TLP **3** (3): 329–376, May 2003. © 2003 Cambridge University Press
DOI: 10.1017/S1471068402001606 Printed in the United Kingdom

Handling defeasibilities in action domains

YAN ZHANG

School of Computing and Information Technology, University of Western Sydney, Locked Bag 1797, Penrith South DC, NSW 1797, Australia
(*e-mail:* yan@cit.uws.edu.au)

Abstract

Representing defeasibility is an important issue in common sense reasoning. In reasoning about action and change, this issue becomes more difficult because domain and action related defeasible information may conflict with general inertia rules. Furthermore, different types of defeasible information may also interfere with each other during the reasoning. In this paper, we develop a prioritized logic programming approach to handle defeasibilities in reasoning about action. In particular, we propose three action languages $\mathcal{AT}^0$, $\mathcal{AT}^1$ and $\mathcal{AT}^2$ which handle three types of defeasibilities in action domains named defeasible constraints, defeasible observations and actions with defeasible and abnormal effects respectively. Each language with a higher superscript can be viewed as an extension of the language with a lower superscript. These action languages inherit the simple syntax of $\mathcal{A}$ language but their semantics is developed in terms of transition systems where transition functions are defined based on prioritized logic programs. By illustrating various examples, we show that our approach eventually provides a powerful mechanism to handle various defeasibilities in temporal prediction and postdiction. We also investigate semantic properties of these three action languages and characterize classes of action domains that present more desirable solutions in reasoning about action within the underlying action languages.

KEYWORDS: reasoning about action, temporal reasoning, logic programming, common sense reasoning, priority, defeasibility

1 Introduction

Representing defeasibility is an important issue in common sense reasoning. In reasoning about action, this issue becomes more difficult because domain and action related defeasible information may conflict with general inertia rules – that are necessary to specify things that persist with respect to actions and usually defeasible as well. Furthermore, different types of defeasible information may also interfere with each other during the reasoning. Therefore, most previous action theories usually ignored such defeasible information in problem domains. However, recent work on causality reveals that in many situations defeasibility plays an important role in temporal prediction and postdiction and ignoring this issue may cause difficulties in deriving correct solutions in reasoning about action.

Let us consider the Switch-Power domain that was first addressed in Zhang (1999), where two domain constraints were taken into account:

if the switch is on, then the light is usually on; (1)

if there is no power, then the light is not on. (2)

Intuitively, the first constraint is defeasible from our common sense. For instance, even if the switch is on, the light might not be on if there is no power, or there is a problem in the circuit, and so on. But if this constraint is not expressed as a defeasible rule, we may have a difficulty in our reasoning. Suppose we simply represent the above two constraints as logical implications $Switch \supset On$ and $\neg Power \supset \neg On$ respectively. If the initial state is $\{On, Power, Switch\}$ and the robot is asked to perform an action *Cut-Power* with effect $\neg Power$ (e.g. a fire alarm leads the robot to perform this action). Clearly, *Cut-Power* will cause a direct effect $\neg Power$, and then from constraint $Switch \supset On$ and $\neg Power \supset \neg On$, an indirect effect $\neg Switch$ is derived. Obviously, this effect is not quite reasonable from our intuition as cutting off the power should be irrelevant to the switch's position.

People may argue that the above problem is due to the duality of logical implication (i.e. $A \supset B \equiv \neg B \supset \neg A$). Now suppose we adopt McCain and Turn's causal theory (McCain and Turner, 1995), where constraints (1) and (2) are represented as inference rules $Switch \Rightarrow On$ and $\neg Power \Rightarrow \neg On$ respectively[1]. Then under the same initial state as above, it turns out that action *Cut-Power* becomes unexecutable because the effect $\neg Power$ together with rule $\neg Power \Rightarrow \neg On$ contradicts fact *On* which is derivable from fact *Switch* and rule $Switch \Rightarrow On$. This is not a desirable solution either.

The above example just illustrates one type of defeasibility – defeasible constraints, which causes difficulties in reasoning about action. In fact, there are other types of defeasible information, such as defeasible observations and actions with defeasible and abnormal effects, that also significantly influence temporal prediction and postdiction. Although the problem of defeasibilities has been investigated by some researchers recently (Baral and Lobo, 1997; Geffner, 1997; Jabłonowski *et al*., 1996; Zhang, 1999), none of the previous proposals is completely satisfactory in terms of representing and handling different types of defeasibilities in temporal reasoning (we will discuss this issue in section 7).

In this paper, we address three basic types of defeasible information related to temporal prediction and postdiction where incomplete information is allowable: defeasible constraints, defeasible observations and actions with defeasible and abnormal effects. Our goal is to handle these three types of defeasibilities in reasoning about action under a unified framework of logic programming.

The issue of representing action in logic programming languages is not new. It was explored by some researchers previously (Eshghi and Kowalski, 1989). However, probably Gelfond and Lifschitz's work (Gelfond and Lifschitz, 1993) was the first time to make a major progress in this direction. By introducing a simple action language $\mathscr{A}$, Gelfond and Lifschitz's action formulation was able to deal with both temporal prediction and postdiction, while properties of actions were characterized by translating action language $\mathscr{A}$ into the language of extended logic programs

[1] Informally, $A \Rightarrow B$ represents a semantics like "if A then B", from which we cannot derive $\neg B \Rightarrow \neg A$. See McCain and Turner (1995) for detail.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

(Gelfond and Lifschitz, 1991). In other words, in Gelfond and Lifschitz's formulation, extended logic program was used as an implementation of the high level action language $\mathscr{A}$.

It has been recognized that logic programming can not only be used as the implementation of a high level action language, but also can be used as a basis for providing a formal semantics of the high level language (Baral and Lobo, 1997). In this paper, we further demonstrate that prioritized logic programming has a great flexibility to serve as a semantic basis to develop high level action languages that handle various information conflicts in reasoning about action. The paper is organized as follows. Section 2 briefly reviews the concept of prioritized logic programs. Section 3 proposes a simple action language $\mathscr{AT}^0$ which can represent actions in domains with defeasible constraints. The syntax of $\mathscr{AT}^0$ is similar to that of $\mathscr{A}$ style action languages. A transition system is proposed to provide a formal semantics of $\mathscr{AT}^0$, where a corresponding prioritized logic program is employed as a basis for defining such a transition system. Section 4 then extends action language $\mathscr{AT}^0$ to $\mathscr{AT}^1$ so that it can represent defeasible observations and shows how it handles the problem of temporal postdiction under the occurrence of defeasible observations. Section 5 further generalizes $\mathscr{AT}^1$ to action language $\mathscr{AT}^2$ to represent actions with defeasible and abnormal effects. Section 6 then investigates various properties of action languages $\mathscr{AT}^0, \mathscr{AT}^1$ and $\mathscr{AT}^2$ and characterize specific classes of action domains that may present desirable solutions in reasoning about action. Section 7 discusses related work, and finally, section 8 concludes the paper with some remarks.

2 Prioritized Logic Programs (PLPs): an overview

We first introduce the extended logic program and its answer set semantics developed by Gelfond and Lifschitz (1991). A language $\mathscr{L}$ of extended logic programs is determined by its object constants, function constants and predicate constants. *Terms* are built as in the corresponding first order language; *atoms* have the form $P(t_1, \ldots, t_n)$, where t_i $(1 \leqslant i \leqslant n)$ is a term and P is a predicate constant of arity n; a *literal* is either an atom $P(t_1, \ldots, t_n)$ or a negative atom $\neg P(t_1, \ldots, t_n)$. A *rule* is an expression of the form:

$$L_0 \leftarrow L_1, \ldots, L_m, not L_{m+1}, \ldots, not L_n, \tag{3}$$

where each L_i $(0 \leqslant i \leqslant n)$ is a literal. L_0 is called the *head* of the rule, while $\{L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n\}$ is called the *body* of the rule. Obviously, the body of a rule could be empty. We also allow the head of a rule to be empty. In this case, the rule with an empty head is called *constraint*. A term, atom, literal, or rule is *ground* if no variable occurs in it. An *extended logic program* Π is a collection of rules. Π is *ground* if each rule in Π is ground.

To evaluate an extended logic program, Gelfond and Lifschitz proposed an answer set semantics for extended logic programs. Let Π be a ground extended logic program not containing *not* and *Lit* the set of all ground literals in the language of Π. An *answer set* of Π is the smallest subset S of *Lit* such that (i) for any rule

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

$L_0 \leftarrow L_1, \ldots, L_m$ from Π, if $L_1, \ldots, L_m \in S$, then $L_0 \in S$; and (ii) if S contains a pair of complementary literals, then $S = Lit$. Now let Π be a ground arbitrary extended logic program. For any subset S of Lit, let Π^S be the logic program obtained from Π by deleting (i) each rule that has a formula *not* L in its body with $L \in S$, and (ii) all formulas of the form *not* L in the bodies of the remaining rules[2]. We define that S is an *answer set* of Π iff S is an answer set of Π^S.

For a non-ground extended logic program Π, we usually view a rule in Π containing variables to be the set of all ground instances of this rule formed from the set of ground literals in the language. The collection of all these ground rules forms the *ground instantiation* Π' of Π. Then a set of ground literals is an answer set of Π if and only if it is an answer set of Π'. It is easy to see that an extended logic program may have one, more than one, or no answer set at all.

A *Prioritized Logic Program* (PLP) $\mathscr{P}$ is a triple $(\Pi, \mathscr{N}, <)$, where Π is an extended logic program, $\mathscr{N}$ is a naming function mapping each rule in Π to a name, and $<$ is a strict partial ordering on names. The partial ordering $<$ in $\mathscr{P}$ plays an essential role in the evaluation of $\mathscr{P}$. We also use $\mathscr{P}(<)$ to denote the set of $<$-relations of $\mathscr{P}$. Intuitively $<$ represents a preference of applying rules during the evaluation of the program. In particular, if $\mathscr{N}(r) < \mathscr{N}(r')$ holds in $\mathscr{P}$, rule r would be preferred to apply over rule r' during the evaluation of $\mathscr{P}$ (i.e. rule r is more preferred than rule r'). Consider the following classical example represented in our formalism:

$\mathscr{P}_1 = (\Pi, \mathscr{N}, <)$:
$N_1 : Fly(x) \leftarrow Bird(x), \text{not } \neg Fly(x),$
$N_2 : \neg Fly(x) \leftarrow Penguin(x), \text{not } Fly(x),$
$N_3 : Bird(Tweety) \leftarrow,$
$N_4 : Penguin(Tweety) \leftarrow,$
$N_2 < N_1.$

Obviously, rules N_1 and N_2 conflict with each other as their heads are complementary literals, and applying N_1 will defeat N_2, and *vice versa*. However, as $N_2 < N_1$, we would expect that rule N_2 is preferred to apply first and then defeat rule N_1 so that the desired solution $\neg Fly(Tweety)$ can be derived.

Definition 1
Let Π be a ground extended logic program and r a ground rule of the form (3) (r does not necessarily belong to Π). Rule r is *defeated* by Π iff Π has an answer set and for any answer set S of Π, there exists some $L_i \in S$, where $m + 1 \leqslant i \leqslant n$.

Now our idea of evaluating a PLP is as follows. Let $\mathscr{P} = (\Pi, \mathscr{N}, <)$. If there are two rules r and r' in Π and $\mathscr{N}(r) < \mathscr{N}(r')$, r' will be ignored in the evaluation of $\mathscr{P}$, *only if* keeping r in Π and deleting r' from Π will result in a defeat of r'. By eliminating all such potential rules from Π, $\mathscr{P}$ is eventually reduced to an extended logic program in which the partial ordering $<$ has been removed. Our evaluation for $\mathscr{P}$ is then based on this *reduced* extended logic program.

[2] We also call Π^S the Gelfond–Lifschitz transformation of Π in terms of S.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Similarly to the case of extended logic programs, the evaluation of a PLP will be based on its ground form. We say that a PLP $\mathscr{P}' = (\Pi', \mathscr{N}', <')$ is the *ground instantiation* of $\mathscr{P} = (\Pi, \mathscr{N}, <)$ if (1) Π' is the ground instantiation of Π; and (2) $\mathscr{N}'(r_1') <' \mathscr{N}'(r_2') \in \mathscr{P}'(<')$ if and only if there exist rules r_1 and r_2 in Π such that r_1' and r_2' are ground instances of r_1 and r_2 respectively and $\mathscr{N}(r_1) < \mathscr{N}(r_2) \in \mathscr{P}(<)$. Under this definition, however, we require a restriction on a PLP, since not every PLP's ground instantiation presents a consistent information with respect to the original PLP. Consider a PLP as follows:

$N_1 : P(f(x)) \leftarrow notP(x)$,
$N_2 : P(f(f(x))) \leftarrow notP(f(x))$,
$N_2 < N_1$.

If the only constant in the language is 0, then the set of ground instances of N_1 and N_2 includes rules like:

$N_1' : P(f(0)) \leftarrow notP(0)$,
$N_2' : P(f(f(0))) \leftarrow notP(f(0))$,
$N_3' : P(f(f(f(0)))) \leftarrow notP(f(f(0)))$,
$\ldots$,

It is easy to see that N_2' can be viewed as an instance for both N_1 and N_2. Therefore, the ordering $<'$ among rules $N_1', N_2', N_3', \ldots$ is no longer a strict partial ordering because of $N_2' <' N_2'$. Obviously, we need to exclude this kind of programs in our context. On the other hand, we also want to avoid a situation like $\cdots <' N_3' <' N_2' <' N_1'$ in the ground prioritized logic program because this $<'$ indicates that there is no most preferred rule in the program.

Given a PLP $\mathscr{P} = (\Pi, \mathscr{N}, <)$. We say that $\mathscr{P}$ is *well formed* if there is no rule r' that is an instance of two different rules r_1 and r_2 in Π and $\mathscr{N}(r_1) < \mathscr{N}(r_2) \in \mathscr{P}(<)$. Then it is not difficult to observe that the following fact holds.

Fact: If a PLP $\mathscr{P} = (\Pi, \mathscr{N}, <)$ is well formed, then in its ground instantiation $\mathscr{P}' = (\Pi', \mathscr{N}', <'), <'$ is a partial ordering and every non-empty subset of Π' has a least element with respect to $<'$.

Due to the above fact, in the rest of this paper, we will only consider well formed PLP programs in our discussions, and consequently, the evaluation for an arbitrary PLP $\mathscr{P} = (\Pi, \mathscr{N}, <)$ will be based on its ground instantiation $\mathscr{P}' = (\Pi', \mathscr{N}', <')$. Therefore, in our context a ground prioritized (or extended) logic program may contain infinite number of rules. In this case, we will assume that this ground program is the ground instantiation of some program that only contains finite number of rules. In the rest of the paper, whenever there is no confusion, we will only consider ground prioritized (extended) logic programs without explicit declaration.

Definition 2
(Zhang and Foo, 1997a) Let $\mathscr{P} = (\Pi, \mathscr{N}, <)$ be a prioritized logic program. $\mathscr{P}^<$ is a *reduct* of $\mathscr{P}$ with respect to $<$ if and only if there exists a sequence of sets Π_i $(i = 0, 1, \ldots)$ such that:

1. $\Pi_0 = \Pi$;

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

2. $\Pi_i = \Pi_{i-1} - \{r_1, r_2, \ldots \mid$ (a) there exists $r \in \Pi_{i-1}$ such that for every j $(j = 1, 2, \ldots)$, $\mathscr{N}(r) < \mathscr{N}(r_j) \in \mathscr{P}(<)$ and $r_1, r_2, \ldots$ are defeated by $\Pi_{i-1} - \{r_1, r_2, \ldots\}$, and (b) there are no rules $r', r'', \ldots \in \Pi_{i-1}$ such that $N(r_j) < N(r')$, $N(r_j) < N(r''), \ldots$ for some j $(j = 1, 2, \ldots)$ and $r', r'', \ldots$ are defeated by $\Pi_{i-1} - \{r', r'', \ldots\}\}$;
3. $\mathscr{P}^< = \bigcap_{i=0}^{\infty} \Pi_i$.

In Definition 2, $\mathscr{P}^<$ is an extended logic program obtained from Π by eliminating some rules from Π. In particular, if $\mathscr{N}(r) < \mathscr{N}(r_1)$, $\mathscr{N}(r) < \mathscr{N}(r_2), \ldots$, and $\Pi_{i-1} - \{r_1, r_2, \ldots\}$ defeats $\{r_1, r_2, \ldots\}$, then rules $r_1, r_2, \ldots$ will be eliminated from Π_{i-1} if no *less preferred rule* can be eliminated (i.e. conditions (a) and (b)). This procedure is continued until a fixed point is reached. It should be noted that condition (b) in the above definition is necessary because without it some unintuitive results may be derived. For instance, consider $\mathscr{P}_1$ again, if we add additional preference $N_3 < N_2$ in $\mathscr{P}_1$, then using a modified version of Definition 2 without condition (b),

$$\{Fly(Tweety) \leftarrow Bird(Tweety), not \neg Fly(Tweety),$$
$$Bird(Tweety) \leftarrow,$$
$$Penguin(Tweety) \leftarrow\}$$

is a reduct of $\mathscr{P}_1$, from which we will conclude that Tweety can fly.

Theorem 1
Every PLP has a reduct.

Definition 3
(Zhang and Foo, 1997a) Let $\mathscr{P} = (\Pi, \mathscr{N}, <)$ be a PLP and *Lit* the set of all ground literals in the language of $\mathscr{P}$. For any subset S of *Lit*, S is an *answer set* of $\mathscr{P}$ iff S is an answer set for some reduct $\mathscr{P}^<$ of $\mathscr{P}$.

Example 1
Using Definitions 2 and 3, it is easy to conclude that $\mathscr{P}_1$ has a unique reduct as follows:

$$\mathscr{P}_1^< = \{\neg Fly(Tweety) \leftarrow Penguin(Tweety), \text{ not } Fly(Tweety),$$
$$Bird(Tweety) \leftarrow,$$
$$Penguin(Tweety) \leftarrow\},$$

from which we obtain the following answer set of $\mathscr{P}_1$:

$$S = \{Bird(Tweety), Penguin(Tweety), \neg Fly(Tweety)\}.$$

Now we consider another program $\mathscr{P}_2$:

$$N_1 : A \leftarrow,$$
$$N_2 : B \leftarrow not\ C,$$
$$N_3 : D \leftarrow,$$
$$N_4 : C \leftarrow not\ B,$$
$$N_1 < N_2, N_3 < N_4.$$

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

According to Definition 2, it is easy to see that $\mathscr{P}_2$ has two reducts:

$\{A \leftarrow, \quad D \leftarrow, \quad C \leftarrow not\ B\}$, *and*
$\{A \leftarrow, \quad B \leftarrow not\ C, \quad D \leftarrow\}$.

From Definition 3, it follows that $\mathscr{P}_2$ has two answer sets: $\{A, C, D\}$ and $\{A, B, D\}$.

3 $\mathscr{AT}^0$: representing actions in domains with defeasible constraints

In this section, we develop an action language $\mathscr{AT}^0$ which is able to handle domains with defeasible constraints. The syntax of language $\mathscr{AT}^0$ is inspired by $\mathscr{A}$ family languages, and a transition system will be developed to provide the semantics of $\mathscr{AT}^0$ where a corresponding prioritized logic program is employed to define the transition function.

3.1 Syntax of $\mathscr{AT}^0$

The language $\mathscr{AT}^0$ has two disjoint sets of names called *actions* and *fluents*. We will use A, A_1, A_2, ... to denote action names, and $F, F_1, F_2, \ldots$ to denote fluent names. We define a *fluent expression* to be a fluent name possibly preceded by a negation sign $\neg$.

A *value proposition* is an expression of the form:

$$L \textbf{ after } A_1, \ldots, A_l, \tag{4}$$

where L is a fluent expression and $A_1, \ldots, A_l$ are action names. A value proposition is also called an *initial proposition* if no action name occurs in it:

$$\textbf{initially } L. \tag{5}$$

A *causal proposition* is an expression of the form:

$$L \textbf{ is caused if } L_1, \ldots, L_m \textbf{ with absence } L_{m+1}, \ldots, L_n, \tag{6}$$

where L, $L_1, \ldots, L_n$ are fluent expressions. This is so-called *defeasible constraint* whose intuitive meaning is that L is caused to be true if $L_1, \ldots, L_m$ are true and $L_{m+1}, \ldots, L_n$ are not present. As a special case, (6) is reduced to a non-defeasible causal rule if no absent fluent expression is mentioned:

$$L \textbf{ is caused if } L_1, \ldots, L_m \tag{7}$$

An *action effect proposition* is an expression of the form:

$$A \textbf{ causes } L \textbf{ if } L_1, \ldots, L_k, \tag{8}$$

where A is an action name and $L_1, \ldots, L_k$ are fluent expressions. (8) means that if preconditions $L_1, \ldots, L_k$ of A are true, then action A causes L to be true. Note the difference between (7) and (8) while no action is involved in the former.

Now we define a *domain description* $\mathscr{D}$ of $\mathscr{AT}^0$ to be a finite set of initial propositions, causal propositions and action effect propositions. It should be noted that here we do not include value propositions of the form (4) into a domain description since at the moment we restrict our formulation only to deal with

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

prediction reasoning while a value proposition (4) is only used as a *query statement* in the language[3]. The following example shows how language $\mathscr{AT}^0$ is used to describe an action domain.

Example 2
Let us consider the Switch-Power domain mentioned in section 1 again. The domain includes two constraints: (a) if the switch is on, then the light is usually on; (b) if there is no power, then the light is not on. We treat the first constraint as a defeasible causal rule. We also suppose that initially the light is on, there is power and the switch is on. An action *Cut-Power* is then performed. It has been shown that the previous approaches have difficulties to deal with this example due to a lack of expressibility of defeasible constraints (Zhang, 1999). This action scenario can be described by specifying a domain description $\mathscr{D}(Switch\text{-}Power)$ of $\mathscr{AT}^0$ as follows. Firstly, $\mathscr{D}(Switch\text{-}Power)$ contains the following three initial propositions:

initially On,
initially $Power$,
initially $Switch$.

$\mathscr{D}(Switch\text{-}Power)$ also includes the following two causal propositions to capture the domain constraints presented above:

On **is caused if** $Switch$ **with absence** $\neg On$,
$\neg On$ **is caused if** $\neg Power$.
Finally, $\mathscr{D}(Switch\text{-}Power)$ has one action effect proposition:

$Cut\text{-}Power$ **causes** $\neg Power$.

3.2 Semantics of $\mathscr{AT}^0$

Similarly to the idea presented in Gelfond and Lifschitz (1993), we define a transition system to provide a formal semantics for $\mathscr{AT}^0$. However, instead of developing an independent transition system for the language, our transition function is defined based on the PLP. This is because the PLP has a powerful mechanism of solving conflicts between defeasible information, which, from our observation, is difficult to handle in the traditional transition system approach.

3.2.1 Translating $\mathscr{AT}^0$ into PLP

We first propose a translation from a domain description $\mathscr{D}$ of $\mathscr{AT}^0$ into a PLP, and our transition function will be defined based on this translated PLP. To implement this translation, we consider a language $\mathscr{L}^P_{\mathscr{AT}^0}$ of PLPs including the following vocabulary:

— *Situation sort*: one situation constant S_0, and situation variables $s, s_1, s_2, \ldots$.
— *Action sort*: action constants $A, A_1, A_2, \ldots$, and action variables $a, a_1, a_2, \ldots$.
— *Propositional fluent sort*: fluent constants $F, F_1, F_2, \ldots$, and fluent variables $f, f_1, f_2, \ldots$.

[3] This restriction will be released in language $\mathscr{AT}^1$.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

— *Function symbol*: a binary function symbol *Result* which takes arguments of *action* and *situation*, respectively, and returns a *situation*.
— *Predicate symbols*: five binary predicate symbols *Holds*, $Caused^+, Caused^-$, $effect^+$ and $Effect^-$, all of which take arguments of *fluent* and *situation*, respectively.

In $\mathscr{L}^P_{\mathscr{AT}^0}$ situation term $Result(a,s)$ indicates the resulting situation after performing action a in s. Atom $Holds(f,s)$ (or literal $\neg Holds(f,s)$) indicates the fact that fluent f is true (or false, resp.) in situation s. Atom $Caused^+(f,s)$ (or $Caused^-(f,s)$ resp.) indicates that fluent f is *caused* to be true (or false, resp.) in situation s. *Causal rules* in $\mathscr{L}^P_{\mathscr{AT}^0}$ have the following forms[4]:

$$Caused^+(F,s) \leftarrow [\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_m,s), \quad not\ [\neg]Holds(F_{m+1},s),\ldots,not\ [\neg]Holds(F_n,s), \tag{9}$$

$$Caused^-(F,s) \leftarrow [\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_m,s), \quad not\ [\neg]Holds(F_{m+1},s),\ldots,not\ [\neg]Holds(F_n,s), \tag{10}$$

$$Holds(f,s) \leftarrow Caused^+(f,s), \tag{11}$$

$$\neg Holds(f,s) \leftarrow Caused^-(f,s). \tag{12}$$

Basically, rule (9) together with rule (11) (or (10) together with (12) resp.) says that if literals $[\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_m,s)$ are true, and there is no explicit statement saying that $[\neg]Holds(F_{m+1},s),\ldots,[\neg]Holds(F_n,s)$ are true, then fluent F is caused to be true (or false resp.) in situation s. As will be seen, in our following translation, rules (9) and (10) are actually related to the domain description $\mathscr{D}$, and hence are *domain specific*, while rules (11) and (12) act as generic rule schemas that are irrelevant to $\mathscr{D}$ and hence are *domain independent*. For simplicity, we denote

$$\Pi^c_{ind} = \{(11),(12)\}.$$

Atoms $Effect^+(f,s)$ and $Effect^-(f,s)$ are used to represent direct effects of actions. Generally, *action effect rules* have the following forms:

$$Effect^+(F,Result(A,s)) \leftarrow [\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_k,s), \tag{13}$$

$$Effect^-(F,Result(A,s)) \leftarrow [\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_k,s), \tag{14}$$

$$Holds(f,s) \leftarrow Effect^+(f,s),. \tag{15}$$

$$\neg Holds(f,s) \leftarrow Effect^-(f,s). \tag{16}$$

Intuitively, rule (13) together with rule (15) (or (14) together with (16) resp.) says that if action A's preconditions $[\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_k,s)$ are true, then fluent F becomes true (or false resp.) after performing action A. Again, rules (13) and (14) are domain specific, while rules (15) and (16) represent domain independent schemas. Similarly, we denote

$$\Pi^{eff}_{ind} = \{(15),(16)\}.$$

[4] Notation $[\neg]$ means that the negation sign $\neg$ may or may not occur.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Definition 4

A PLP is called a *translation* of domain description $\mathscr{D}$ of $\mathscr{AT}^0$, denoted by $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) = (\Pi, \mathscr{N}, <)$, if it is obtained as follows:

1. Π consists of the following rules:
   Initial fact rules: For each initial proposition (5) in $\mathscr{D}$, there is a rule of the form $[\neg]Holds(F, S_0) \leftarrow$[5].
   Causal rules: for each causal proposition (6) in $\mathscr{D}$, there is a causal rule of the form (9) or (10). Two domain independent causal rules (11) and (12) are also included in this set.
   Action effect rules: for each action effect proposition (8), there is an action effect rule of the form (13) or (14). Two domain independent action effect rules (15) and (16) are also included in this set.
   Inertia rules[6]:

$$Holds(f, Result(a, s)) \leftarrow Holds(f, s), not\ \neg Holds(f, Result(a, s)), \tag{17}$$

$$\neg Holds(f, Result(a, s)) \leftarrow \neg Holds(f, s), not\ Holds(f, Result(a, s)). \tag{18}$$

2. Naming function $\mathscr{N}$ assigns a unique name to each rule in Π.
3. For each causal rule N_c and each inertia rule N_i, $<$-relation $N_c < N_i$ holds.

In $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$ specified above, Π represents initial facts, domain constraints (causal rules) and action effects corresponding to $\mathscr{D}$, and inertia rules are used to capture things that do not change with respect to actions. As we are allowed to represent defeasible causal rules while inertia rules are also defeasible, possible conflicts may occur between these two types of rules. To solve such conflicts, we specify that a causal rule is more preferred than an inertia rule. The intuition behind this is clear: generally causal rules are used to derive indirect effects of actions, and whenever there is no explicit condition to block a defeasible causal rule, this rule should be triggered to derive necessary indirect effects. This point is illustrated in Example 3.

It is also obvious that to translate a specific domain description $\mathscr{D}$ into $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, we only need to translate domain specific information such as initial propositions, causal and action effect propositions into logic program rules, while other domain independent schema rules such as Π^c_{ind}, Π^{eff}_{ind} and rules (17) and (18) are automatically embedded in every translated PLP. Formally, in a given $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, we denote a set of domain specific rules as Π^0_{spec}, and specify the set of domain independent rules as

$$\Pi^0_{ind} = \Pi^c_{ind} \cup \Pi^{eff}_{ind} \cup \Pi^i_{ind}, \tag{19}$$

where $\Pi^i_{ind} = \{(17), (18)\}$.

[5] Here $Holds(F, S_0)$ or $\neg Holds(F, S_0)$ is corresponding to whether L occurring in **initially** L is F or $\neg F$ respectively. This assumption is also adopted in the rest of this paper.

[6] Note that these two inertia rules actually represent a set of inertia rules by substituting fluent and action variables f and a with every fluent and action constants occurring in the domain, respectively.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Example 3
(Example 2 continued.) According to Definition 4, the domain description $\mathscr{D}(Switch\text{-}Power)$ presented in Example 2 can be translated into a PLP, denoted by $\mathscr{P}^{\mathscr{AT}^0}(Switch\text{-}Power) = (\Pi^0_{spec} \cup \Pi^0_{ind}, \mathscr{N}, <)$, where Π^0_{spec} consists of the following rules:
Initial fact rules:
$N_1 : Holds(On, S_0) \leftarrow$,
$N_2 : Holds(Power, S_0) \leftarrow$,
$N_3 : Holds(Switch, S_0) \leftarrow$,
Causal rules:
$N_4 : Caused^{+}(On, s) \leftarrow Holds(Switch, s), not\ \neg Holds(On, s)$,
$N_5 : Caused^{-}(On, s) \leftarrow \neg Holds(Power, s)$,
Action effect rule:
$N_6 : Effect^{-}(Power, Result(Cut\text{-}Power, s)) \leftarrow$.
Naming rules in Π^0_{ind}:
Assigning a unique name to each rule in Π^0_{ind}. That is, we assign names N_7, N_8, N_9, N_{10}, N_{11} and N_{12} to rules (11), (12), (15), (16), (17) and (18), respectively.
$<$-relations:
$N_c < N_i$, while N_c and N_i are names of any causal rule and inertia rule in Π, respectively. That is, we have $\{N_4, N_5, N_7, N_8\} < \{N_{11}, N_{12}\}$[7].

3.2.2 *Transition function, models and entailment*

To define the transition function, we first introduce the concept of state. A *state* is a collection of fluent expressions. A state is *consistent* if it does not contain a fluent F and its negative correspondent $\neg F$. We use symbols $\hat{S}_0$, $\hat{S}_1$, $\hat{S}_2$, ... to denote states. Then transition function $\mathscr{R}$ maps a state to a power set of states by some action.

Definition 5
Given a domain description $\mathscr{D}$ and its translation $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, let $\mathbb{A}$ be the set of all answer sets of $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$. The *transition function* $\mathscr{R}(A, \hat{S})$ of $\mathscr{D}$ with respect to action A and state $\hat{S}$ is defined as follows:

1. If $\mathbb{A}$ is empty or includes an inconsistent answer set of $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, then $\mathscr{R}(A, \hat{S})$ is undefined;
2. $\hat{S}_0 = \{[\neg]F \mid [\neg]Holds(F, S_0) \in Ans\}$, where $Ans \in \mathbb{A}$;
3. $\mathscr{R}(A, \hat{S}) = \{\{[\neg]F \mid [\neg]Holds(F, Result(A, S')) \in Ans$, and for any F' $[\neg]F' \in \hat{S}$ iff $[\neg]Holds(F', S') \in Ans\} \mid Ans \in \mathbb{A}\}$.

It should be noted that we define a state to be a collection of fluent expressions, that is very different from the state defined in standard $\mathscr{A}$-style action languages where states correspond to possible *physical* worlds and every fluent is either true or false in a state (Gelfond and Lifschitz, 1993). In our context, a state may not present a complete information for fluents. If a fluent is not present in a state, then this

[7] This is an abbreviation of a set of $<$-relations of the form $N_i < N_j$, where $i = 4, 5, 7, 8$ and $j = 11, 12$.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

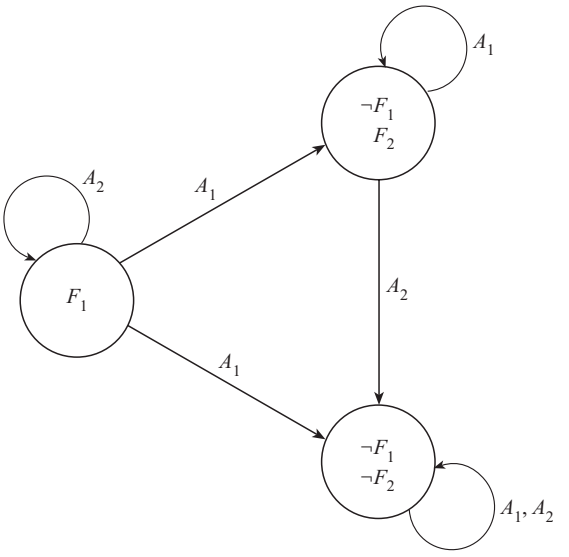

Fig. 1. State transitions.

fluent's truth value is viewed as *unknown*. Defining states in this way will bring us a flexibility to develop a formal semantics for our action theories where incomplete information related to defeasibility is admitted.

In Definition 5, $\hat{S}_0$ is called the *initial state* of $\mathscr{D}$, and $\mathscr{R}(A, \hat{S})$ represents the set of all possible states resulting from the execution of A on state $\hat{S}$. From the feature of $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, it is quite obvious that the initial state $\hat{S}_0$ is always unique. On the other hand, $\mathscr{R}(A, \hat{S})$ may include more than one state. To see how transition function $\mathscr{R}$ works, we consider a domain $\mathscr{D}$ consisting of the following propositions:

initially F_1,
F_2 **is caused if** $\neg F_1$ **with absence** $\neg F_2$,
$\neg F_2$ **is caused if** $\neg F_1$ **with absence** F_2,
A_1 **causes** $\neg F_1$,
A_2 **causes** $\neg F_2$ **if** $\neg F_1, F_2$.

Since two causal propositions conflict with each other and action A_1 is executable in the initial situation, it is not difficult to see that $\mathscr{D}$'s PLP translation $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$ has two different answer sets such that $Holds(F_2, Result(A_1, S_0))$ is in one and $\neg Holds(F_2, Result(A_1, S_0))$ is in the other. Then from Definition 5, state transitions of $\mathscr{D}$ specified by transition function $\mathscr{R}$ can be described by Figure 1, where $\{F_1\}$ is the initial state.

Let $\bar{A}$ denote an action string $A_1 \ldots A_l$ (as a special case, an empty action string is denoted as ϵ). A *structure* Ψ is a partial function from strings of actions to states whose domain is prefix closed. We refer $\Psi(\epsilon) = \hat{S}_0$, i.e. the initial state of $\mathscr{D}$. The following definition describes possible trajectories of the dynamic system (domain description) defined in $\mathscr{AT}^0$ under structure Ψ.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Definition 6
Given a structure Ψ.

1. An initial proposition of the form (5) is *satisfied* in Ψ if $L \in \Psi(\epsilon)$;
2. A causal proposition of the form (6) or an action effect proposition of the form (8) is *satisfied* in Ψ if the following conditions hold:
   - for any action string $\bar{A}$ and action constant A, if $\Psi(\bar{A})$ and $\mathscr{R}(A, \Psi(\bar{A}))$ are defined, then $\Psi(\bar{A} \cdot A) \in \mathscr{R}(A, \Psi(\bar{A}))$;
   - otherwise $\Psi(\bar{A} \cdot A)$ is undefined.

A fluent expression L is *true* in a state $\Psi(\bar{A})$ if $L \in \Psi(\bar{A})$.

Definition 7
Given a domain description $\mathscr{D}$, a structure Ψ is a *model* of $\mathscr{D}$ if all initial, causal and action effect propositions in $\mathscr{D}$ are satisfied in Ψ, and for any action string $\bar{A}$ and fluent F, F and $\neg F$ are not both true in $\Psi(\bar{A})$. We say a value proposition of the form (4): L **after** $A_1, \ldots, A_l$ is *satisfied* in Ψ if $L \in \Psi(\bar{A})$, where $\bar{A} = A_1 \ldots A_l$. We say $\mathscr{D}$ *entails* value proposition (4), denoted as $\mathscr{D} \models_{\mathscr{AT}^0} L$ **after** $A_1, \ldots, A_l$, if (4) is satisfied in all models of $\mathscr{D}$.

Example 4
Example 3 continued. From $\mathscr{D}(Switch\text{-}Power')$'s PLP translation $\mathscr{P}^{\mathscr{AT}^0}(Switch\text{-}Power)$ as shown in Example 3, it can be verified that $\mathscr{P}^{\mathscr{AT}^0}(Switch\text{-}Power)$ has a unique answer set that includes the following ground literals[8]:

$Holds(On, S_0)$,
$Holds(Power, S_0)$,
$Holds(Switch, S_0)$,
$\neg Holds(Power, Result(Cut\text{-}Power, S_0))$,
$\neg Holds(On, Result(Cut\text{-}Power, S_0))$ *and*
$Holds(Switch, Result(Cut\text{-}Power, S_0))$.

Since $\hat{S}_0 = \{On, Power, Switch\}$, we have $\mathscr{R}(Cut\text{-}Power, \hat{S}_0) = \{\hat{S}_1\}$, where $\hat{S}_1 = \{\neg On, \neg Power, Switch\}$. Now it is easy to see that structure Ψ is a model of $\mathscr{D}(Switch\text{-}Power)$, where $\Psi(\epsilon) = \hat{S}_0$ and $\Psi(Cut\text{-}Power) = \hat{S}_1$. Furthermore, according to Definition 7, we have

$\mathscr{D}(Switch\text{-}Power) \models_{\mathscr{AT}^0} \neg Power$ **after** $Cut\text{-}Power$,
$\mathscr{D}(Switch\text{-}Power) \models_{\mathscr{AT}^0} \neg On$ **after** $Cut\text{-}Power$,
$\mathscr{D}(Switch\text{-}Power) \models_{\mathscr{AT}^0} Switch$ **after** $Cut\text{-}Power$.

Now we slightly modify the domain of Switch-Power as stated in Example 2. Suppose initially the light is not on and the switch is off, and another action $Turn\text{-}On$ is also available. Then the modified domain description $\mathscr{D}(Switch\text{-}Power')$ includes the following initial propositions:

[8] Obviously, the answer set also includes many other ground literals that we are not interested in listing here.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

initially $\neg On$,
initially $\neg Switch$,

and the action effect proposition

$Turn\text{-}On$ **causes** $Switch$,

together with the effect proposition of action $Cut\text{-}Power$ and two causal propositions as given in Example 2. Ignoring the detail, we can derive the following results:

$\mathscr{D}(Switch\text{-}Power') \models_{\mathscr{AT}^0} On$ **after** $Turn\text{-}On$,
$\mathscr{D}(Switch\text{-}Power') \models_{\mathscr{AT}^0} \neg On$ **after** $Turn\text{-}On, Cut\text{-}Power$,
$\mathscr{D}(Switch\text{-}Power') \models_{\mathscr{AT}^0} Switch$ **after** $Turn\text{-}On, Cut\text{-}Power$.

4 $\mathscr{AT}^1$: combining defeasible observations into action domains

We have shown that language $\mathscr{AT}^0$ handles temporal prediction where defeasible constraints are admitted. It, however, cannot deal with temporal postdiction, e.g. within the framework of $\mathscr{AT}^0$ we cannot reason from the current state to the past under some observations. It has been realized that observations on any intermediate states (including the final state) play an important role in temporal postdiction (Jabłonowski *et al*., 1996). Here, an observation is viewed as an agent's beliefs about the domain that is either obtained from the outside world or from the agent's own assumption. In the case that an agent makes an observation under some assumption, such observation becomes defeasible because once the assumption is proved not to be true, the agent's observation should be defeated.

In this section, we extend $\mathscr{AT}^0$ to $\mathscr{AT}^1$ such that the extended language can handle temporal prediction and postdiction where both defeasible constraints and observations are admitted.

4.1 Syntax of $\mathscr{AT}^1$

The syntax of $\mathscr{AT}^1$ is the same as $\mathscr{AT}^0$'s except that $\mathscr{AT}^1$ also has an *observation proposition* of the form:

$$L \textbf{ is observed if } L_1, \ldots, L_m \textbf{ with absence } L_{m+1}, \ldots, L_n \textbf{ after } A_1, \ldots, A_l, \tag{20}$$

where $L, L_1, \ldots, L_n$ are fluent expressions, and $A_1, \ldots, A_l$ are actions. Intuitively, (20) says after actions $A_1, \ldots, A_l$ are performed sequentially, L is observed to be true if $L_1, \ldots, L_m$ are true while $L_{m+1}, \ldots, L_n$ are absent. Obviously, (20) represents a kind of defeasible information. In the case that no action occurs in (20), (20) can be written as the following form:

$$\textbf{initially } L \textbf{ is observed if } L_1, \ldots, L_m \textbf{ with absence } L_{m+1}, \ldots, L_n. \tag{21}$$

Under the language $\mathscr{AT}^1$, we define a *domain description* $\mathscr{D}$ to be a finite set of observation propositions, causal propositions and action effect propositions. $\mathscr{AT}^1$ will still have the value proposition (4) and its special case the initial proposition (5), but are only used as query statements in $\mathscr{AT}^1$.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Example 5

Let us consider a modified shooting action scenario which we name *Shooting*-1. Suppose the turkey is observed alive in the initial situation, and as there is no explicit information about whether the gun is loaded in the initial situation, the agent would assume that the gun is initially not loaded by default. After actions *Shoot* and *Wait* are successively performed, it is observed that the turkey is dead (not alive). This scenario can be naturally described by language $\mathscr{AT}^1$. In particular, we specify a domain description $\mathscr{D}$(*Shooting*-1) which has the following observation propositions:

initially *Alive* **is observed**,
initially $\neg$*Loaded* **is observed with absence** *Loaded*,
$\neg$*Alive* **is observed after** *Shoot*, *Wait*,

and an action effect proposition:

Shoot **causes** $\neg$*Alive* **if** *Loaded*.

4.2 Semantics of $\mathscr{AT}^1$

We will use a similar way as described in section 3.2 to develop a formal semantics of $\mathscr{AT}^1$ based on a transition system that is defined on the basis of the translation from a $\mathscr{AT}^1$ domain description into a PLP.

4.2.1 Translating $\mathscr{AT}^1$ into PLP

As we have mentioned earlier, the major improvement from $\mathscr{AT}^0$ to $\mathscr{AT}^1$ is that we allow defeasible observations to be presented in a domain description so that temporal postdiction becomes possible. It is quite straightforward to translate an observation proposition of the form (20) into the following logic rule:

$$[\neg]Holds(F,S) \leftarrow [\neg]Holds(F_1,S),\ldots,[\neg]Holds(F_m,S), \\ not\ [\neg]Holds(F_{m+1},S),\ldots,not\ [\neg]Holds(F_n,S), \qquad (22)$$

where $S = Result(A_l, Result(\ldots, Result(A_1, S_0)\ldots))$.

To do postdiction reasoning, for each action effect proposition in $\mathscr{D}$, we need to have some action explanation rules which will be used to derive action preconditions based on proper observations. First, if there is an action effect rule (13), the following rule explains that the fact $Holds(F, Result(A,s))$ is caused by performing action A:

$$Effect^+(F, Result(A,s)) \leftarrow Holds(F, Result(A,s)), not\ Holds(F,s), \\ not\ Caused^+(F, Result(A,s)), \qquad (23)$$

Clearly, the function of rule (23) is to identify action A's actual execution. The intuition is that if fluent F is true (or false, resp.) in situation $Result(A,s)$, and there is no explicit information saying that F is true in the previous situation s or F is caused to be true by some causal rule, then it derives that F's truth value in $Result(A,s)$ is a direct effect of action A.

If a fluent F is a direct effect of some action A, i.e. $Effect^+(F, Result(A,s))$ holds, then each precondition of A must also hold in the previous situation. That is, we

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

should have rules like:

$$[\neg]Holds(F_i, s) \leftarrow Effect^+(F, Result(A, s)), \tag{24}$$

where $i = 1, \ldots, k$ and A **causes** F **if** $[\neg]F_1, \ldots, [\neg]F_k$ is an action effect proposition in domain $\mathscr{D}$. However, it should be noted that sometimes one action may cause the same effect under different preconditions. In this case, deriving all possible action preconditions may cause contradictions. For instance, consider the following domain description $\mathscr{D}(Door)$:

initially $\neg$ *HasKey*,
DoorOpened **is observed after** *OpenDoor*,
OpenDoor **causes** *DoorOpened* **if** *HasCard*,
OpenDoor **causes** *DoorOpened* **if** *HasKey*.

In this domain, action *OpenDoor* has two independent preconditions *HasCard* and *HasKey*. If we translate this domain according to our proposal above, we will have the following logic rules:

$Holds(DoorOpened, Result(OpenDoor, S_0)) \leftarrow$,

$$\begin{aligned} Effect^+(DoorOpened, Result(OpenDoor, s)) \leftarrow \\ Holds(DoorOpened, Result(OpenDoor, s)), \\ not\ \ Holds(DoorOpened, s), \\ not\ \ Caused^+(DoorOpened, Result(OpenDoor, s)), \end{aligned}$$

$Holds(HasCard, s) \leftarrow Effect^+(DoorOpened, Result(OpenDoor, s))$, and
$Holds(HasKey, s) \leftarrow Effect^+(DoorOpened, Result(OpenDoor, s))$.

From the above logic rules, we will deduce both $Holds(HasCard, S_0)$ and $Holds(HasKey, S_0)$. But from $\mathscr{D}(Door)$, we know that $\neg HasKey$ initially holds. To avoid this kind of contradiction, instead of using rule (24), we should have a weaker rule to derive action preconditions: *whenever there is no conflict, we only deduce a minimal number of preconditions to explain an action.* Under this principle, we will change rule (24) to the following form:

$$\begin{aligned} [\neg]Holds(F_i, s) \leftarrow\ & Effect^+(F, Result(A, s)), \\ & not\ \overline{[\neg]Holds(F_i, s)}, \\ & not\ [\neg]Holds(F'_1, s), \ldots, not [\neg]Holds(F'_l, s), \end{aligned} \tag{25}$$

where $i = 1, \ldots, k$, and fluents $[\neg]F'_1, \ldots [\neg]F'_l$ occur as preconditions in all other action effect propositions of A that have the same effect[9].

The following rules represent the dual case of rules (23) and (25) corresponding to action effect rule (14):

$$\begin{aligned} Effect^-(F, Result(A, s)) \leftarrow\ & Holds(F, Result(A, s)), not\ \ Holds(F, s), \\ & not\ \ Caused^+(F, Result(A, s)), \end{aligned} \tag{26}$$

[9] $\overline{[\neg]Holds(F_i, s)}$ denotes the complementary literal of $[\neg]Holds(F_i, s)$.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

$$[\neg]Holds(F_i, s) \leftarrow Effect^{-}(F, Result(A, s)), \\ not\ \overline{[\neg]Holds(F_i, s)}, \\ not\ [\neg]Holds(F'_1, s), \ldots, not\ [\neg]Holds(F'_l, s). \quad (27)$$

Now the following definition describes the formal translation from a domain description $\mathscr{D}$ of $\mathscr{AT}^1$ into a PLP.

Definition 8

A PLP is called a *translation* of domain description $\mathscr{D}$ of $\mathscr{AT}^1$, denoted by $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) = (\Pi, \mathscr{N}, <)$, if it is obtained as follows:

1. Π consists of the following rules:
   Observation rules: for each observation proposition of (20), there is a rule of the form (22).
   Causal rules: the same as in Definition 4.
   Action effect rules: the same as in Definition 4.
   Action explanation rules: for each action effect rule (13), there are rules (23) and (25), and for each action effect rule (14), there are rules (26) and (27).
   Inertia rules: (17), (18) and:

$$Holds(f, s) \leftarrow Holds(f, Result(a, s)), not\ \neg Holds(f, s), \\ not\ Caused^{+}(f, Result(a, s)), \\ not\ Effect^{+}(f, Result(a, s)), \quad (28)$$

$$\neg Holds(f, s) \leftarrow \neg Holds(f, Result(a, s)), not\ Holds(f, s), \\ not\ Caused^{-}(f, Result(a, s)), \\ not\ Effect^{-}(f, Result(a, s)). \quad (29)$$

2. Naming function $\mathscr{N}$ assigns a unique name to each rule in Π.
3. For each observation rule N_o, causal rule N_c, action explanation rule N_{ex} and inertia rule N_i, the following $<$-relations hold:

$$N_{ex} < N_c < N_i < N_o. \quad (30)$$

Compared with Definition 4, the PLP translation specified in Definition 8 presents several new features. First, $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ allows to represent defeasible observations not only at the initial situation but also at any other situations. Second, $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ includes action explanation rules (23), (25), (26), and (27). Finally, the extra inertia rules (28) and (29) allow us to reason about fluents' truth values from the current situation to the past. That is, if a fluent f is true (or false, resp.) currently, and there is no explicit information saying that f is not true (or not false, resp.) in the previous situation, or f is caused to be true by some causal rule, or f is true (or false, resp.) as a direct effect of some action, then it derives that f is true (or false, resp.) in the previous situation.

Since both observation and action explanation rules may be defeasible, more possible conflicts may occur in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$. For instance, conflicts may not only occur between causal rules and inertia rules, but also between action explanation rules and inertia rules, observation rules and causal rules, etc.. To solve these possible

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

conflicts, the underlying $<$-relation is specified as (30). (30) presents that action explanation rules are most preferred because the execution of an action usually override defeasible causal and inertia rules, while observation rules are less preferred than inertia rules due to the intuition that a fluent's truth value normally persists if there is no explicit action or causal rule to change it.

Note that in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$, action explanation rules (23), (25), (26) and (27) are domain specific because they are specified based on action effect rules (13) and (14). On the other hand, the new inertia rules (28) and (29) are domain independent. Therefore, we can denote domain independent rules in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ as follows:

$$\Pi^1_{ind} = \Pi^c_{ind} \cup \Pi^{eff}_{ind} \cup \Pi^{i'}_{ind}, \tag{31}$$

where $\Pi^{i'}_{ind} = \Pi^i_{ind} \cup \{(28), (29)\}$. We also denote the set of domain specific rules of $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ as Π^1_{spec}.

Example 6

Example 5 continued. According to Definition 8, it is not difficult to obtain the translation of domain description $\mathscr{D}(Shooting\text{-}1)$, $\mathscr{P}^{\mathscr{AT}^1}(Shooting\text{-}1) = (\Pi^1_{spec} \cup \Pi^1_{ind}, \mathscr{N}, <)$, where Π^1_{spec} consists of the following rules:

Observation rules:

$N_1 : Holds(Alive, S_0) \leftarrow.$

$N_2 : \neg Holds(Loaded, S_0) \leftarrow not\ Holds(Loaded, S_0).$

$N_3 : \neg Holds(Alive, Result(Wait, Result(Shoot, S_0))) \leftarrow.$

Action effect rules:

$N_4 : Effect^-(Alive, Result(Shoot, s)) \leftarrow Holds(Loaded, s).$

Action explanation rules:

$$\begin{aligned} N_5 : Effect^-(Alive, Result(Shoot, s)) \leftarrow\ & \neg Holds(Alive, Result(Shoot, s)), \\ & not\ \neg Holds(Alive, s), \\ & not\ Caused^-(Alive, Result(Shoot, s)). \end{aligned}$$

$N_6 : Holds(Loaded, s) \leftarrow Effect^-(Alive, Result(Shoot, s)), not\ \neg Holds(Loaded, s).$

Naming rules in Π^1_{ind}:

Assigning a unique name to each rule in Π^1_{ind}. Therefore, we have names $N_7, N_8, N_9, N_{10}, N_{11}, N_{12}, N_{13}$, and N_{14} for rules (11), (12), (15), (16), (17), (18), (28) and (29), respectively.

$<$-relations:

$N_{ex} < N_i < N_o$. That is, we have:

$\{N_5, N_6\} < \{N_{11}, N_{12}, N_{13}, N_{14}\} < \{N_1, N_2, N_3\}$.

4.2.2 Transition function, models and entailment

Transition function $\mathscr{R}$, structures and models Ψ are defined in the same way as in $\mathscr{AT}^0$ (see section 3.2.2). We denote the entailment relation under Ψ in $\mathscr{AT}^1$ as $\models_{\mathscr{AT}^1}$. The only thing we should emphasize is that since we allow a domain description to include defeasible initial observation propositions, it is possible that

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

one initial observation proposition conflicts with the other. Therefore, different initial states $\hat{S}_0$ may be deduced from different answer sets of the corresponding translated PLP $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ of $\mathscr{D}$.

Example 7
Example 5 continued. In the shooting action scenario as described in Example 5, the question we are interested in is when the turkey died and whether the gun was actually loaded initially. This is a question about postdiction that we need to reason from the current situation to the past. After translating the domain description $\mathscr{D}(Shooting\text{-}1)$ into $\mathscr{P}^{\mathscr{AT}^1}(Shooting\text{-}1)$ as illustrated in Example 6, we obtain the following results:

$\mathscr{D}(Shooting\text{-}1) \models_{\mathscr{AT}^1} \neg Alive$ **after** $Shoot$,
$\mathscr{D}(Shooting\text{-}1) \models_{\mathscr{AT}^1}$ **initially** $Loaded$,

where the first solution says that the turkey was dead after the execution of action *Shoot*, and the second indicates that initially the gun was actually loaded, which defeats the original observation.

5 $\mathscr{AT}^2$: representing actions with defeasible and abnormal effects

It is common that in temporal reasoning under some circumstances, an action might be abnormally executed and the original expected action effect is defeated. Sometimes, an abnormal effect associated with this action may be also produced. Consider the classic shooting scenario (Sandewall, 1994) in which it is usually assumed that if the gun is loaded, then the *shoot* action causes a direct effect that the turkey is not alive. However, it is probably more natural to treat *shoot* as a defeasible action. For instance, if the bullet is dumb, the turkey would be still alive after executing action *shoot*, or it could be an abnormal effect of *shoot* if after shooting the turkey is still alive but the pigeon is dead. In this section, we try to further generalize our action language $\mathscr{AT}^1$ to $\mathscr{AT}^2$ in order to capture actions with defeasible and/or abnormal effects as described above.

5.1 Syntax of $\mathscr{AT}^2$

$\mathscr{AT}^2$ includes the same forms of observation propositions, causal propositions, and value propositions of $\mathscr{AT}^1$, but has different forms of action effect propositions. First, an *action effect proposition* of $\mathscr{AT}^2$ is of the following form:

$$A \textbf{ normally causes } L \textbf{ if } L_1, \ldots, L_k, \tag{32}$$

where A is an action and $L, L_1, \ldots, L_k$ are fluent expressions. Intuitively, this action effect proposition is defeasible since we consider that if an action is *abnormally* executed, its normal effect then cannot be produced.

Therefore, the following *action abnormal effect proposition* represents the abnormal effect of an action:

$$A \textbf{ abnormally causes } L \textbf{ if } L_1, \ldots, L_k. \tag{33}$$

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Finally, an *abnormal condition proposition* represents the condition under which an action can be considered to be abnormal:

$$A \textbf{ is abnormal if before } L_1, \ldots, L_h \textbf{ after } L_{h+1}, \ldots, L_p. \tag{34}$$

Usually, the abnormality of an action can be identified from observations on the changes of some particular fluents' truth values before and after the action execution. Hence, (34) says that if $L_1, \ldots, L_h$ are true *before* action A is executed, and $L_{h+1}, \ldots, L_p$ are true *after* action A is executed, then A is identified to be abnormal.

A *domain description* $\mathscr{D}$ of $\mathscr{AT}^2$ is a finite set of observation propositions, causal propositions, action effect propositions, abnormal action effect propositions, and abnormal condition propositions. The following example shows how we can use $\mathscr{AT}^2$ to represent domains where actions may have abnormal or/and defeasible effects.

Example 8

Let us consider a different shooting scenario named *Shooting*-2 in which action *Shoot* has a defeasible effect and it is abnormally executed if initially the gun is loaded and after performing the action, the turkey is observed still alive. Initially the gun is loaded and turkey is alive. This scenario is easy to formalize by using $\mathscr{AT}^2$. We specify a domain description $\mathscr{D}$(*Shooting*-2) that has the following observation propositions:

initially *Loaded* **is observed**,
initially *Alive* **is observed**,

a defeasible action effect proposition:

Shoot **normally causes** $\neg$ *Alive* **if** *Loaded*,

and an abnormal condition proposition:

Shoot **is abnormal if before** *Loaded* **after** *Alive*.

5.2 Semantics of $\mathscr{AT}^2$

Similarly to previous languages $\mathscr{AT}^0$ and $\mathscr{AT}^1$, we will propose a transition system to provide a formal semantics of $\mathscr{AT}^2$. Again, this transition system is defined based on a translation from a domain description of $\mathscr{AT}^2$ into a PLP.

5.2.1 Translating $\mathscr{AT}^2$ into PLP

To translate an action domain of $\mathscr{AT}^2$, we need to extend the language $\mathscr{L}^P_{\mathscr{AT}^0}$ of PLPs introduced in section 3.2 to a new language $\mathscr{L}^P_{\mathscr{AT}^2}$ of PLPs by adding following symbols:

- *Ab*: a binary predicate symbol taking arguments *action* and *situation*, respectively.
- $AbEffect^+$ and $AbEffect^-$: binary predicate symbols taking arguments *fluent* and *situation*, respectively.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Intuitively, atom $Ab(a,s)$ expresses that action a is abnormally executed at situation s, while atoms $AbEffect^{+}(f,s)$ and $AbEffect^{-}(f,s)$ are used to represent abnormal effects of actions (see the following for detail).

Considering the defeasibility of action executions, we need to modify our original action effect rules (13) and (14) presented in section 3.2 to the following forms, respectively:

$$Effect^{+}(F, Result(A,s)) \leftarrow [\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_k,s), \quad not\ \ Ab(A,s), \tag{35}$$

$$Effect^{-}(F, Result(A,s)) \leftarrow [\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_k,s), \quad not\ \ Ab(A,s). \tag{36}$$

Rule (35) (resp. (36)) says if A's preconditions $[\neg]Holds(F_1,s),\ldots,[\neg Holds(F_k,s)$ hold, and there is no explicit information stating that A is abnormally executed at situation s, then fluent F will be true (or false, resp.) in situation $Result(A,s)$ as a direct effect of A. Additionally we also need a generic schema for any action a:

$$\neg Ab(a,s) \leftarrow not\ \ Ab(a,s), \tag{37}$$

which simply expresses that if there is no explicit information saying that action A is abnormally executed at situation s, then it is assume that A is *not* abnormally executed at situation s. To simplify our following presentation, we denote

$$\Pi_{ind}^{eff'} = \Pi_{ind}^{eff} \cup \{(37)\}^{10}.$$

Consequently, action explanation rules (23) and (26) in $\mathscr{AT}^1$ are also modified as follows respectively:

$$Effect^{+}(F, Result(A,s)) \leftarrow Holds(F, Result(A,s)), not\ \ Holds(F,s), \quad not\ \ Caused^{+}(F, Result(A,s)), \quad not\ \ Ab(A,s). \tag{38}$$

$$Effect^{-}(F, Result(A,s)) \leftarrow \neg Holds(F, Result(A,s)), not\ \ \neg Holds(F,s), \quad not\ \ Caused^{-}(F, Result(A,s)), \quad not\ \ Ab(A,s). \tag{39}$$

(38) states that if fluent F is true in situation $Result(A,s)$, and there is no evidence to show that (a) F is true in the previous situation s; (b) F is caused to be true in situation $Result(A,s)$; and (c) action A is abnormal at situation s, then it is derived that F must be a direct effect of action A in situation $Result(A,s)$. (39) has a similar interpretation.

As we mentioned earlier, some actions with defeasible effects may also produce abnormal effects. Hence, we also specify action abnormal effect rules of the following

[10] Note that $\Pi_{ind}^{eff} = \{(15),(16)\}$ (see section 3.2).

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

forms:

$$AbEffect^{+}(F, Result(A,s)) \leftarrow [\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_l,s), Ab(A,s). \quad (40)$$

$$AbEffect^{-}(F, Result(A,s)) \leftarrow [\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_l,s), Ab(A,s). \quad (41)$$

$$Holds(f,s) \leftarrow AbEffect^{+}(f,s). \quad (42)$$

$$\neg Holds(f,s) \leftarrow AbEffect^{-}(f,s). \quad (43)$$

$$Ab(A,s) \leftarrow [\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_h,s), [\neg]Holds(F_{h+1}, Result(A,s)),\ldots, [\neg]Holds(F_p, Result(A,s)). \quad (44)$$

Basically, rule (40) (resp. (41)) says if conditions $[\neg]Holds(F_1,s),\ldots,[\neg]Holds(F_l,s)$ hold and A is abnormally executed, then fluent F will be true (or false resp.) as an abnormal effect of A in situation $Result(A,s)$. Rule (44), on the other hand, is a direct translation of abnormal condition proposition (34). Clearly, rules (42) and (43) are domain independent while rules (40), (41) and (44) are domain specific. Again for simplicity, we denote

$$\Pi^{ab}_{ind} = \{(42),(43)\}.$$

Now we are able to describe our translation from a domain description of $\mathscr{AT}^2$ into a PLP as follows.

Definition 9
A PLP is called a *translation* of domain description $\mathscr{D}$ of $\mathscr{AT}^2$, denoted by $\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) = (\Pi, \mathscr{N}, <)$, if it obtained as follows:

1. Π consists of following rules:
   Observation rules: the same as in Definition 8.
   Causal rules: the same as in Definition 8.
   Action effect rules: for each action effect proposition (32), there is a rule of the form (35) or (36). Three domain independent action effect rules (15), (16) and (37) are also included in this set.
   Action explanation rules: for each action effect rule of the form (35), there are rules (38) and (25), and for each action effect rule of the form (36), there are rules (39) and (27),
   Action abnormal effect rules: for each action abnormal effect proposition (33), there are rules (40)–(43), and for each abnormal condition proposition (34), there is a rule (44).
   Inertia rules: (17), (18) and:

$$Holds(f,s) \leftarrow Holds(f, Result(a,s)), not\ \neg Holds(f,s), not\ Caused^{+}(f, Result(a,s)), not\ Effect^{+}(f, Result(a,s)), not\ AbEffect^{+}(f, Result(a,s)). \quad (45)$$

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

$$\neg Holds(f,s) \leftarrow \neg Holds(f, Result(a,s)), not\ \ Holds(f,s), \\ not\ \ Caused^{-}(f, Result(a,s)), \\ not\ \ Effect^{-}(f, Result(a,s)), \\ not\ \ AbEffect^{-}(f, Result(a,s)). \quad (46)$$

2. Naming function $\mathcal{N}$ assigns a unique name to each rule in Π.
3. For each observation rule N_o, causal rule N_c, action effect rule N_{eff}, action explanation rule N_{ex}, and inertia rule N_i, there are $<$-relations (30):

$$N_{ex} < N_c < N_i < N_o$$

and

$$N_{eff} < N_c < N_i < N_o. \quad (47)$$

Note that inertia rules (45) and (46) are a natural extension of inertia rules (28) and (29) in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ respectively. The $<$-relations in $\mathscr{AT}^2$ are specified in a similar way as in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ except one more schema (47) is added. This is because in $\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D})$ action effect rules (35) and (36) are defeasible, possible conflicts between these rules and other defeasible rules (e.g. causal rules, inertia rules and observation rules) may occur indirectly through the action abnormal effect rule (44). Therefore, $<$-relations (30) (see section 4.2) and (47) are needed, as we always assume that an action's successful execution should have the highest priority.

We denote domain independent rules in $\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D})$ as follows:

$$\Pi^2_{ind} = \Pi^c_{ind} \cup \Pi^{eff'}_{ind} \cup \Pi^{ab}_{ind} \cup \Pi^{i''}_{ind}, \quad (48)$$

where $\Pi^{i''}_{ind} = \Pi^{i}_{ind} \cup \{(45), (46)\}$, and denote the set of domain specific rules by Π^2_{spec}.

5.2.2 Transition function, models and entailment

Transition function $\mathscr{R}$, structures and models Ψ of $\mathscr{AT}^2$ are defined exactly the same as in section 3.2.2. The entailment relation under Ψ in $\mathscr{AT}^2$ is denoted as $\models_{\mathscr{AT}^2}$. Again, it is observed that the initial state of a domain description $\mathscr{D}$ of $\mathscr{AT}^2$ may not be unique due to a possible conflict occurring between two defeasible initial observation propositions in $\mathscr{D}$.

Example 9

(Example 8 continued.) Given the domain description $\mathscr{D}(Shooting\text{-}2)$ as presented in Example 8, the translated PLP $\mathscr{P}^{\mathscr{AT}^2}(Shooting\text{-}2)$ is easy to be obtained according to Definition 9. Let $\mathscr{P}^{\mathscr{AT}^2}(Shooting\text{-}2) = (\Pi^2_{spec} \cup \Pi^2_{ind}, \mathcal{N}, <)$, where Π^2_{spec} consists of the following rules:

Observation rules:

$N_1 : Holds(Loaded, S_0) \leftarrow.$
$N_2 : Holds(Alive, S_0) \leftarrow.$

Action effect rule:

$$N_3 : Effect^{-}(Alive, Result(Shoot, s)) \leftarrow Holds(Loaded, s), not\ \ Ab(Shoot, s).$$

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Action explanation rule:

$$N_4 : Effect^{-}(Alive, Result(Shoot, s)) \leftarrow \neg Holds(Alive, Result(Shoot, s)).\ not\ \neg Holds(Alive, s).\ not\ Caused^{-}(Alive, Result(Shoot, s)).\ not\ Ab(Shoot, s).$$

Action abnormal effect rule:

$$N_5 : Ab(Shoot, s) \leftarrow Holds(Loaded, s), Holds(Alive, Result(Shoot, s)).$$

Naming rules in Π^2_{ind}: Assigning a unique name to each rule in Π^2_{ind}.
$<$-relations: (30) and (47).

Since the action effect rule N_3 is defeasible, it is not difficult to see that a conflict on the truth value of $Holds(Alive, Result(Shoot, S_0))$ occurs between rule N_3 and an inertia rule

$$N' : Holds(Alive, Result(Shoot, s)) \leftarrow Holds(Alive, s),\ not\ \neg Holds(Alive, Result(Shoot, s))$$

which is an instance of the generic inertia rule (17) included in Π^2_{ind}[11]. However, this conflict is solved by $N_3 < N'$. Therefore, we have the final *Result*:

$$\mathscr{D}(Shooting\text{-}2) \models_{\mathscr{AT}^2} \neg Alive \textbf{ after } Shoot,$$

from which it is concluded that action *Shoot* is not abnormally executed.

6 Characterizations of action domains

Among all action domains specified by languages $\mathscr{AT}^0$, $\mathscr{AT}^1$ and $\mathscr{AT}^2$, there are some classes of action domains that may have more desirable properties than other classes of domains. In this section, we investigate these desirable properties and characterize different action domains within languages $\mathscr{AT}^0$, $\mathscr{AT}^1$ and $\mathscr{AT}^2$, respectively.

In particular, we will explore the following questions that are important for evaluating an action formulation: (a) How can we decide whether an action domain description is consistent (has a model)? (b) Given an action domain description, how is a fluent's truth value affected by executing some action(s)? (c) Under what conditions does the reasoning within an action domain become monotonic? and (d) Is it possible to characterize a set of fluents that are *temporally definite* with respect to the underlying action domain description? For instance, if fluent F's truth value is known currently, will its truth value be also known after some action or action sequence is executed? Furthermore, we will also discuss how to improve our action formulation to handle domain dependent preferences so that they can be suited for more general cases in reasoning about action.

[11] Note that the conflict is introduced through rule N_5.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

6.1 Consistency of action domains

In this section, we consider the problem of how we can decide if a domain description is consistent (has a model). In our semantics development, the transition function $\mathscr{R}$ is defined based on a translation from the underlying domain description $\mathscr{D}$ to a PLP. Hence, it is not difficult obtain a general PLP characterization for consistent domain descriptions.

Proposition 1
Let $\mathscr{D}$ be a domain description of $\mathscr{AT}^i$ and $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ $(i = 0, 1, 2)$ the corresponding PLP translation of $\mathscr{D}$ specified previously. $\mathscr{D}$ is consistent if and only if $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ has a consistent answer set.

Proposition 1, however, cannot always be used as a feasible way to decide the consistency of a domain description because in general deciding whether a PLP has an answer set is NP-complete (Zhang, 2001)[12]. So it is important to study syntactic characterizations on different cases. Our investigation on this issue starts from language $\mathscr{AT}^0$.

6.1.1 Characterizing consistent action domains of $\mathscr{AT}^0$

Given a domain description $\mathscr{D}$ of $\mathscr{AT}^0$, we first introduce the following notions:

$\mathscr{F}^+_{Initial} = \{F \mid \textbf{initially} F \in \mathscr{D}\}$,
$\mathscr{F}^-_{Initial} = \{F \mid \textbf{initially} \neg F \in \mathscr{D}\}$,
$\mathscr{F}^+_{Effect} = \{F \mid A \textbf{ causes } F \textbf{ if } L_1, \ldots, L_m \in \mathscr{D}\}$,
$\mathscr{F}^-_{Effect} = \{F \mid A \textbf{ causes} \neg F \textbf{ if } L_1, \ldots, L_m \in \mathscr{D}\}$,
$\mathscr{F}^+_{Caused} = \{F \mid F \textbf{ is caused if} \ldots \in \mathscr{D}\}$,
$\mathscr{F}^-_{Caused} = \{F \mid \neg F \textbf{ is caused if} \ldots \in \mathscr{D}\}$.

For convenience, we use $\overline{\mathscr{F}^-_{Initial}}$ to denote the set containing those complementary elements of $\mathscr{F}^-_{Initial}$. That is,

$$\overline{\mathscr{F}^-_{Initial}} = \{\neg F \mid F \in \mathscr{F}^-_{Initial}\}.$$

Similar notations may be used for other sets, e.g. $\overline{\mathscr{F}^-_{Effect}}$, $\overline{\mathscr{F}^-_{Caused}}$, etc..

Definition 10
Given a domain description $\mathscr{D}$ of $\mathscr{AT}^0$, two fluent expressions L and L' are *mutually exclusive* in $\mathscr{D}$ if:

$L \in (\mathscr{F}^+_{Initial} \cup \overline{\mathscr{F}^-_{Initial}} \cup \mathscr{F}^+_{Effect} \cup \overline{\mathscr{F}^-_{Effect}} \cup \mathscr{F}^+_{Caused} \cup \overline{\mathscr{F}^-_{Caused}})$ implies
$L' \notin (\mathscr{F}^+_{Initial} \cup \overline{\mathscr{F}^-_{Initial}} \cup \mathscr{F}^+_{Effect} \cup \overline{\mathscr{F}^-_{Effect}} \cup \mathscr{F}^+_{Caused} \cup \overline{\mathscr{F}^-_{Caused}})$.

Intuitively, if two fluent expressions are mutually exclusive, it means that these two fluent expressions cannot be both true in any state. Based on the concept of mutual exclusion, we will provide a sufficient condition to decide the consistency of

[12] Note that deciding whether an extended logic program has an answer set is also NP-complete (Marek and Truszczyński, 1993).

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

a domain description. Before we present the result, we need to introduce further notions. For a domain description $\mathscr{D}$, we assign a unique label l to each proposition in $\mathscr{D}$ so that we can use l to refer a proposition in $\mathscr{D}$. Let l be a causal or action effect proposition in $\mathscr{D}$. That is, l has one of the following forms:

L **is caused if** $L_1, \ldots, L_m$ **with absence** $L_{m+1}, \ldots, L_n$, *or*
A **causes** L **if** $L_1, \ldots, L_m$.

We use $pre(l)$, $default(l)$ and $eff(l)$ to denote the set $\{L_1, \ldots, L_m\}$, $\{L_{m+1}, \ldots, L_n\}$ and $\{L\}$ respectively. Clearly, $default(l) = \emptyset$ if l is an action effect proposition or the causal proposition does not include absent fluent expressions. For the case that l is an initial proposition **initially** L, $pre(l) = default(l) = \emptyset$ and $eff(l) = \{L\}$.

Definition 11
Given a domain description $\mathscr{D}$ of $\mathscr{AT}^0$. Two propositions l and l' in $\mathscr{D}$ are *complementary* if one of the following conditions holds:

(i) both l and l' are causal propositions, and $eff(l)$ is a complement of $eff(l')$;
(ii) l is a causal proposition, l' is an action effect proposition, and $eff(l)$ is a complement of $eff(l')$, i.e.

l: F **is caused if** $L_1, \ldots, L_m$ **with absence** $L_{m+1}, \ldots, L_n$,
l': A **causes** $\neg F$ **if** $L'_1, \ldots, L'_k$;

(iii) both l and l' are action effect propositions of the same action, and $eff(l)$ is a complement of $eff(l')$, i.e.

l: A **causes** F **if** $L_1, \ldots, L_h$,
l': A **causes** $\neg F$ **if** $L'_1, \ldots, L'_k$.

Definition 12
Given a domain description $\mathscr{D}$ of $\mathscr{AT}^0$. $\mathscr{D}$ is *normal* if $\mathscr{D}$ satisfies all of the following conditions.

(i) $\mathscr{F}^+_{Initial} \cap \mathscr{F}^-_{Initial} = \emptyset$;
(ii) For any two causal propositions l_1 and l_2 in $\mathscr{D}$,
$\overline{eff(l_i)} \cap pre(l_i) = \emptyset$ and
$default(l_i) \cap eff(l_j) = \emptyset$ $(i, j = 1, 2)$[13];
(iii) For any pair (l, l') of complementary propositions in $\mathscr{D}$, there is a pair of fluent expressions (L, L') in $\mathscr{D}$ such that L and L' are mutually exclusive, where $L \in pre(l)$ and $L' \in pre(l')$.

Let us explain the intuition behind a normal domain description in some details. Condition (i) ensures a consistent initial state deduced from the domain description $\mathscr{D}$. Condition (ii), on the other hand, says that for each causal proposition in $\mathscr{D}$, the complement of its effect should not occur in its preconditions, and furthermore, the effect of this causal proposition does not occur in the absence component (i.e. the default part) of all other (including itself) causal propositions in $\mathscr{D}$. Finally,

[13] Note that this condition includes $default(l_i) \cap eff(l_i) = \emptyset$ $(i = 1, 2)$.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Condition (iii) represents a non-trivial restriction for complementary propositions in $\mathscr{D}$. Since two complementary propositions may cause two complementary fluents to be true in some state, this condition actually indicates that if there are two complementary propositions in the domain description, then the effects of these two propositions cannot be both true in any state. The following theorem gives a sufficient condition to guarantee a domain description to be consistent.

Theorem 2
Every normal domain description of $\mathscr{AT}^0$ is consistent.

6.1.2 Characterizing consistent action domains of $\mathscr{AT}^1$ and $\mathscr{AT}^2$

Now we try to investigate an analogue of Theorem 2 for $\mathscr{AT}^1$ and $\mathscr{AT}^2$. As $\mathscr{AT}^2$ is viewed as an extension of $\mathscr{AT}^1$, here we only need to consider domain descriptions of $\mathscr{AT}^2$. To achieve our purpose, we must modify the concept of mutual exclusion of fluent expressions in order to cover observation and abnormal action effect propositions in a domain description that are not allowed in $\mathscr{AT}^0$. In particular, we define

$\mathscr{F}^+_{\bar{A}} = \{F \mid L$ **is observed if** $\ldots$ **after** $\bar{A}\}$,
$\mathscr{F}^-_{\bar{A}} = \{\neg F \mid L$ **is observed if** $\ldots$ **after** $\bar{A}\}$.

As a special case, $\mathscr{F}^+_{\epsilon}$ is formed based on initial observation propositions of $\mathscr{D}$. Let

$\mathscr{F}^+_{Observe} = \bigcup \mathscr{F}^+_{\bar{A}}$, and
$\mathscr{F}^-_{Observe} = \bigcup \mathscr{F}^-_{\bar{A}}$,

where each action string $\bar{A}$ occurs in some observation proposition of $\mathscr{D}$. Under the context of $\mathscr{AT}^2$, we also redefine the following notions:

$\mathscr{F}^+_{Effect} = \{F \mid A$ **normally causes** F **if** $\ldots\} \cup$
$\{F \mid A$ **abnormally causes** F **if** $\ldots\}$,
$\mathscr{F}^-_{Effect} = \{F \mid A$ **normally causes** $\neg F$ **if** $\ldots\} \cup$
$\{F \mid A$ **abnormally causes** $\neg F$ **if** $\ldots\}$.

Given domain description $\mathscr{D}$, we use label l to (uniquely) refer to an observation proposition, causal proposition, action effect proposition, or action abnormal effect proposition. Then notions $pre(l)$, $default(l)$ and $eff(l)$ are defined in an obvious way. Two fluent expressions L and L' are *mutually exclusive* in $\mathscr{D}$ if

$L \in (\mathscr{F}^+_{Observe} \cup \overline{\mathscr{F}^-_{Observe}} \cup \mathscr{F}^+_{Effect} \cup \overline{\mathscr{F}^-_{Effect}} \cup \mathscr{F}^+_{Caused} \cup \overline{\mathscr{F}^-_{Caused}})$ implies
$L' \notin (\mathscr{F}^+_{Observe} \cup \overline{\mathscr{F}^-_{Observe}} \cup \mathscr{F}^+_{Effect} \cup \overline{\mathscr{F}^-_{Effect}} \cup \mathscr{F}^+_{Caused} \cup \overline{\mathscr{F}^-_{Caused}})$.

Finally, we should also modify the definition of complementary propositions as follows.

Definition 13
Given a domain description $\mathscr{D}$ of $\mathscr{AT}^1$ or $\mathscr{AT}^2$. Two propositions l and l' in $\mathscr{D}$ are *complementary* if one of the following conditions holds:

(i) both l and l' are causal propositions and $eff(l)$ is a complement of $eff(l')$;

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

(ii) l is a causal proposition and l' is an action effect or abnormal effect proposition and $eff(l)$ is a complement of $eff(l')$;

(iii) both l and l' are action effect propositions of the same action where $eff(l)$ is a complement of $eff(l')$, i.e.

l: A **normally causes** F **if** $L_1, \ldots, L_h$,
l': A **normally causes** $\neg F$ **if** $L'_1, \ldots, L'_k$;

(iv) both l and l' are action abnormal effect propositions of the same action and $eff(l)$ is a complementary of $eff(l')$, i.e.

l: A **abnormally causes** F **if** $L_1, \ldots, L_h$,
l': A **abnormally causes** $\neg F$ **if** $L'_1, \ldots, L'_k$.

The following definition then extends the concept of normal domain description to $\mathcal{AT}^1$ and $\mathcal{AT}^2$.

Definition 14
Given a domain description $\mathcal{D}$ of $\mathcal{AT}^1$ or $\mathcal{AT}^2$. $\mathcal{D}$ is *normal* if $\mathcal{D}$ satisfies all of the following conditions.

(i) For any action string $\bar{A}$ occurring in observation propositions of $\mathcal{D}$, $\mathcal{F}^+_{\bar{A}} \cap \mathcal{F}^-_{\bar{A}} = \emptyset$;

(ii) For any two observation or causal propositions l_1 and l_2 in $\mathcal{D}$, $\overline{eff(l_i)} \cap pre(l_i) = \emptyset$ and $default(l_i) \cap eff(l_j) = \emptyset$ $(i, j = 1, 2)$;

(iii) For any pair (l, l') of complementary propositions in $\mathcal{D}$, there is a pair of fluent expressions (L, L') in $\mathcal{D}$ such that L and L' are mutually exclusive, where $L \in pre(l)$ and $L' \in pre(l')$.

Theorem 3
Every normal domain description of $\mathcal{AT}^1$ or $\mathcal{AT}^2$ is consistent.

6.2 Cause of change on fluents' truth values

In the rest of the paper, our discussion will focus on consistent action domains. First, the following theorem illustrates a basic property of any (consistent) action domain of $\mathcal{AT}^0$ showing that a fluent's truth value can only be affected by some action effect proposition or causal proposition.

Theorem 4
Let $\mathcal{D}$ be a consistent domain description and $\mathcal{P}^{\mathcal{AT}^0}(\mathcal{D})$ the corresponding PLP translation of $\mathcal{D}$. Then the following results hold:

(i) If $\mathcal{D} \models_{\mathcal{AT}^0} F$ **after** $\bar{A} \cdot A$ and $\mathcal{D} \not\models_{\mathcal{AT}^0} F$ **after** $\bar{A}$, then
$\mathcal{P}^{\mathcal{AT}^0}(\mathcal{D}) \models Effect^+(F, Result(A, S))$ or $\mathcal{P}^{\mathcal{AT}^0}(\mathcal{D}) \models Caused^+(F, Result(A, S))$,
where
$S = Result(A_l, \ldots, Result(A_1, S_0)\ldots)$ and $\bar{A} = A_1 \ldots A_l$[14];

[14] Without further explanation, this notion is also used in our other statements presented in this section.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

(ii) If $\mathscr{D} \models_{\mathscr{AT}^0} \neg F$ **after** $\bar{A} \cdot A$ and $\mathscr{D} \not\models_{\mathscr{AT}^0} \neg F$ **after** $\bar{A}$, then
$\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Effect^-(F, Result(A,S))$ or $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Caused^-(F, Result(A,S))$.

While the intuition of Theorem 4 is quite clear, it does not hold for action domains of $\mathscr{AT}^1$ and $\mathscr{AT}^2$ since observation propositions of the form (20) is allowed in a domain description of $\mathscr{AT}^1$ or $\mathscr{AT}^2$ that may override an inertia rule in the corresponding PLP translation and present a change of a fluent's truth value even if there is no action or causal rule to cause such a change. In this case, we may think that either the fluent's truth value is changed by some external event that is not described in the domain description or the domain description is not properly specified. Weaker results may be obtained for domains of $\mathscr{AT}^1$ and $\mathscr{AT}^2$ under some restrictions.

Theorem 5
Let $\mathscr{D}$ be a consistent domain description of $\mathscr{AT}^1$ and $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ the corresponding PLP translation of $\mathscr{D}$. Suppose each observation proposition in $\mathscr{D}$ has the form

L **is observed if** $L_1, \ldots, L_m$ **with absence** $\bar{L}, L_{m+1}, \ldots, L_n$ **after** $\bar{A}$,

where $\bar{A}$ is not an empty string of actions. Then the following results hold:

(i) If $\mathscr{D} \models_{\mathscr{AT}^1} \neg F$ **after** $\bar{A}$, and $\mathscr{D} \models_{\mathscr{AT}^1} F$ **after** $\bar{A} \cdot A$, then
$\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Effect^+(F, Result(A,S))$ or $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Caused^+(F, Result(A,S))$;
(ii) If $\mathscr{D} \models_{\mathscr{AT}^1} F$ **after** $\bar{A}$ and $\mathscr{D} \models_{\mathscr{AT}^1} \neg F$ **after** $\bar{A} \cdot A$, then
$\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Effect^-(F, Result(A,S))$ or $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Caused^-(F, Result(A,S))$.

Theorem 6
Let $\mathscr{D}$ be a consistent domain description of $\mathscr{AT}^2$ and $\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D})$ the corresponding PLP translation of $\mathscr{D}$. Suppose each observation proposition in $\mathscr{D}$ has the form

L **is observed if** $L_1, \ldots, L_m$ **with absence** $\bar{L}, L_{m+1}, \ldots, L_n$ **after** $\bar{A}$,

where $\bar{A}$ is not an empty string of actions. Then the following results hold:

(i) If $\mathscr{D} \models_{\mathscr{AT}^2} \neg F$ **after** $\bar{A}$, and $\mathscr{D} \models_{\mathscr{AT}^2} F$ **after** $\bar{A} \cdot A$, then one of the following: results holds:
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Effect^+(F, Result(A,S))$;
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models AbEffect^+(F, Result(A,S))$; or
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Caused^+(F, Result(A,S))$;
(ii) If $\mathscr{D} \models_{\mathscr{AT}^2} F$ **after** $\bar{A}$, and $\mathscr{D} \models_{\mathscr{AT}^2} \neg F$ **after** $\bar{A} \cdot A$, then one of following Results holds:
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Effect^-(F, Result(A,S))$;
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models AbEffect^-(F, Result(A,S))$; or
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Caused^-(F, Result(A,S))$.

6.3 Restricted monotonicity

Monotonicity is a desirable property for reasoning about action in the sense that whenever new domain specific information is added to a domain description, no previous conclusion will be retracted. However, it is well known that most current

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

action formulations are nonmonotonic in general. In this section, we investigate some restricted monotonicity for action domains. Formally, let $\mathscr{D}$ be a domain description of $\mathscr{AT}^0$, $\mathscr{AT}^1$, or $\mathscr{AT}^2$. A domain description $\mathscr{D}'$ is called an *augment* of $\mathscr{D}$ if $\mathscr{D} \subseteq \mathscr{D}'$ and the only extra propositions in $\mathscr{D}'$ are observation propositions (or initial propositions in the case that $\mathscr{D}$ and $\mathscr{D}'$ are domain descriptions of $\mathscr{AT}^0$).

Definition 15
A domain description $\mathscr{D}$ of $\mathscr{AT}^i$ $(i = 0, 1, 2)$ is *monotonic with respect to observations* (we also simply call *O-monotonic*) if for each augment $\mathscr{D}'$ of $\mathscr{D}$, $\mathscr{D} \models_{\mathscr{AT}^i} L$ **after** $\bar{A}$ implies $\mathscr{D}' \models_{\mathscr{AT}^i} L$ **after** $\bar{A}$ $(i = 0, 1, 2)$.

It is clear that in general O-monotonicity does not hold for any domain description $\mathscr{D}$ due to a possibility that in the PLP translation of $\mathscr{D}'$, some new added observations may defeat previous conclusions derived through defeasible causal rules, inertial rules, action effect rules or action explanation rules. As an alternative, we can investigate proper restricted conditions under which O-monotonicity holds.

Theorem 7
Let $\mathscr{D}$ be a domain description $\mathscr{AT}^0$. $\mathscr{D}$ is O-monotonic if

(i) each causal proposition in $\mathscr{D}$ is of the form

L **is caused if** $L_1, \ldots, L_m$, and

(ii) $\mathscr{F}^+_{Initial} \cap (\mathscr{F}^-_{Effect} \cup \mathscr{F}^-_{Caused}) = \emptyset$,
$\mathscr{F}^-_{Initial} \cap (\mathscr{F}^+_{Effect} \cup \mathscr{F}^+_{Caused}) = \emptyset$, and
$(\mathscr{F}^+_{Effect} \cup \mathscr{F}^+_{Caused}) \cap (\mathscr{F}^-_{Effect} \cup \mathscr{F}^-_{Caused}) = \emptyset$.

Intuitively, Theorem 7 says that to guarantee a domain description $\mathscr{D}$ of $\mathscr{AT}^0$ to be O-monotonic, (i) all causal propositions in $\mathscr{D}$ should be non-defeasible, and (ii) all fluents involved in initial propositions, action effect propositions and causal propositions should be *irrelevant* in such a way: fluents involved in positive (or negative, resp.) initial propositions should be disjoint with fluents involved in negative (or positive, resp.) action effect and causal propositions, and fluents involved in positive action effect and causal propositions should be disjoint with fluents involved in negative action effect and causal propositions. Let $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$ be the PLP translation of $\mathscr{D}$. Condition (i) is necessary since this follows that adding any new initial fact rules in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$ will not defeat any fact $Holds(F, S)$ or $\neg Holds(F, S)$ that is derived through some causal rules at the initial situation S_0. Condition (ii), on the other hand, guarantees that initiating any action effect rules or causal rules by adding new initial fact rules into $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$ will not affect any previous facts derived through old initial fact rules, action effect rules, or causal rules.

An analogous result of Theorem 7, however, does not hold for domain descriptions of $\mathscr{AT}^1$ and $\mathscr{AT}^2$. In fact, since both $\mathscr{AT}^1$ and $\mathscr{AT}^2$ allow domain descriptions to have observations not only at the initial state but also at any other intermediate states, the property of O-monotonicity is hard to be achieved. For

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

instance, consider the PLP translation $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ of a domain description $\mathscr{D}$ of $\mathscr{AT}^1$, if $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Holds(F,S)$ and $Holds(F,S)$ is derived through instances of action explanation rules (23) and (25) in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$:

$$Effect^+(F', Result(A,S)) \leftarrow Holds(F', Result(A,S)), not\ \ Holds(F',S), \\ not\ \ Caused^+(F', Result(A,S)), \tag{49}$$

$$Holds(F,S) \leftarrow Effect^+(F', Result(A,S)), not\ \neg Holds(F,S), \tag{50}$$

Adding a new observation rule $Holds(F',S) \leftarrow$ into $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$, rule $Holds(F',S) \leftarrow$ will always override rule (49) and the fact $Holds(F,S)$ cannot be derived from the new PLP obtained by adding rule $Holds(F',S) \leftarrow$ into $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$. A similar example can be given for a domain description of $\mathscr{AT}^2$ as well. Therefore, in general, domain descriptions of $\mathscr{AT}^1$ and $\mathscr{AT}^2$ are not O-monotonic under the condition of Theorem 7.

6.4 Temporal definiteness

Besides O-monotonicity, there is also a class of action domains that satisfies a so-called temporal definiteness property in temporal reasoning. Consider a domain description $\mathscr{D}$. We say that $\mathscr{D}$ is *temporally definite* if for any value proposition of the form (4), $\mathscr{D} \models_{\mathscr{AT}^i} L$ **after** $\bar{A}$ implies $\mathscr{D} \models_{\mathscr{AT}^i} L$ **after** $\bar{A}'$ or $\mathscr{D} \models_{\mathscr{AT}^i} \bar{L}$ **after** $\bar{A}'$ $(i = 0, 1, 2)$, where $\bar{A}$ is a substring of $\bar{A}'$, i.e. $\bar{A}' = \bar{A} \cdot A_1 \ldots A_k$. Intuitively, temporal definiteness expresses a kind of definite information on fluents' truth values with respect to actions. For instance, if the switch is on initially, then we would expect that no matter what actions are executed afterward, the switch should be either on or off. It would be undesirable if after executing some actions, the status of switch becomes *unknown*.

As only deterministic actions are considered in our context, the temporal definiteness seems a reasonable requirement for our temporal reasoning. It is easy to verify that domain descriptions $\mathscr{D}(Switch\text{-}Power)$ and $\mathscr{D}(Switch\text{-}Power')$ described in section 3 are temporally definite. However, as defeasible information is allowed in domain descriptions, this property does not always hold.

Example 10

Consider a scenario where there are constraints: (1) birds normally can fly; (2) a wounded bird normally cannot fly. Suppose we initially know that a specific bird Tweety is not wounded. Then after being shot, Tweety is wounded. What we are interested in is whether Tweety can fly after she is shot. We name this scenario *Shooting*-3 which can be described by our action language $\mathscr{AT}^0$. Let $\mathscr{D}(Shooting\text{-}3)$ be a domain description of $\mathscr{AT}^0$ including the following propositions:

initially $\neg$ *Wounded*,
Fly **is caused if with absence** $\neg$*Fly*,
$\neg$ *Fly* **is caused if** *Wounded* **with absence** *Fly*,
Shoot **causes** *Wounded*.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Now we translate $\mathscr{D}(Shooting\text{-}3)$ into the corresponding PLP $\mathscr{P}^{\mathscr{AT}^0}(Shooting\text{-}3) = (\Pi^0_{spec} \cup \Pi^0_{ind}, \mathscr{N}, <)$, where Π^0_{spec} consists of the following rules[15]:

Initial fact rule:
$N_1 : \neg Holds(Wounded, S_0) \leftarrow.$
Causal rules:
$N_2 : Caused^+(Fly, s) \leftarrow not\ \neg Holds(Fly, s).$
$N_3 : Caused^-(Fly, s) \leftarrow Holds(Wounded, s), not\ Holds(Fly, s).$
Action effect rule:
$N_4 : Effect^+(Wounded, Result(Shoot, s)) \leftarrow.$

Then it is easy to see $\mathscr{P}^{\mathscr{AT}^0}(Shoot\text{-}3) \models Holds(Fly, S_0)$ (e.g. Tweety can fly initially). Furthermore, it is also not difficult to conclude that there exist two answer sets for $\mathscr{P}^{\mathscr{AT}^0}(Shoot\text{-}3)$ such that $Holds(Fly, Result(Shoot, S_0))$ is in one answer set and $\neg Holds(Fly, Result(Shoot, S_0))$ is in another. So we have

$\mathscr{D}(Shooting\text{-}3) \models_{\mathscr{AT}^0}$ **initially** *Fly*,
$\mathscr{D}(Shooting\text{-}3) \not\models_{\mathscr{AT}^0}$ *Fly* **after** *Shoot*,
$\mathscr{D}(Shooting\text{-}3) \not\models_{\mathscr{AT}^0} \neg Fly$ **after** *Shoot*.

So $\mathscr{D}(Shooting\text{-}3)$ is not temporally definite. But intuitively, we would prefer that Tweety cannot fly after being shot because causal rule N_3 seems to be more specific than N_2. Solving this problem involves the issue of representing domain-dependent preference which will be discussed in section 6.5.

Lemma 1
A domain description $\mathscr{D}$ of $\mathscr{AT}^i$ $(i = 0, 1, 2)$ is temporally definite if its PLP translation $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ has a unique answer set.

The converse of Lemma 1, however, does not hold. That is, for a temporally definite domain description, its PLP translation may have more than one answer set. For instance, in domain description $\mathscr{D}(Shooting\text{-}3)$ described above, if we initially know that Tweety is already wounded, then the modified domain description becomes temporally definite but its PLP translation will still have more than one answer sets, i.e. one answer set includes $Holds(Fly, S_0)$ while the other includes $\neg Holds(Fly, S_0)$.

Lemma 1 actually presents a sufficient condition to ensure a domain description to be temporally definite. Observing Example 10, we can see that $\mathscr{P}^{\mathscr{AT}^0}(Shooting\text{-}3)$ has more than one answer set because two causal rules N_2 and N_3 conflict with each other on fluent *Fly*'s truth value in situation $Result(Shoot, S_0)$, while *Fly*'s truth value is initially true, i.e. $\mathscr{P}^{\mathscr{AT}^0}(Shooting\text{-}3) \models Holds(Fly, S_0)$. This observation motivates our examination on the structure of an action domain.

Consider an extended logic program Π. Using a procedure proposed by Gelfond and Lifschitz (see Appendix A), we can actually transform Π into a general logic program[16], denoted by $Trans(\Pi)$. It has been showed that a sufficient condition to

[15] For simplicity, here we omit the explicit description of naming function $\mathscr{N}$ and $<$-relations.
[16] A *general logic program* is a set of rules of the form $A \leftarrow B_1, \ldots, B_m, not\ B_{m+1}, \ldots, not\ B_n$, where $A, B_1, \ldots, B_n$ are atoms. Also see Appendix A.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

ensure that $Trans(\Pi)$ has a unique stable model (or answer set under the context of extended logic program) is that $Trans(\Pi)$ is *locally stratified*. That means, there does not exist any potential conflict among any rules in $Trans(\Pi)$ (see Appendix A for a technical description on local stratification). Therefore, we have the following result.

Theorem 8
A domain description $\mathscr{D}$ of $\mathscr{AT}^i$ $(i=0,1,2)$ is temporally definite if its PLP translation $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ has a unique reduct Δ^i and $Trans(\Delta^i)$ is locally stratified.

Theorem 8 implies that to guarantee a domain description $\mathscr{D}$ to be temporally definite, no conflict should occur among the same type of defeasible rules after reducing $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ to its reduct Δ^i. In Example 10, since two causal rules N_2 and N_3 contain a potential conflict with each other, it causes $\mathscr{D}(Shooting\text{-}3)$ to be not temporally definite. However, conflicts between different types of defeasible rules will not affect the temporal definiteness for a domain description because such a conflict can be resolved during the process of generating a reduct of the PLP translation of the domain description.

6.5 Indefiniteness and domain-dependent preferences

As we mentioned before, temporal definiteness is a desirable property in temporal reasoning. However, it is also the fact that sometimes a domain description which is not temporally definite may still present right results from our intuition. For instance, in the domain of Switch-Power presented in section 3, if we add one more causal proposition into $\mathscr{D}(Switch\text{-}Power)$:

$\neg On$ **is caused if with absence** $Power$.

which says that if there is no explicit information stating that there is power, then it is assumed that the light is not on, the circumstance will then change. Suppose that initially we know that the light is not on, the switch is off, and there is no any information about if there is power. Then after turning on the switch, we would like to know whether the light is on. It is not difficult to show that the modified domain description, say $\mathscr{D}(Switch\text{-}Power'')$, is not temporally definite. Specifically, we have

$\mathscr{D}(Switch\text{-}Power'') \models_{\mathscr{AT}^0}$ **initially** $\neg On$,
$\mathscr{D}(Switch\text{-}Power'') \not\models_{\mathscr{AT}^0} On$ **after** $Turn\text{-}On$,
$\mathscr{D}(Switch\text{-}Power'') \not\models_{\mathscr{AT}^0} \neg On$ **after** $Turn\text{-}On$.

Although action $Turn\text{-}On$ is deterministic (see its effect proposition in Example 4 in section 3.2.2), the above indefinite result seems reasonable from our intuition because without having definite information about power, it is impossible to decide whether the light is on after performing action $Turn\text{-}On$ due to a conflict between two causal propositions in $\mathscr{D}(Switch\text{-}Power'')$.

This example reveals that although temporal definiteness sometimes indeed describes a desired property, it should not become a particular restriction on action domains. So far, in our domain descriptions, preferences are used as built-in

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

mechanisms of their PLP translations to handle conflicts among different types of propositions. It is observed that domain-dependent preferences also play important roles in temporal reasoning. For instance, in some domains, it is the case that within the same type of defeasible propositions, one proposition is more preferred than the other. Consider Example 10 presented in section 6.4 once again. We have mentioned that two causal propositions

Fly **is caused if with absence** $\neg Fly$,
$\neg Fly$ **is caused if** $Wounded$ **with absence** Fly,

contain a conflict under the circumstance by knowing that Tweety is wounded. This conflict leads $\mathscr{D}(Shooting\text{-}3)$ to be temporally indefinite. But from our intuition, the second causal proposition seems to represent more specific information than the first causal proposition. Therefore, during the temporal reasoning, once conflict occurs between these two causal rules, we would prefer the second causal proposition to defeat the first one (e.g. the wounded bird Tweety normally cannot fly if we do not know she can fly).

This problem may be handled by including domain-dependent preferences on causal and observation propositions into the corresponding PLP translations of domain descriptions. For instance, in Example 10, we may add preference $N_3 < N_2$ into $\mathscr{P}^{\mathscr{AT}^0}(Shooting\text{-}3)$, and then $\mathscr{P}^{\mathscr{AT}^0}(Shooting\text{-}3)$ becomes temporally definite and the fact $\neg Fly$ **after** $Shoot$ is entailed from the modified domain description.

In general, to represent domain-dependent preferences in a domain description, we need to extend the language so that preference between two propositions can be explicitly expressed. One way of doing this is to introduce *labels* in the language and each proposition in the domain description is assigned a unique label. A *preference proposition* can be proposed as follows:

$$l_1 \textbf{ is more preferred than } l_2, \tag{51}$$

where l_1 and l_2 are labels for causal or observation propositions in the domain description. Then we define the PLP translation of the extended domain description as $(\Pi, \mathscr{N}, < \cup <_C \cup <_O)$ $(i = 0, 1, 2)$, where Π, $\mathscr{N}$ and $<$ are the same as before, and $<_C$ and $<_O$ are the preference orderings on causal and observation rules respectively that correspond to the specified preference propositions of the form (51) in the domain description.

7 Related work

In this section, we discuss some related work. In the research of reasoning about action, it is difficult to evaluate various action theories from a systematic standard though some studies on this topic have been developed (Sandewall, 1994). To compare with competing approaches, people usually have to demonstrate their methods with a small number of typical examples. It is still not clear yet what should be the unified standard for an action theory to satisfy. We feel that it would be rather weak to compare our approach with other action theories just through a small number of examples. As defeasibility handling is the central issue in our action

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

formulation proposed in this paper, we focus on this point as a major criterion to compare our approach with other methods.

An early effort on handling defeasible causal rules in reasoning about action was due to the author's previous work (Zhang, 1999), in which the author identified the restriction of McCain and Turner's causal theory of actions (McCain and Turner, 1995) and claimed that, in general, a causal rule should be treated as a defeasible rule in order to solve the ramification problem properly. In Zhang (1999), constraints (1) and (2) simply correspond to defaults $Switch : On/On$ and $\neg Power : /\neg On$, respectively. By combining Reiter's default theory (Reiter, 1980) and Winslett's PMA (Winslett, 1988), the author developed a causality-based minimal change principle for reasoning about action and change which subsumes McCain and Turner's causal theory.

Although the work presented in Zhang (1999) provided a natural way to represent causality in reasoning about action, there were several restrictions in this action theory. First, due to technical restrictions, only normal defaults or defaults without justifications are the suitable forms to represent causal rules in problem domains. Secondly, this action theory did not handle the other two major defeasibilities – defeasible observations and actions with defeasible and abnormal effects.

Probably Jabłonowski, Łukaszewicz and Madalińska-Bugaj's work (Jabłonowski *et al*., 1996) was one of the early efforts on handling the problem of defeasible observations and actions with abnormal effects. Following Dijkstra's semantics on programming languages (W. Łukaszewcz and Madalińsks-Bugaj, 1995), they proposed an action theory in which both defeasible observations and actions with abnormal effects were expressible. Their work actually presented a few new features. For instance, by employing Dijkstra's semantics in action theory, their method reduced the computational cost in action reasoning; it also dealt with both temporal prediction and postdiction reasoning while incomplete information is allowable in problem domains.

However, the major limitation of this approach is that it did not solve the ramification problem properly. To deal with domain constraints in action scenarios, the action theory has to be extended by adding statements like

$$A; release(F_1); \ldots; release(F_n),$$

which means that fluents $F_1, \ldots, F_n$ involved in domain constraints may not obey the inertia rule with respect to the performance of action A (W. Łukaszewcz and Madalińsks-Bugaj, 1995). For example, if we combine a constraint like "*the fact that the turkey is not alive implies that the turkey is not walking*" into the previous shooting scenario, in order to derive an indirect effect $\neg Walk$ of action *Shoot*, a statement like $Shoot; release(Walk)$ has to be added into the action theory. But to specify such statements, we have to know how each action exactly affects fluents involved in the domain constraint. Obviously for a complex problem domain this usually is not practicable without taking causality into account. Not surprisingly, due to such restriction, this approach is also hard to be extended to handle defeasible constraints in reasoning about action.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Baral and Lobo recently also proposed an action formulation to address the issue of defeasible constraints and actions with defeasible effects (Baral and Lobo, 1997). Following a similar spirit of Gelfond and Lifschitz's action language $\mathcal{A}$ (Gelfond and Lifschitz, 1993), Baral and Lobo proposed an action language named $\mathcal{ADC}$ to describe action domains in which both defeasible constraints and actions with defeasible effects are admitted. In their language $\mathcal{ADC}$ a defeasible constraint like (1) is represented as

Switch **normally suffices for** *On*

and the defeasible *Shoot* action illustrated in Example 5 is represented as

Shoot **normally causes** $\neg$*Alive* **if** *Loaded*.

As shown in Baral and Lobo (1997), $\mathcal{ADC}$ has a simple syntax. Based on an extended logic program translation, a transition system is defined to provide a formal semantics of $\mathcal{ADC}$.

It is worth mentioning that our idea of defining semantics for $\mathcal{AT}^0$, $\mathcal{AT}^1$ and $\mathcal{AT}^2$ is similar to Baral and Lobo's proposal for $\mathcal{ADC}$. Both of these two approaches directly use logic programs to define a transition system for the action language, instead of developing a separate semantics like $\mathcal{A}$ language. Also, both approaches define states in a different way from the standard $\mathcal{A}$ language, that is, instead of defining a state to be a truth value assignment on fluents, these two approaches define a state to be a collection of fluent expressions so that incomplete information about fluents becomes allowable.

Nevertheless, some restrictions exist in action language $\mathcal{ADC}$: it can only reason about forward, i.e. temporal prediction, and observations on intermediate situations and final situation are not expressible. Therefore, their approach cannot deal with temporal postdiction. On the other hand, although actions with defeasible effects are allowed in the domain description, it seems that the issue of solving conflicts between defeasible action effect propositions and defeasible constraints was not addressed in detail.

Finally, we briefly mention Geffner's recent work on causal theory of action (Geffner, 1997) which is closely related to models of causal reasoning based on Bayesian networks and structural equation models (Goldszmidt and Pearl, 1992). To provide a well-founded solution to the ramification problem, Geffner claimed that causal rules of the domain should be defeasible in general. Although with a very different language and methodology, Geffner's system actually addressed the same problem discussed in Zhang (1999) and Baral and Lobo (1997). However, from the viewpoint of defeasibility handling, this system is restricted because defeasible observations and actions with defeasible and abnormal effects were not considered.

8 Conclusions

We have developed a unified action formulation to handle three types of defeasibilities in reasoning about action. Our formulation consists of three action languages named $\mathcal{AT}^0$, $\mathcal{AT}^1$ and $\mathcal{AT}^2$ respectively. We have showed that our action formulation is applicable to both temporal prediction and postdiction with

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

incomplete information while defeasible constraints, defeasible observations and actions with defeasible and abnormal effects are admitted. As discussed in the previous section, although the issue of defeasibility in reasoning about action has been addressed by some researchers recently, our work presented here is the first effort to handle various defeasible information in temporal reasoning by using a prioritized logic programming approach. It enhances the viewpoint that the logic programming languages can be employed as efficient low level formal languages for reasoning about action.

Besides the author's work (Zhang and Foo, 1997a), different prioritized logic programming formalisms have been proposed recently (Brewka, 1996; Brewka and Eiter, 1999; Grosof, 1997). The reason why we choose our PLPs to develop our action formulation is as follows. First, we think that the answer set semantics for PLPs provides an intuitive and natural interpretation for conflict resolution in logic programs, and hence it is easy to use not only in reasoning about action, but also in other aspects of modeling system dynamics (Zhang and Foo, 1997b; Zhang and Foo, 1998). Secondly, a propositional prioritized logic programming system (PLPS) has been implemented recently by the author and his students (Zhang *et al.*, 2001). We believe that our PLPS can finally provide a practical programming language prototype for representing actions with the capability of the defeasibility handling within the framework we proposed in this paper.

The computational issue of prioritized logic programs has been addressed in other work (Zhang, 2001). Briefly, the author has proved that for a propositional prioritized logic program, deciding whether it has an answer set is NP-complete, and deciding whether a given ground literal is entailed from this prioritized logic program is Π_2^P-complete.

It is also easy to observe that, since a rule containing variables in a PLP is viewed as a set of ground instances of this rule by replacing variables with all possible constants occurring in the PLP, under the case that a PLP does not have function symbols, the number of defeated rules eliminated from this PLP as described in Definition 2 is always finite. Hence, we can always compute a finite reduct of such PLP[17].

In the case that there are function symbols occurring in a PLP, the situation is different. Basically, the set of ground instances of a rule, that includes variables and function symbols, may be infinite and therefore it might be possible that there are infinite number of defeated rules which should be eliminated from the original PLP. Under this situation, a reduct containing infinite rules may be produced according to Definition 2. From a practical viewpoint, we are only able to deal with finite reducts. To overcome this problem, we can set a proper restriction on the variable substitution. For instance, in the modified Switch-Power domain discussed in Example 4 (see section 3.2.2), if all we are interested is to know what are the effect after actions *Cut-Power* and *Turn-On* are executed, then in the computation of the answer set of $\mathscr{P}^{\mathscr{AT}^0}(Switch\text{-}Power')$, we only need to consider situations S_0,

[17] Note that Theorem 1 shows that every PLP has a reduct, but such a reduct may contain infinite rules.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Result(*Turn-On*, S_0), *Result*(*Cut-Power*, S_0), and *Result*(*Turn-On*, *Result*(*Cut-Power*, S_0)). This implies that the ground form of PLP $\mathscr{PAT}^0$(*Switch-Power'*) only has finite rules, and hence it always has a finite reduct.

Finally, we should mention that currently our action formulation cannot represent nondeterministic actions and disjunctive domain information. That is, we only consider deterministic problem domains in this paper. This is due to the limit of prioritized logic programs inherited from extended logic programs. But we would argue that our prioritized logic programs are extendedable to represent disjunctive information by using a similar method described in (Gelfond and Lifschitz, 1991) for extended logic programs, and our action formulation can then be extended to represent nondeterministic actions.

Acknowledgements

The author thanks anonymous referees for many valuable comments on the early version of this paper.

Appendix A: General logic programs and stratification

A *general logic program* is a finite set of rules of the form

$$A \leftarrow B_1, \ldots, B_m, not\ B_{m+1}, \ldots, not\ B_n, \tag{52}$$

where $A, B_1, \ldots, B_m, \ldots, B_n$ are atoms.

Gelfond and Lifschitz developed a transformation to reduce an extended logic program to a general logic program (Gelfond and Lifschitz, 1991). Consider an extended logic program Π. For any predicate P occurring in Π, let P' be a new predicate of the same arity. The atom $P'(x)$ is called the *positive form* of the negative literal $\neg P(x)$. Every positive literal is, by definition, its own positive form. The positive form of a literal L will be denoted by L^+. Π^+ stands for the general program obtained from Π by replacing each rule $L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$ in Π by rule

$$L_0^+ \leftarrow L_1^+, \ldots, L_m^+, not\ L_{m+1}^+, \ldots, not\ L_n^+.$$

Proposition 2
(Gelfond and Lifschitz, 1991). A consistent set $S \subset Lit$ is an answer set of Π iff S^+ is an answer set of Π^+.

Definition 16
(Local stratification (Apt and Bol, 1994)).
Let Π be a general logic program.

- A *local stratification* for Π is a function ψ from the Herbrand base of Π, B_Π, to the countable ordinals.
- Given a local stratification ψ, we extend it to ground negative literals[18] by setting $\psi(not\ A) = \psi(A) + 1$.

[18] Note that here we mean weak negation *not*.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

- A rule with form (52) of Π is called *locally stratified with respect to a local stratification* ψ if for every ground instance of (52),

$$A' \leftarrow B'_1, \ldots, B'_m, not\ B'_{m+1}, \ldots, not\ B'_n,$$

$\psi(A') \geqslant \psi(B'_i)$, *where* $1 \leqslant i \leqslant m$, and
$\psi(A') \geqslant \psi(not\ B'_j)$, *where* $m+1 \leqslant i \leqslant n$.

- Π is called *locally stratified with respect to a local stratification* ψ if all its rules are. Π is called *locally stratified* if it is locally stratified with respect to some local stratification.

Proposition 3
(Gelfond and Lifschitz, 1988). If a general logic program Π is locally stratified, then by treating Π as an extended logic program where each rule does not contain classical negation, it has a unique answer set.

Appendix B: Proofs

Theorem 1
Every PLP has a reduct.

To prove Theorem 1, we need to introduce the concept of $<$-partition for a PLP.

Definition 17
Let $\mathscr{P} = (\Pi, \mathscr{N}, <)$ be an arbitrary PLP. A $<$-*partition* of Π in $\mathscr{P}$ is a finite collection $\{\Pi_1, \ldots, \Pi_k\}$, where $\Pi = \Pi_1 \cup \ldots \cup \Pi_k$ and Π_i and Π_j are disjoint for any $i \neq j$, such that

1. $\mathscr{N}(r) < \mathscr{N}(r') \in \mathscr{P}(<)$ implies that there exist some i and j $(1 \leqslant i < j)$ such that $r' \in \Pi_j$ and $r \in \Pi_i$;
2. for each rule $r' \in \Pi_j$ $(j > 1)$, there exists some rule $r \in \Pi_i$ $(1 \leqslant i < j)$ such that $\mathscr{N}(r) < \mathscr{N}(r') \in \mathscr{P}(<)$.

Example 11
Consider a PLP $\mathscr{P}_3 = (\Pi, \mathscr{N} <)$:

$\mathscr{P}_3 : N_1 : A \leftarrow not\ B, not\ C,$
$N_2 : B \leftarrow not\ \neg C,$
$N_3 : C \leftarrow not\ A, not\ \neg C,$
$N_4 : \neg C \leftarrow not\ C,$
$N_1 < N_2, N_2 < N_4, N_3 < N_4.$

It is easy to verify that a $<$-partition of Π in $\mathscr{P}_3$ is $\{\Pi_1, \Pi_2, \Pi_3\}$, where

Π_1 : $N_1 : A \leftarrow not\ B, not\ C,$
$N_3 : C \leftarrow not\ A, not\ \neg C,$
Π_2 : $N_2 : B \leftarrow not\ \neg C,$
Π_3 : $N_4 : \neg C \leftarrow not\ C.$

In fact, this program has a unique answer set $\{B, C\}$.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Lemma 2
Every prioritized logic program has a $<$-partition.

Proof
For a given PLP $\mathscr{P} = (\Pi, \mathscr{N}, <)$, we construct a series of subsets of Π as follows:

$\Pi_1 = \{r \mid$ there does not exist a rule $r' \in \Pi$ such that $\mathscr{N}(r') < \mathscr{N}(r)\}$;
$\Pi_i = \{r \mid$ for all rules such that $\mathscr{N}(r') < \mathscr{N}(r)$, $r' \in \bigcup_{j=1}^{i-1} \Pi_j\}$.

We prove that $\{\Pi_1, \Pi_2, \ldots\}$ is a $<$-partition of $\mathscr{P}$. First, it is easy to see that Π_i and Π_j are disjoint. Now we show that this partition satisfies Conditions 1 and 2 described in Definition 17. Let $\mathscr{N}(r) < \mathscr{N}(r') \in \mathscr{P}(<)$. If there does not exist any rule $r'' \in \Pi$ such that $\mathscr{N}(r'') < \mathscr{N}(r)$, then $r \in \Pi_1$. Otherwise, there exists some i $(1 < i)$ such that $r \in \Pi_i$ and for all rules satisfying $\mathscr{N}(r'') < \mathscr{N}(r)$ $r'' \in \Pi_1 \cup \cdots \cup \Pi_{i-1}$. Let $r' \in \Pi_j$. Since $\mathscr{N}(r) < \mathscr{N}(r')$, it follows that $1 < j$. From the construction of Π_j, we also conclude $r \in \Pi_1 \cup \cdots \cup \Pi_{j-1}$. Since $r' \in \Pi_i$, it follows $i \leqslant j - 1$. That is, $i < j$. Condition 2 directly follows from the construction of the partition described above.

Now we show that $\{\Pi_1, \Pi_2, \ldots\}$ must be a finite set. First, if Π is finite, it is clear $\{\Pi_1, \Pi_2, \ldots\}$ must be a finite set. If Π contains infinite rules, then according to our assumption presented in section 2, $\mathscr{P}$ must be the ground instantiation of some program, say $\mathscr{P}^* = (\Pi^*, \mathscr{N}^*, <^*)$ where Π^* is finite. Then we can use the same way to define a $<$-partition for $\mathscr{P}^*$. Since Π^* is finite, the partition of $\mathscr{P}^*$ must be also finite: $\{\Pi_1^*, \Pi_2^*, \ldots, \Pi_k^*\}$. As $\mathscr{P}^*$ is well formed, it implies that for each i $(i = 1, 2, \ldots)$, Π_i is the ground instantiation of Π_i^*. So $\{\Pi_1, \Pi_2, \ldots\} = \{\Pi_1, \Pi_2, \ldots, \Pi_k\}$ which is finite. □

Proof of Theorem 1

Let $\mathscr{P} = (\Pi, \mathscr{N}, <)$. From Lemma 2, we can assume Π has a partition $\Pi = \Pi_1 \cup \cdots \cup \Pi_k$. We will show that $\mathscr{P}$ has a fixpoint in the process of reduction according to Definition 2. As $\Pi_1, \ldots, \Pi_k$ are disjoint and for any $\mathscr{N}(r) < \mathscr{N}(r')$, it implies $r \in \Pi_i$ and $r' \in \Pi_j$ where $i < j$, we can use notation

$$\Pi_1 < \Pi_2 < \ldots < \Pi_k$$

to illustrate this property. It is easy to see that for each rule in Π_i $(1 < i < k)$, there must exist some j and h that $j < i < h$ such that $\mathscr{N}(r') < \mathscr{N}(r) < \mathscr{N}(r'')$ and $r' \in \Pi_j$, $r'' \in \Pi_h$. Now we construct a sequence of reductions that starts from those least preferred rules in Π_k, then from rules in $\Pi_{k-1} \cup \Pi_k$, and so on as illustrated below:

$\Pi^{(0)} = \Pi = \Pi_1 \cup \cdots \cup \Pi_k$;
$\Pi^{(1)} = \Pi^{(0)} - \{r_1, r_2, \ldots \mid r_1, r_2, \ldots \in \Pi_k$ and $r_1, r_2, \ldots$ satisfy the conditions as stated in Definition 2$\}$;
$\Pi^{(2)} = \Pi^{(1)} - \{r_1, r_2, \ldots \mid r_1, r_2, \ldots \in \Pi_{k-1} \cup \Pi_k$ and $r_1, r_2, \ldots$ satisfy the conditions as stated in Definition 2$\}$;
$\Pi^{(3)} = \Pi^{(2)} - \{r_1, r_2, \ldots \mid r_1, r_2, \ldots \in \Pi_{k-2} \cup \Pi_{k-1} \cup \Pi_k$ and $r_1, r_2, \ldots$ satisfy the conditions as stated in Definition 2$\}$;
...
$\Pi^{(k-1)} = \Pi^{(k-2)} - \{r_1, r_2, \ldots \mid r_1, r_2, \ldots \in \Pi_2 \cup \ldots \cup \Pi_k$ and $r_1, r_2, \ldots$ satisfy the conditions as stated in Definition 2$\}$.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

It is observed that in the above reduction process, after obtaining $\Pi^{(k-1)}$, no more rules can be eliminated from $\Pi^{(k-1)}$ by applying the conditions of Definition 2 because after the ith reduction, all orderings inherited from $\Pi_{k-i+1} < \cdots < \Pi_k$ will no longer play any roles in the further $(i+1)$th,..., and $(k-1)$th reductions. In particular, in the ith reduction of obtaining $\Pi^{(i)}$, all rules eliminated from $\Pi^{(i-1)}$ (note that there may be infinite number of rules to be eliminated in the ith reduction) are due to some rules in $\Pi_1 \cup \cdots \cup \Pi_{k-i}$ which are more preferred than those eliminated rules in $\Pi_{k-i+1} \cup \cdots \cup \Pi_k$. As k is a finite number, from Definition 2 $\Pi^{(k-1)}$ is also a reduct of $\mathscr{P}$. □

Proposition 1

Let $\mathscr{D}$ be a domain description of $\mathscr{AT}^i$ and $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ $(i = 0, 1, 2)$ the corresponding PLP translation of $\mathscr{D}$ specified previously. $\mathscr{D}$ is consistent if and only if $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ has a consistent answer set.

Proof

Here we only prove the result for $\mathscr{AT}^0$, proofs for other cases are similar.

Suppose $\mathscr{D}$ has a model Ψ. Then according to Definition 7, for any action string $\bar{A}$ such that $\Psi(\bar{A})$ is defined and any fluent F, F and $\neg F$ cannot be both true in $\Psi(\bar{A})$. From the definition of $\Psi(\bar{A})$, i.e. Definition 6, it follows that $\Psi(\bar{A}' \cdot A) \in \mathscr{R}(A, \Psi(\bar{A}'))$, where $\bar{A} = \bar{A}' \cdot A$. Here we assume that $\bar{A}$ is not empty (otherwise, $\Psi(\epsilon) = \hat{S}_0$ that we will consider next). Also since Ψ's domain is prefix closed, $\Psi(\bar{A}')$ is also defined. Then from Definition 5 of transition function $\mathscr{R}$, it follows that $\mathscr{R}(A, \Psi(\bar{A}')$ contains a consistent set of fluent expressions. As this set is directly deduced from some answer set *Ans* of $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, it concludes that the subset of *Ans* consisting of all literals of the form $Holds(F, S)$ or $\neg Holds(F, S)$ is consistent (note $S \neq S_0$). Now we consider the case of empty action string. In this case $\Psi(\epsilon) = \hat{S}_0$. As Ψ is a model, $\hat{S}_0$ must be a consistent set. Again, as $\hat{S}_0$ is deduced from some answer set *Ans* of $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, it concludes that the subset of *Ans* consisting all literals of the form $Holds(F, S_0)$ or $\neg Holds(F, S_0)$ is consistent. Therefore, the subset of *Ans* of the following form is consistent:

$$\{[\neg]Holds(F, S_0), \ldots\} \cup \cdots \{[\neg]Holds(F, S), \ldots\}.$$

Recall that *Ans* also contains a subset that consists of atoms of the forms $Effect^+(F, S)$, $Effect^-(F, S)$, $Caused^+(F, S)$ and $Caused^-(F, S)$. Clearly, this subset of *Ans* is also consistent. So *Ans* is consistent.

Now suppose *Ans* is a consistent answer set of $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$. Then from Definitions 5, 6, and 7, we can construct a model Ψ for $\mathscr{D}$ in an obvious way. □

Theorem 2

Every normal domain description of $\mathscr{AT}^0$ is consistent.

Proof

Let $\mathscr{D}$ be a normal domain description of $\mathscr{AT}^0$. That is, $\mathscr{D}$ satisfies Conditions (i), (ii) and (iii) in Definition 12. According to Proposition 1, we only need to show that the PLP translation $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$ of $\mathscr{D}$ has a consistent answer set. Let $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) = (\Pi, \mathscr{N}, <)$. First, from Condition (ii) and the construction of $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, it is observed that Π does not contain rules of the following forms:

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

r_1: $L_1 \leftarrow \cdots, not\ L^*, \ldots,$
r_2: $L_2 \leftarrow \cdots, L_1, \ldots,$
$\ldots,$
r_k: $L_k \leftarrow \cdots, L_{k-1}, \ldots,$
r_{k+1}: $L^* \leftarrow \cdots, L_k, \ldots.$

This actually ensures that Π has an answer set *Ans*. To show this, we assume that Π does not have an answer set. Then there must exist some literal L^* satisfying the condition: for any set S of ground literals (S can be empty) (a) if $L^* \notin S$, then L^* is in the answer set of program Π^S (Π^S is obtained from Π by doing Gelfond–Lifschitz transformation on Π in terms of S); and (b) if $L^* \in S$, then L^* is not in the answer set of program Π^S. It is worth to mention that since Π^S does not contain rules including negation as failure sign, Π^S always has an answer set. From case (a), it is implied that Π must contain a rule of the form:

r'_{k+1}: $L^* \leftarrow \cdots.$

On the other hand, from case (b), it is easy to observe that all rules of the form r'_{k+1} cannot be triggered in Π^S due to $L^* \in S$. That is, some rule of the form

r_1: $L_1 \leftarrow \cdots, not\ L^*, \ldots$

must be contained in Π (we do not exclude the case that $L_1 = L^*$). This follows that rule r'_{k+1} actually has a form:

r'_{k+1} : $L^* \leftarrow \cdots, L', \ldots$

such that the deletion of r_1 from Π will cause literal L' not to be triggered and hence L^* can not be derived from Π^S. Without loss of generality, we can assume that Π contains a sequence of rules $r_1, \ldots, r_{k+1}$ as described above.

Now we consider Condition (i). From Condition (i), we know that Π does not contain a pair of rules of the forms:

$Holds(F, S_0) \leftarrow,$
$\neg Holds(F, S_0) \leftarrow.$

This follows that a subset of *Ans* in which each liteal is associated with initial situation S_0:

$\{[\neg]Holds(F_1, S_0), \ldots, [\neg]Holds(F_k, S_0)\}$

is consistent. Now we consider a pair of complementary propositions (l, l') in $\mathscr{D}$. To simplify our presentation, for a rule of the form:

r: $L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$

we denote $pos(r) = \{L_1, \ldots, L_m\}$ and $neg(r) = \{L_{m+1}, \ldots, L_n\}$. Then r can be simply represented as $L_0 \leftarrow pos(r), neg(r)$. Under this notation, a pair of complementary propositions l and l' in $\mathscr{D}$ may have one of the following possible translations in Π:

(a) r : $Caused^+(F, s) \leftarrow pos(r), neg(r),$
r' : $Caused^-(F, s) \leftarrow pos(r'), neg(r'),$

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

(b) $r : Caused^{+}(F, s) \leftarrow pos(r), neg(r)$,
$r' : Effect^{-}(F, Result(A, s)) \leftarrow pos(r'), neg(r')$,
(c) $r : Effect^{+}(F, Result(A, s)) \leftarrow pos(r), neg(r)$,
$r' : Effect^{-}(F, Result(A, s)) \leftarrow pos(r'), neg(r')$.

From Condition (iii), we know that in each case of (a), (b) and (c), $pos(r)$ and $pos(r')$ cannot be both true in answer set *Ans*. Hence, for any situation term S, none of these three pairs of atoms $Caused^{+}(F, S)$ and $Caused^{-}(F, S)$, $Caused^{+}(F, S)$ and $Effect^{-}(F, S)$, or $Effect^{+}(F, S)$ and $Effect^{-}(F, S)$ cannot both true in *Ans*. This concludes that *Ans* does not contain any complementary literals $Holds(F, S)$ and $\neg Holds(F, S)$ for any F and S. So *Ans* is a consistent answer set of Π. Furthermore, every answer set of Π is also consistent (Lifschitz and Turner, 1994). Finally, from the property that a PLP $(\Pi, \mathscr{N}, <)$ has an answer set iff Π has an answer set and every answer set of $(\Pi, \mathscr{N}, <)$ is also an answer set of Π (Zhang, 2001), it concludes that $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$ has a consistent answer set (and its every answer set is also consistent). □

Theorem 3
Every normal domain description of $\mathscr{AT}^1$ or $\mathscr{AT}^2$ is consistent.

Proof
The proof is similar to the proof of Theorem 2 but with additional considerations on action explanation rules and action abnormal effect rules in $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$. We omit it here. □

Theorem 4
Given a domain description $\mathscr{D}$ of $\mathscr{AT}^0$ and its PLP translation $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, the following results hold:

(i) If $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F, Result(A, S))$ and $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \not\models Holds(F, S)$, then $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Effect^{+}(F, Result(A, S))$ or $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Caused^{+}(F, Result(A, S))$;
(ii) If $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models \neg Holds(F, Result(A, S))$ and $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \not\models \neg Holds(F, S)$, then $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Effect^{-}(F, Result(A, S))$ or $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Caused^{-}(F, Result(A, S))$.

Proof
It is sufficient to only prove (i). Since $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F, Result(A, S))$ and $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \not\models Holds(F, S)$, it follows that for each answer set *Ans* of $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, $Holds(F, Result(A, S)) \in Ans$, and there is some answer set *Ans*′ such that $Holds(F, S) \notin Ans'$. Therefore, the fact that $Holds(F, Result(S, A))$ is true is *not* due to inertia rules (17) and (18) in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, but due to action effect rules (13) and (14), or causal rules (9) and (10). That is, $\mathscr{P}^{\mathscr{AT}^0} \models Effect^{+}(F, Result(A, s))$ or $\mathscr{P}^{\mathscr{AT}^0} \models Caused^{+}(F, Result(A, S))$. □

Theorem 5
Let $\mathscr{D}$ be a domain description of $\mathscr{AT}^1$ and $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ be its PLP translation. Suppose each observation proposition in $\mathscr{D}$ has the form

L **is observed if** $L_1, \ldots, L_m$ **with absence** $\bar{L}, L_{m+1}, \ldots, L_n$ **after** $\bar{A}$,

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

where $\bar{A}$ is not an empty string of actions. Then the following results hold:

(i) If $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models \neg Holds(F,S)$ and $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Holds(F, Result(A,S))$, then $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Effect^+(F, Result(A,S))$ or $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Caused^+(F, Result(A,S))$;
(ii) If $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Holds(F,S)$ and $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models \neg Holds(F, Result(A,S))$, then $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Effect^-(F, Result(A,S))$ or $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Caused^-(F, Result(A,S))$.

Proof
It is sufficient to only prove (i). As $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models \neg Holds(F,S)$ and $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Holds(F, Result(A,S))$, it is clear that the fact that $Holds(F, Result(A,S))$ is true in each answer set of $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ is *not* due to inertia rules (17), (18), (28) and (29), but due to

(1) some observation rules in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$, or
(2) action effect rules (13) and (14), or
(3) causal rules (9) and (10).

Consider case (1). We suppose there exists some observation rule in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$ of the form

$$Holds(F, Result(A,S)) \leftarrow [\neg]Holds(F_1,S),\ldots,[\neg]Holds(F_m,S), \\ not\ [\neg]Holds(F_{m+1},S),\ldots,not\ [\neg]Holds(F_n,S),$$

But from the condition, we know that the above observation rule must be of the form:

$$Holds(F, Result(A,S)) \leftarrow \cdots, not \neg Holds(F, Result(A,S)),\ldots.$$

This results in a conflict with inertia rule (18):

$$Holds(f, Result(a,s)) \leftarrow Holds(f,s), not\ \neg Holds(f, Result(a,s))$$

in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$. As we specify inertia rules have higher priorities than observation rules in $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D})$, it turns out that $\neg Holds(F, Result(A,S))$ is derived. So case (1) is impossible. Hence, only cases (2) or (3) is possible to derive $Holds(F, Result(A,s))$. That is, $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Effect^+(F, Result(A,S))$ or $\mathscr{P}^{\mathscr{AT}^1}(\mathscr{D}) \models Caused^+(F, Result(A,S))$. □

Theorem 6
Let $\mathscr{D}$ be a domain description of $\mathscr{AT}^2$ and $\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D})$ be its PLP translation. Suppose each observation proposition in $\mathscr{D}$ has the form

L **is observed if** $L_1,\ldots,L_m$ **with absence** $\bar{L}, L_{m+1},\ldots,L_n$ **after** $\bar{A}$,

where $\bar{A}$ is not an empty string of actions. Then the following results hold:

(i) If $\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models \neg Holds(F,S)$ and $\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Holds(F, Result(A,S))$, then one of following results Holds:
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Effect^+(F, Result(A,S))$;
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models AbEffect^+(F, Result(A,S))$; or
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Caused^+(F, Result(A,S))$;

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

(ii) If $\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Holds(F,S)$ and $\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models \neg Holds(F, Result(A,S))$, then one of following results holds:
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Effect^-(F, Result(A,S))$;
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models AbEffect^-(F, Result(A,S))$; or
$\mathscr{P}^{\mathscr{AT}^2}(\mathscr{D}) \models Caused^-(F, Result(A,S))$.

Proof
The proof of Theorem 6 is similar to that of Theorem 5, as described above. □

Theorem 7
Let $\mathscr{D}$ be a domain description $\mathscr{AT}^0$. $\mathscr{D}$ is O-monotonic if

(i) each causal proposition in $\mathscr{D}$ is of the form

L **is caused if** $L_1, \ldots, L_m$, *and*

(ii) $\mathscr{F}^+_{Initial} \cap (\mathscr{F}^-_{Effect} \cup \mathscr{F}^-_{Caused}) = \emptyset$, $\mathscr{F}^-_{Initial} \cap (\mathscr{F}^+_{Effect} \cup \mathscr{F}^+_{Caused}) = \emptyset$, and $(\mathscr{F}^+_{Effect} \cup \mathscr{F}^+_{Caused}) \cap (\mathscr{F}^-_{Effect} \cup \mathscr{F}^-_{Caused}) = \emptyset$.

Proof
Let $\mathscr{D}'$ be an augment of $\mathscr{D}$, $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$ and $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}')$ be the PLP translations of $\mathscr{D}$ and $\mathscr{D}'$ respectively. To prove the result, it is sufficient to prove that $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F,S)$ implies $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}') \models Holds(F,S)$. From the construction of $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, it is clear that $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F,S)$ implies

(1) $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Effect^+(F,S)$,
(2) $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Caused^+(F,S)$, or
(3) $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F,S')$ due to inertia rules in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, where $S = Result(A,S')$.

Adding more observation propositions into $\mathscr{D}$ to form $\mathscr{D}'$, the new program $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}')$ then may have the following effects:

(a) initiating some action effect rules in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$;
(b) initiating some casual rules in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$;
(c) defeating some casual rules in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$;
(d) not initiating any action effect and causal rules in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$.

First, since each causal proposition in $\mathscr{D}$ has the form

L **is caused if** $L_1, \ldots, L_m$,

this follows that each corresponding causal rule in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$ is non-defeasible, i.e. no negation as failure sign *not* is included in the body. Hence, the effect (c) will not be presented. On the other hand, since both causal rules and action effect rules are non-defeasible in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, it is clear that initiating more action effect rules or causal rules in $\mathscr{AT}^0$ will *not* affect the truth values of literals $Effect^+(F,S)$ and $Caused^+(F,S)$ if $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Effect^+(F,S)$ and $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Caused^+(F,S)$, respectively.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Now suppose $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F,S)$ is due to some inertia rule in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$:

$$N : Holds(F,S) \leftarrow Holds(F,S'), not \neg Holds(F,S),$$

where $S = Result(A,S')$, and $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F,S')$. We prove $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}') \models Holds(F,S)$.

Case 1. Suppose $S = S_0$ and $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}') \models Holds(F,S_0)$. Since no inertia rule is needed to drive $Holds(F,S_0)$, the only possibility to have $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}') \models Holds(F,S_0)$ is either $Holds(F,S_0) \leftarrow$ is in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}')$, or $Caused^+(F,s) \leftarrow \cdots$ is in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}')$ (note that such causal rule is non-defeasible). Obviously, in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}')$, the truth value of $Holds(F,S_0)$ will not be affected. Hence the Result holds.

Case 2. Now consider the case that S is not the initial situation. Suppose $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F,S)$. It implies that there exists some action constant A such that $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F,S')$ due to the inertia rule in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D})$, where $S = Result(A,S')$.

Now suppose $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}') \not\models Holds(F,S)$. So the inertia rule:

$$N' : Holds(F,S) \leftarrow Holds(F,S'), not \neg Holds(F,S),$$

where $S = Result(A,S')$, is defeated in $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}')$. Hence it must be the case that $F \in (\mathscr{F}^-_{Effect} \cup \mathscr{F}^-_{Caused})$. On the other hand, from the fact that that $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}) \models Holds(F,S')$, it follows that $F \in \mathscr{F}^+_{Initial}$ or $F \in (\mathscr{F}^+_{Effect} \cup \mathscr{F}^+_{Caused})$. But this contradicts conditions of Theorem 7. So it must have $\mathscr{P}^{\mathscr{AT}^0}(\mathscr{D}') \models Holds(F,S)$. □

Lemma 1

A domain description $\mathscr{D}$ of language $\mathscr{AT}^i$ $(i = 0,1,2)$ is temporally definite if its PLP translation $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ has a unique answer set.

Proof

Let $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ be the PLP translation of $\mathscr{D}$. From the definition of temporal definiteness, it is sufficient to prove that $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D}) \models Holds(F,S)$ implies $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D}) \models Holds(F,Result(A,S))$ or $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D}) \models \neg Holds(F,Result(A,S))$ for any action constant A. Suppose $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D}) \models Holds(F,S)$ and $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ has a unique answer set Ans^i. So $Holds(F,S) \in Ans^i$. Then, it is clear that if one of the following cases holds, the result is true:

(1) $Effect^+(F,Result(A,S))$ or $Effect^-(F,Result(A,S))$ is in Ans^i;
(2) $AbEffect^+(F,Result(A,S))$ or $AbEffect^-(F,Result(A,S))$ is in Ans^i, here $i = 2$;
(3) $Caused^+(F,Result(A,S))$ or $Caused^-(F,Result(A,S))$ is in Ans^i;
(4) $\neg Holds(F,Result(A,S))$ is in Ans^i.

Now suppose none of the above cases is held. Then from the instance of inertia rule in $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$:

$$Holds(F,Result(A,S)) \leftarrow Holds(F,S), not \neg Holds(F,Result(A,S)),$$

it follows that $Holds(F,Result(A,S))$ is in Ans^i. So the result is still true. □

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Theorem 8
A domain description $\mathscr{D}$ of $\mathscr{AT}^i$ $(i = 0, 1, 2)$ is temporally definite if its PLP translation $\mathscr{P}^{\mathscr{AT}^i}(\mathscr{D})$ has a unique reduct Δ^i and $Trans(\Delta^i)$ is locally stratified.

Proof
The proof is directly from Lemma 1 and Proposition 3 in Appendix A. □

References

Apt, K. R. and Bol, R. N. (1994) Logic programming and negation: A survey. *Journal of Logic Programming*, **19**, **20**, 9–71.

Baral, C. and Lobo, J. (1997) Defeasible specifications in action theories. *Proceedings 15th International Joint Conference on Artificial Intelligence* (*IJCAI'97*), pp. 1441–1446. Morgan.

Brewka, G. (1996) Well-founded semantics for extended logic programs with dynamic preferences. *Journal of Artificial Intelligence Research*, **4**, 19–36.

Brewka, G. and Eiter, T. (1999) Preferred answer sets for extended logic programs. *Artificial Intelligence*, **109**, 297–356.

Eshghi, K. and Kowalski, R. (1989) Abduction compared with negation as failure. *Proceedings Sixth International Conference on Logic Programming*, pp. 234–255. MIT Press.

Geffner, H. (1997) Causality, constraints and the indirect effects of actions. *Proceedings 15th International Joint Conference on Artificial Intelligence* (*IJCAI'97*), pp. 555–560. Morgan Kaufmann.

Gelfond, M. and Lifschitz, V. (1988) The stable model semantics for logic programming. *Proceedings Fifth Joint International Conference and Symposium*, pp. 1070–1080. MIT Press.

Gelfond, M. and Lifschitz, V. (1991) Classical negation in logic programs and disjunctive databases. *New Generation Computing*, **9**, 365–386.

Gelfond, M. and Lifschitz, V. (1993) Representing action and change by logic programs. *Journal of Logic Programming*, **17**, 301–322.

Goldszmidt, M. and Pearl, J. (1992) Rank-based systems. *Proceedings of KR'92*, pp. 661–672. Morgan Kaufmann.

Grosof, B. N. (1997) Prioritized conflict handling for logic programs. *Proceedings International Logic Programming Symposium* (*ILPS'97*), pp. 197–212.

Jabłonowski, J., Łukaszewcz, W. and Madalińsks-Bugaj, E. (1996) Reasoning about action and change: Defeasible observations and actions with abnormal effects. *Proceedings of KI-96*, pp. 136–147.

Lifschitz, V. and Turner, H. (1994) Splitting a logic program. *Proceedings 11th International Conference on Logic Programming*, pp. 23–37. MIT Press.

Marek, V. W. and Truszczyński, M. (1993) *Nonmonotonic Logic: Context-dependent reasoning*. Springer-Verlag.

McCain, N. and Turner, H. (1995) A causal theory of ramifications and qualifications. *Proceedings 14th International Conference on Artificial Intelligence* (*IJCAI-95*), pp. 1978–1984. Morgan Kaufmann.

Reiter, R. (1980) A logic for default reasoning. *Artificial Intelligence*, **13**, 81–132.

Sandewall, E. (1994) *Features and Fluents: The representation of knowledge about dynamical systems*. Oxford Science Publications.

W. Łukaszewcz, W. and Madalińsks-Bugaj, E. (1995) Reasoning about action and change using Dijkstra's semantics for programming language: Preliminary report. *Proceedings 14th International Joint Conference on Artificial Intelligence* (*IJCAI'95*), pp. 1950–1955. Morgan Kaufmann.

https://doi.org/10.1017/S1471068402001606 Published online by Cambridge University Press

Winslett, M. (1988) Reasoning about action using a possible models approach. *Proceedings 17th National Conference on Artificial Intelligence* (*AAAI'88*), pp. 89–93. Morgan Kaufmann.

Zhang, Y., Wu, C.-M. and Bai, Y. (2001) Implementing prioritized logic programming. *Artificial Intelligence Communications*, **14**, 183–196.

Zhang, Y. (1999) Specifying causality in action theories: A default logic approach. *Theoretical Computer Science*, **228**, 489–513.

Zhang, Y. (2001) The complexity of logic program update. *Proceedings 14th Australian Joint Conference on Artificial Intelligence* (*AI2001*), pp. 630–643. Springer.

Zhang, Y. and Foo, N. Y. (1997a) Answer sets for prioritized logic programs. *Proceedings International Logic Programming Symposium* (*ILPS'97*), pp. 69–83. MIT Press.

Zhang, Y. and Foo, N. Y. (1997b) Towards generalized rule-based updates. *Proceedings 15th International Joint Conference on Artificial Intelligence* (*IJCAI'97*), pp. 82–88. Morgan Kaufmann.

Zhang, Y. and Foo, N. Y. (1998) Updating logic programs. *Proceedings 13th European Conference on Artificial Intelligence* (*ECAI'98*), pp. 403–407. Wiley.