Definition 15 In the language $\mathcal{C}$,

(i) an expression of the form

$$\textbf{default } F \textbf{ if } G \tag{11}$$

where F and G are state formulas, stands for the static law

$$\textbf{caused } F \textbf{ if } F \wedge G,$$

(ii) an expression of the form

$$U \textbf{ may cause } F \textbf{ if } G \tag{12}$$

where U is a propositional combination of elementary action names and F, G are state formulas, stands for the dynamic law

$$\textbf{caused } F \textbf{ if } F \textbf{ after } G \wedge U.$$

In (11) and (12), the part **if** G can be dropped if G is *True*.

We use **default** to say that a certain condition holds whenever it has not been prevented by executing an action.[20] The abbreviation **may cause** helps us specify the effects of some nondeterministic actions.[21,22]

7 Axioms, Queries and Histories

The syntax of an action query language is defined by two classes of syntactic expressions: "axioms" and "queries." The semantics of the language is defined by specifying, for every transition system T, every set Γ of axioms, and every query Q, whether Q is a "consequence" of Γ in T; in symbols, whether $\Gamma \models_T Q$.

The semantics of an action query language can be often defined in terms of "histories," which are simply paths in the directed graph representing T. More precisely:

Definition 16 A *history* of a transition system $\langle S, V, R\rangle$ of *length* n is a sequence

$$s_0, A_1, s_1, \ldots, A_n, s_n$$

($s_0, \ldots, s_n \in S$, $A_1, \ldots, A_n \in \mathbf{A}$, $n \geq 0$) such that

$$\langle s_{i-1}, A_i, s_i\rangle \in R \qquad (1 \leq i \leq n).$$

8 Action Query Language $\mathcal{P}$

The language $\mathcal{P}$ is designed for describing temporal projection: its axioms are observations made in the current state of the world, and its queries are hypothetical assertions about the effects of the sequential execution of actions in this state.

Consider a fixed propositional action signature $\langle\{\mathsf{f}, \mathsf{t}\}, \mathbf{F}, \mathbf{A}\rangle$.

Definition 17 In the language $\mathcal{P}$,

(i) an *axiom* is an expression of the form

$$\mathbf{now}\ L$$

where L is a literal,

(ii) a *query* is an expression of the form

$$\mathbf{necessarily}\ F\ \mathbf{after}\ A_1; \ldots; A_k$$

where F is a propositional combination of fluent names, and $A_1, \ldots, A_k$ ($k \geq 1$) are action names.[23,24,25]

Definition 18 Let $T = \langle S, V, R\rangle$ be a transition system. In the language $\mathcal{P}$,

(i) a history $s_0, A_1, s_1, \ldots, A_n, s_n$ of T *satisfies* an axiom **now** L if the interpretation $P \mapsto V(P, s_0)$ satisfies L,

(ii) a query **necessarily** F **after** $A_1; \ldots; A_k$ is a *consequence* of a set Γ of axioms in T if, for every history of T of the form $s_0, A_1, s_1, \ldots, A_k, s_k$ that satisfies all axioms in Γ, the interpretation $P \mapsto V(P, s_k)$ satisfies F.[26]

9 Action Query Language $\mathcal{Q}$

In the language $\mathcal{Q}$, both axioms and queries are about a sequence of actions that have been already executed.

We begin with an action signature of the form $\langle\{\mathsf{f}, \mathsf{t}\}, \mathbf{F}, \{\mathsf{f}, \mathsf{t}\}^{\mathbf{E}}\rangle$. In addition, we select a nonnegative integer n—the length of the sequence of actions under consideration. Both the syntax and semantics of the language are affected by the choice of n, and to stress this fact we will denote the language by $\mathcal{Q}_n$. The symbols $t_0, \ldots, t_n$ used in axioms and queries of $\mathcal{Q}_n$ refer, intuitively, to the times when the consecutive actions began and ended.

Definition 19 In the language $\mathcal{Q}_n$,

(i) *atoms* are expressions of the form

$$E \textbf{ occurs at } t_i$$

where E is an elementary action name and $i < n$, and expressions of the form

$$P \textbf{ holds at } t_i$$

where P is a fluent name,

(ii) an *axiom* is an atom possibly preceded by $\neg$,

(iii) a *query* is a propositional combination of atoms.

Thus axioms in $\mathcal{Q}_n$ are a special case of queries. Expressions of the form F **holds at** t_i, where F is a propositional combination of fluent names, can be used as abbreviations; for instance,

$$P \wedge \neg Q \textbf{ holds at } t_0$$

stands for

$$(P \textbf{ holds at } t_0) \wedge \neg(Q \textbf{ holds at } t_0).$$

Definition 20 Let T be a transition system. In the language $\mathcal{Q}_n$, the satisfaction relation between a history of T of length n and a query is defined recursively, as follows:

(i) a history $s_0, A_1, s_1, \ldots, A_n, s_n$ *satisfies* an atomic query E **occurs at** t_i if $E \in A_{i+1}$,

(ii) a history $s_0, A_1, s_1, \ldots, A_n, s_n$ *satisfies* an atomic query P **holds at** t_i if s_i satisfies P,

(iii) for non-atomic queries, *satisfaction* is defined by the truth tables of propositional logic.

A query Q is a *consequence* of a set Γ of axioms in T if every history of T of length n that satisfies all axioms in Γ satisfies Q also.[27,28,29]

10 Action Query Language $\mathcal{R}$

The language $\mathcal{R}$ combines the expressive possibilities of the languages $\mathcal{P}$ and $\mathcal{Q}$: we can use it to talk both about the events that have actually taken place and about the hypothetical actions that could have been executed at any stage of this process.[30]

As before, we consider a fixed action signature of the form $\langle\{\mathsf{f},\mathsf{t}\}, \mathbf{F}, 2^{\mathbf{E}}\rangle$ and a fixed nonnegative integer n.

Definition 21 In the language $\mathcal{R}_n$, *atoms* and *axioms* are defined as in $\mathcal{Q}_n$ (Definition 19). A *hypothetical query* is an expression of the form

$$\textbf{necessarily } F \textbf{ after } A_1; \ldots; A_k \textbf{ at } t_i$$

where F is a propositional combination of fluent names, and $A_1, \ldots, A_k$ $(k \geq 1)$ are action names. A *query* is a propositional combination of atoms or a hypothetical query.

Thus the availability of hypothetical queries is the only new syntactic feature of $\mathcal{R}$ in comparison with $\mathcal{Q}$.

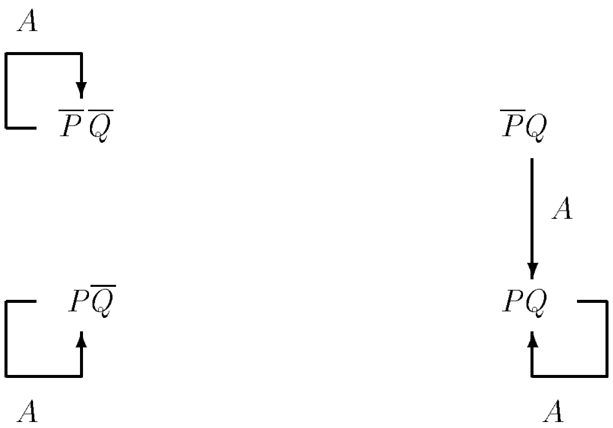

Figure 1: A transition system.

Definition 22 Let $T = \langle S, V, R\rangle$ be a transition system. In the language $\mathcal{R}_n$, the satisfaction relation between a history of T of length n and a propositional combination of atoms is defined in the same way as in $\mathcal{Q}_n$ (Definition 20). A history $s_0, A_1, s_1, \ldots, A_n, s_n$ *satisfies* a hypothetical query

$$\textbf{necessarily } F \textbf{ after } A'_1; \ldots; A'_k \textbf{ at } t_i$$

if, for every history of T of the form $s'_0, A'_1, s'_1, \ldots, A'_k, s'_k$ such that $s'_0 = s_i$, the interpretation $P \mapsto V(P, s'_k)$ satisfies F. A query Q is a *consequence* of a set Γ of axioms in T if every history of T of length n that satisfies all axioms in Γ satisfies Q also.[31,32]

11 Notes and Examples

1. We usually think of a state of the world as instantaneous. Alternatively, we can imagine that a state has a duration, but the fluents represented by the symbols from $\mathbf{F}$ do not change their values within a state. In this case, value names may represent functions of time that describe how the world changes while it remains in the same state.

2. Nothing is assumed about the values of fluents during the execution of an action.

3. The execution of a nondeterministic action can be thought of as a two-step process: first the agent selects one of the possible ways to execute the action, and then the action is executed as determined by this decision.

4. Consider, for instance, the action signature whose value names are f (false) and t (true), whose fluent names are P are Q, and whose only action name is A. Figure 1 represents a transition system of this signature that has 4 states; the literals placed at every vertex show the values of P and Q in the corresponding state. The action A is executable and deterministic in every state. The execution of A makes P true if the "precondition" Q is satisfied, and it never affects Q.

5. A domain description of an arbitrary action signature $\langle \mathbf{V}, \mathbf{F}, \mathbf{A}\rangle$ can be identified, in a natural way, with a domain description of the signature

$\langle \mathbf{V}, \mathbf{F}, \{\mathsf{f}, \mathsf{t}\}^{\mathbf{A}} \rangle$ in which the only executable actions are elementary. In this sense, we can restrict attention to the action signatures satisfying (1) without loss of generality.

6. For examples of action languages applicable to nonpropositional transition systems, see [Lifschitz, 1996] and [Giunchiglia *et al.*, 1997].

7. These are essentially "effect propositions" in the sense of [Gelfond and Lifschitz, 1993]. "Value propositions" are introduced here in Section 8 to define a separate language—the action query language $\mathcal{P}$.

8. Alternatively, any propositional combination of fluent names could have been allowed as F. This would not have made the language more expressive: having converted F to its disjunctive normal form $F_1 \vee \cdots \vee F_k$, we can replace

$$A \textbf{ causes } L \textbf{ if } F$$

with k propositions

$$A \textbf{ causes } L \textbf{ if } F_i$$

$(i = 1, \ldots, k)$ that have conjunctions of literals after **if**.

9. Example: in the language $\mathcal{A}$, the domain description consisting of the single proposition

$$A \textbf{ causes } P \textbf{ if } Q$$

describes the transition system shown in Figure 1.

10. Definition 7 differs from the semantics of $\mathcal{A}$ as it was defined originally [Gelfond and Lifschitz, 1993] in how it interprets pairs of propositions of the form

$$A \textbf{ causes } P \textbf{ if } F_1,$$
$$A \textbf{ causes } \neg P \textbf{ if } F_2.$$

The new semantics makes A nonexecutable in the states that satisfy both F_1 and F_2. According to the old semantics, a pair of propositions like this, with satisfiable $\{F_1, F_2\}$, makes the domain description "inconsistent."

11. STRIPS, as defined Section 3, can be embedded into $\mathcal{A}$ in the following way. Let D be an action description in STRIPS. The corresponding action description in the language $\mathcal{A}$ (extended as in Note 8) consists of the propositions

$$A \textbf{ causes } L$$

for every action name A and every effect L of $D(A)$, and

$$A \textbf{ causes } P \textbf{ if } \neg F,$$
$$A \textbf{ causes } \neg P \textbf{ if } \neg F$$

for every action name A, where P is a fixed fluent name and F stands for the precondition of $D(A)$.

12. The analogy between causal laws and inference rules plays an important role in [Baral *et al.*, 1995] and [McCain and Turner, 1995]. The language $\mathcal{B}$ is essentially a subset of the action language from [Turner, 1997].

13. This definition turns into Definition 7 when Z is empty, in view of the fact that, for any interpretations s and s', (3) is equivalent to

$$s' = E(A, s) \cup (s \cap s').$$

Generally, actions described in $\mathcal{B}$ can be nondeterministic.

14. Example: the spring-loaded suitcase with two latches [Lin, 1995] can be described in $\mathcal{B}$ by the propositions

$$\begin{array}{l} \mathit{Open}\ \mathbf{if}\ \mathit{Up}(L1) \wedge \mathit{Up}(L2) \\ \mathit{Toggle}(l)\ \mathbf{causes}\ \neg\,\mathit{Up}(l)\ \mathbf{if}\ \mathit{Up}(l) \\ \mathit{Toggle}(l)\ \mathbf{causes}\ \mathit{Up}(l)\ \mathbf{if}\ \neg\,\mathit{Up}(l) \end{array}$$

($l \in \{L1, L2\}$). The corresponding transition system has 7 states, and each of the actions $\mathit{Toggle}(l)$ is executable and deterministic in each of these states. The action $\mathit{Toggle}(L1)$, performed in the state that satisfies

$$\neg\,\mathit{Up}(L1),\ \ \mathit{Up}(L2),\ \ \neg\mathit{Open},$$

changes the values of $\mathit{Up}(L1)$ (a direct effect) and of Open (an indirect effect).

15. The action language $\mathcal{C}$ is proposed in [Giunchiglia and Lifschitz, 1998] on the basis of the ideas of [McCain and Turner, 1997].

16. For instance, a paper cup with a hole in it is empty unless it has just been filled with liquid. The level of liquid in this cup can be viewed as a "non-inertial fluent." Other examples of this kind can be found in [Giunchiglia and Lifschitz, 1998], p. 628.

17. This view has been developed by several authors, including Pearl [1988], Geffner [1990], Lin [1995] and McCain and Turner [1995]. In application to the language $\mathcal{B}$, this understanding of causal laws suggests using the syntax

$$\mathbf{caused}\ L\ \mathbf{if}\ \mathbf{caused}\ F$$

instead of L **if** F.

18. This is similar to the "principle of universal causation" [McCain and Turner, 1997], according to which, in a causally possible world history, every fact that obtains is caused. In the semantics of $\mathcal{C}$, this principle is applied to the world histories that consist of just two time instants—before and after executing an action—and is restricted to the facts obtained in the second of these time instants.

19. For instance, the domain description in $\mathcal{C}$ similar to the example from Note 9 above is

$$\begin{array}{l} A\ \mathbf{causes}\ P\ \mathbf{if}\ Q \\ \mathbf{intertial}\ P, \neg P, Q, \neg Q. \end{array} \tag{13}$$

These propositions describe the transition system shown in Figure 2. With the set $\mathbf{E}$ consisting of one elementary action name A, there are two actions in this example: doing A and doing nothing (0). Each of the actions is executable and deterministic in every state.

The edges labeled A in Figure 2 are the same as in Figure 1. This fact illustrates the general theorem about embedding $\mathcal{A}$ into $\mathcal{C}$ according to which any domain description D in $\mathcal{A}$ can be translated into $\mathcal{C}$ by adding the dynamic laws **inertial** L for all literals L (Proposition 1 from [Giunchiglia and Lifschitz, 1998]).

The suitcase example (Note 14 above) is formalized in $\mathcal{C}$ as follows:

$$\begin{array}{l} \mathbf{caused}\ \mathit{Open}\ \mathbf{if}\ \mathit{Up}(L1) \wedge \mathit{Up}(L2) \\ \mathit{Toggle}(l)\ \mathbf{causes}\ \neg\,\mathit{Up}(l)\ \mathbf{if}\ \mathit{Up}(l) \\ \mathit{Toggle}(l)\ \mathbf{causes}\ \mathit{Up}(l)\ \mathbf{if}\ \neg\,\mathit{Up}(l) \\ \mathbf{inertial}\ \mathit{Up}(l), \neg\,\mathit{Up}(l) \\ \mathbf{inertial}\ \mathit{Open}, \neg\,\mathit{Open} \end{array}$$

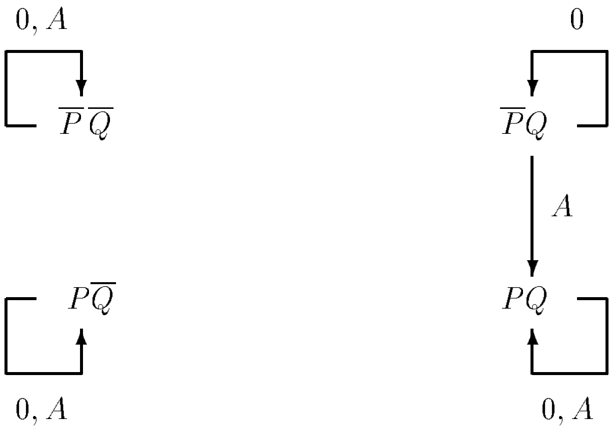

Figure 2: Transition system described by (13).

($l \in \{L1, L2\}$). There are 4 actions:

$$\mathit{Wait},\ \ \mathit{Toggle}(L1),\ \ \mathit{Toggle}(L2),\ \ \mathit{Toggle}(L1) + \mathit{Toggle}(L2),$$

and each of them is executable and deterministic in every state.

20. For instance, a formalization of the example from Note 16 might include the static law

$$\textbf{default}\ \mathit{Empty}\ \textbf{if}\ \mathit{Hole}$$

(instead of **inertial** *Empty*).

As an indication of how $\mathcal{C}$ differs from $\mathcal{B}$, note that, in the semantics of $\mathcal{B}$, the rule

$$L\ \textbf{if}\ L$$

is trivial—it does not express the same idea as

$$\textbf{caused}\ L\ \textbf{if}\ L$$

in $\mathcal{C}$.

21. For instance, replacing the first line of (13) with

$$A\ \textbf{may cause}\ P\ \textbf{if}\ Q$$

would add one more edge to Figure 2: a loop at $\overline{P}Q$ labeled A. In the modified transition system, A is nondeterministic in the state $\overline{P}Q$.

22. It should be noted that $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ share some features which are not presupposed by the general action language framework. First, the syntax of each of the three languages assumes that the underlying action signature is propositional (see Note 6). Second, in a transition system described in any of these languages, states are functions from fluent names to value names. But the set of states can be defined in other ways. As a result, a transition system may include a pair of states s_1, s_2 that are different from each other although $V(P, s_1) = V(P, s_2)$ for all $P \in \mathbf{F}$. This possibility is sometimes important [Giunchiglia and Lifschitz, 1995]. Third, the syntax of an action description language may include variables [Dung, 1993]. They are similar to the metavariable l used in our formalizations of the suitcase

example above (Notes 14 and 19) but, unlike l, they are treated as part of the language. Finally, the semantics of an action description language may allow an action description to represent a class of transition systems, rather that just one system.

23. Axioms and queries of the language $\mathcal{P}$ are similar to "value propositions" as defined in [Gelfond and Lifschitz, 1993]. The use of **now** in axioms stresses the fact that an axiom is an observation about the *current* state of the world. Axioms are allowed to contain literals, but not complex combinations of fluents, because we think of an observation as producing the value of a single fluent—not the truth value of a complex condition. (Technically, it would be easy to accomodate all propositional combinations of fluent names in axioms.) The use of **necessarily** in queries stresses the fact that a query refers to *all* states that may result after the execution of the given sequence of (generally, nondeterministic) actions.

24. The language $\mathcal{P}$, unlike the corresponding fragment of the language from [Gelfond and Lifschitz, 1993], is limited to temporal projection. Accordingly, assertions about the initial values of fluents can only be used as axioms, not as queries; assertions about the values of fluents after the execution of actions can only be used as queries, not as axioms. The problem with using expressions like F **after** A as axioms that the authors of [Gelfond and Lifschitz, 1993], regrettably, glossed over is that such an expression can be understood in several ways.

One possible interpretation is that A was actually executed in some initial state, and F was true after that. Assumptions of this kind can be expressed by axioms in the languages $\mathcal{Q}$ and $\mathcal{R}$ described in Sections 9 and 10; such assumptions were investigated in [Baral *et al.*, 1997]. Note that, under this interpretation, the pair of axioms

$$\begin{array}{l} F_1 \textbf{ after } A_1, \\ F_2 \textbf{ after } A_2 \end{array}$$

should be treated as inconsistent whenever $A_1 \neq A_2$, because these two axioms give conflicting information about the action executed in the initial state.

Alternatively, F **after** A can be thought of as a hypothetical assumption: if A were executed in the current state then F would be true—necessarily, or possibly, or actually. The last version of the hypothetical interpretation does not seem to have a clear intuitive meaning for some nondeterministic actions. What do we express, for instance, by the assumption that, if Jack were to toss a coin, it would come up heads?

25. In a useful extension of $\mathcal{P}$, queries are formed from fluent names using propositional connectives and the following additional formation rule: if Q is a query and A is an action name then **necessarily** Q **after** A is a query. A query of the form

$$\textbf{necessarily } (\textbf{necessarily } Q \textbf{ after } A_2) \textbf{ after } A_1$$

can be abbreviated as

$$\textbf{necessarily } Q \textbf{ after } A_1; A_2$$

and similarly for longer sequences of actions. The queries of $\mathcal{P}$ are a special case when the **necessarily** ... **after** construct is applied in the process of query formation only at the end, and at least once.

In this extension of $\mathcal{P}$, **possibly** Q **after** A can be understood as shorthand for

$$\neg(\textbf{necessarily } \neg Q \textbf{ after } A),$$

and **executable** A can be viewed as shorthand for

$$\textbf{possibly } \mathit{True} \textbf{ after } A.$$

The last abbreviation differs from the use of **nonexecutable** in the action description language $\mathcal{C}$ (Definition 14) in that it serves for talking about the executability of A in a given transition system, rather than specifying a transition system in which A is not executable under certain conditions. In the spirit of the discussion of executable plans in [McCain and Turner, 1998], this abbreviation can be generalized as follows:

$$\textbf{executable } A_1, \ldots, A_k$$

stands for

$$(\textbf{executable } A_1) \wedge \bigwedge_{i=2}^{k} \textbf{necessarily } (\textbf{executable } A_i) \textbf{ after } A_1; \ldots; A_{i-1}.$$

26. For instance, the query **necessarily** $P \wedge Q$ **after** A is a consequence of the axiom **now** Q in the transition system from Note 4 (Figure 1).

27. Consider, for instance, the transition system shown in Figure 2. The query

$$(P \textbf{ holds at } t_0) \vee (A \textbf{ occurs at } t_0) \tag{14}$$

is a consequence of the axiom

$$P \textbf{ holds at } t_1 \tag{15}$$

in $\mathcal{Q}_n$ for any $n \geq 1$.

28. Instead of requiring that the information about the non-occurence of actions be represented in axioms explicitly, we could have allowed deriving it "by the closed world assumption." In such a modification of $\mathcal{Q}$, expressions of the form $\neg(E$ **occurs at** $t_i)$ would not be permitted as axioms. A query of this form might be treated as a consequence of a set Γ of axioms whenever Γ does not include the corresponding atom E **occurs at** t_i. In the example from Note 27, with the semantics of $\mathcal{Q}$ modified in this way, not only (14) would be a consequence of (15) but also the stronger query

$$(P \textbf{ holds at } t_0) \wedge \neg(A \textbf{ occurs at } t_0).$$

For the same transition system, the pair of axioms

$$\neg P \textbf{ holds at } t_0, \;\; P \textbf{ holds at } t_1 \tag{16}$$

would be inconsistent. The consequence relation of this language is nonmonotonic.

Alternatively, we can permit the occurrence of actions that are not explicitly postulated as long as the set of these actions is minimal subject to the constraints expressed by the axioms (compare Definition 3.4 from [Baral *et al.*, 1997]). Under this "soft" version of the closed world assumption, axioms (16) are consistent and entail A **occurs at** t_0.

29. Instead of assuming a fixed temporal order among the time instants t_i, we could have represented it by expressions like t_i **precedes** t_j that could be used both in axioms and in queries [Baral *et al.*, 1997]. This enhancement of $\mathcal{Q}$ would allow us to formalize narratives in which the order of events is not completely specified.

30. A language with these expressive capabilities is described in [Baral *et al.*, 1997].

31. Consider for instance, the transition system for the suitcase example with concurrent actions (Note 19), and let the axioms be

$$\begin{array}{ll} \neg Up(L1) \textbf{ holds at } t_0, & \\ Toggle(l) \textbf{ occurs at } t_0 & (l \in \{L1, L2\}), \\ \neg Open \textbf{ holds at } t_1 & \end{array}$$

—both latches were toggled simultaneously in a state in which the first of them was down, and in the resulting state the suitcase was closed. These axioms entail, both in $\mathcal{Q}$ and in $\mathcal{R}$, that the second latch was initially up:

$$Up(L2) \textbf{ holds at } t_0.$$

In $\mathcal{R}$, the axioms entail also the counterfactual assertion that the suitcase would have opened if the first latch alone were toggled:

$$\textbf{necessarily } Open \textbf{ after } Toggle(L1) \textbf{ at } t_0.$$

32. The extensions and modifications discussed in Notes 25, 28 and 29 can be applied to the language $\mathcal{R}$ as well.

Acknowledgements

We are grateful to Tim Fernando, Enrico Giunchiglia, Yi Mao, Norman McCain and Emilio Remolina for comments on a draft of this paper. The work of the second author was partially supported by National Science Foundation under grant IRI-9732744.

References

[Baral *et al.*, 1995] Chitta Baral, Michael Gelfond, and Alessandro Provetti. Representing actions I: Laws, observations and hypotheses. In *Working Notes of the Symposium on Extending Theories of Actions*, 1995.

[Baral *et al.*, 1997] Chitta Baral, Michael Gelfond, and Alessandro Provetti. Reasoning about actions: laws, observations and hypotheses. *Journal of Logic Programming*, 31:201–244, 1997.

[Dung, 1993] Phan Minh Dung. Representing actions in logic programming and its applications in database updates. In *Logic Programming: Proceedings of the Tenth Int'l Conf. on Logic Programming*, pages 222–238, 1993.

[Fikes and Nilsson, 1971] Richard Fikes and Nils Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[Geffner, 1990] Hector Geffner. Causal theories for nonmonotonic reasoning. In *Proc. AAAI-90*, pages 524–530, 1990.

[Gelfond and Lifschitz, 1993] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[Giunchiglia and Lifschitz, 1995] Enrico Giunchiglia and Vladimir Lifschitz. Dependent fluents. In *Proc. IJCAI-95*, pages 1964–1969, 1995.

[Giunchiglia and Lifschitz, 1998] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI-98*, pages 623–630, 1998.

[Giunchiglia *et al.*, 1997] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Representing action: indeterminacy and ramifications. *Artificial Intelligence*, 95:409–443, 1997.

[Lifschitz, 1996] Vladimir Lifschitz. Two components of an action language. In *Working Papers of the Third Symposium on Logical Formalizations of Commonsense Reasoning*, 1996.

[Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation*, 4:655–678, 1994.

[Lin, 1995] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. IJCAI-95*, pages 1985–1991, 1995.

[McCain and Turner, 1995] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *Proc. IJCAI-95*, pages 1978–1984, 1995.

[McCain and Turner, 1997] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. AAAI-97*, pages 460–465, 1997.

[McCain and Turner, 1998] Norman McCain and Hudson Turner. Satisfiability planning with causal theories. In Anthony Cohn, Lenhart Schubert, and Stuart Shapiro, editors, *Proc. Sixth Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 212–223, 1998.

[Pearl, 1988] Judea Pearl. Embracing causality in default reasoning. *Artificial Intelligence*, 35:259–271, 1988.

[Pednault, 1987] Edwin Pednault. Formulating multi-agent, dynamic world problems in the classical planning framework. In Michael Georgeff and Amy Lansky, editors, *Reasoning about Actions and Plans*, pages 47–82. Morgan Kaufmann, San Mateo, CA, 1987.

[Turner, 1997] Hudson Turner. Representing actions in logic programs and default theories: a situation calculus approach. *Journal of Logic Programming*, 31:245–298, 1997.