Causal Action Theories and Satisfiability Planning

by

Charles Hudson Turner, B.A., M.L.I.S., M.S.C.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 1998

Causal Action Theories and Satisfiability Planning

Approved by
Dissertation Committee:

_____

_____

_____

_____

_____

This dissertation is dedicated to my wife, Carol George,
and my parents, Charles and Clarice Turner.

Acknowledgments

I am deeply grateful to Vladimir Lifschitz, who, among other things, made this work possible for me, and to Norm McCain, who carefully and patiently discussed with me many of the ideas as they took shape. I am grateful as well for the help and encouragement of many other friends, teachers and colleagues.

Charles Hudson Turner

The University of Texas at Austin
August 1998

Causal Action Theories and Satisfiability Planning

Publication No. _____

Charles Hudson Turner, Ph.D.
The University of Texas at Austin, 1998

Supervisor: Vladimir Lifschitz

This dissertation addresses the problem of representing and reasoning about commonsense knowledge of action domains. Until recently, most such work has suppressed the notion of causality, despite its central role in everyday talking and reasoning about actions. There is good reason for this. In general, causality is a difficult notion, both philosophically and mathematically. Nonetheless, it turns out that action representations can be made not only more expressive but also mathematically simpler by representing causality more explicitly. The key is to formalize only a relatively simple kind of causal knowledge: knowledge of the conditions under which facts are caused. In the first part of the dissertation we do this using inference rules and rule-based nonmonotonic formalisms. As we show, an inference rule $\frac{\phi}{\psi}$ can be understood to represent the knowledge that if ϕ is caused then ψ is caused. (Notice that we do not say "ϕ causes ψ.") This leads to simple and expressive action representations in Reiter's default logic, a rule-based nonmonotonic formalism. This approach also yields action descriptions in logic programming, thus raising the possibility, at least in principle, of automated reasoning about actions and planning. In the second part of the dissertation, we introduce a new modal non-

monotonic logic—the logic of "universal causation" (UCL)—specifically designed for describing the conditions under which facts are caused. We show that UCL provides a more traditional semantic account of the mathematically simple approach to causal knowledge that underlies our causal theories of action. For instance, instead of the inference rule $\frac{\phi}{\psi}$, we write the modal formula $\mathsf{C}\phi \supset \mathsf{C}\psi$, where C is a modal operator read as "caused." In the third part of the dissertation, we show that a subset of UCL is well-suited for automated reasoning about actions. In particular, we show that the class of "simple" UCL theories provides an expressive basis for the computationally challenging task of automated planning. Simple UCL theories have a concise translation into classical logic, and, as we show, the classical models of the translation correspond to valid plans. This enables "satisfiability planning" with causal action theories, with "state of the art" performance on large classical planning problems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

John McCarthy in his 1959 paper "Programs with Common Sense" [McC59] proposed that researchers in artificial intelligence try to formalize and automate commonsense reasoning about actions. The challenge is to obtain correct conclusions about the outcomes of actions on the basis of concise declarative representations of commonsense knowledge about action domains. This has proved difficult.

It is widely remarked that the notion of causality plays little or no role in descriptions of the world in the physical sciences. The same has been generally true of proposed formalizations of reasoning about action, despite the central role played by causal notions in everyday discourse and thought about actions. This dissertation belongs to a line of recent work investigating the advantages of considering causality more explicitly.

1.1 The Frame Problem and Nonmonotonicity

A fundamental difficulty in reasoning about action—the so-called "frame problem"—was recognized and named by McCarthy and Hayes in their 1969 paper "Some Philosophical Problems from the Standpoint of Artifical Intelligence" [MH69]. A

natural strategy for making action representations concise is to focus on describing the changes caused by an action, while leaving implicit our knowledge of facts unaffected by the action. About facts unaffected by an action, we assume that they simply persist, according to a "commonsense law of inertia." Thus, generally speaking, the frame problem is the problem of making the commonsense law of inertia mathematically precise.

It is clear that solutions to the frame problem will be nonmonotonic: that is, in contrast to classical logic, conclusions may be lost when premises are added. For example, consider an action domain description involving two propositional fluents, P and Q, and a single action A. (A propositional fluent is a proposition whose value depends on time.) Suppose you are told that P and Q are initially false, and that A makes P true. You are expected to conclude not only that P would become true if A were performed, but also that Q would not. Now suppose that you are told in addition that A makes Q true. You should no longer conclude that Q would be false after A; instead, Q would be true after A.

Although the previous informal example demonstrates that solutions to the frame problem will be nonmonotonic, it does little to suggest that such solutions may be subtle or difficult to find. Nonetheless, this seems so far to be the case, particularly as we attempt to represent more elaborate kinds of domain knowledge. For instance, in this dissertation we are interested not only in how to represent the "direct" effects of actions, but also in how to represent "background knowledge" concerning relationships between fluents, in order to correctly infer the "indirect" effects of actions. For example, you might be told not only that A makes P true, but also that Q can be made true by making P true. You should again conclude that Q would be true after A.

1.2 Commonsense Inertia as Minimal Change

In most proposals for reasoning about action, the commonsense law of inertia is understood according to a principle of minimal change. Roughly speaking, the idea is to capture the assumption that things change as little as possible, while also reflecting our knowledge of what does change.

1.2.1 The Yale Shooting Problem

In 1986 McCarthy proposed a formalization of commonsense knowledge about actions in which the commonsense law of inertia is understood according to a principle of minimal change [McC86]. Essentially, McCarthy said that change is abnormal, and he used technical means—namely, circumscription (introduced in [McC80])—to select models of his action theory in which that kind of abnormality is minimal. That is, he preferred models in which things change as little as possible.

Hanks and McDermott famously exposed a fundamental difficulty with McCarthy's proposal, by introducing a counterexample widely known as the "Yale Shooting" domain [HM87]. The essential elements can be described as follows. There is a turkey (Fred) and a gun. If the gun is loaded, shooting it kills Fred. The question is this: If Fred is initially alive and the gun is initially loaded, will Fred be dead after the actions *Wait* and *Shoot* are performed in sequence? Clearly the answer should be yes. Unfortunately, McCarthy's formalization could not predict this.

The fundamental difficulty with McCarthy's 1986 proposal is that it minimizes change globally (i.e. across all situations). In the intended models of the Yale Shooting domain, no fluents change as a result of the *Wait* action—in particular, the gun remains loaded—and then Fred becomes dead as a result of the *Shoot* action. McCarthy calls the death of Fred abnormal, and is in principle willing to trade the death of Fred for other possible abnormalities. Thus, the global minimization policy

is satisfied by anomalous models in which the gun becomes unloaded as a result of the *Wait* action, and then no fluents change as a result of the *Shoot* action—in particular, Fred remains alive after the *Shoot* action.[1]

This account of the Yale Shooting problem suggests that it is wrong to minimize change globally, but does not show that the principle of minimal change will never do.

1.2.2 Possible Next States

The principle of minimal change can carry us a long way if applied more carefully. The essence of the frame problem can be formulated as follows. Given an initial state of the world and a description of the effects of an action when performed in that state, we must say which states of the world may result—so far as we know—after the action is performed. We say a definition of this kind identifies "possible next states." Winslett [Win88] proposed a definition of possible next states in which the commonsense law of inertia is captured mathematically as the straightforward requirement that things change as little as possible (while still satisfying the effects of the action).

This simple idea (in various guises) has led to considerable progress. In fact, it seems that a good deal of the widely-remarked technical difficulty in work on reasoning about action can be attributed to the need to find mathematical means for capturing this simple definition of possible next states within descriptions that are more complex primarily because they encompass more than an initial situation, an action and a resulting situation. (See for example [Bak91], or Chapter 4 of this dissertation.)[2]

[1] According to the informal description given here, there will be another class of models, in which Fred dies as a result of the *Wait* action (while the gun remains loaded). As it happens, this kind of anomalous model is ruled out in McCarthy's style of formalization, which forces an abnormality with respect to *Alive* whenever *Shoot* is performed with a loaded gun, *even* if Fred is already dead.

[2] In passing we remark that Winslett also emphasized a second crucial element in the possible next states setting. Reasoning about action is by its nature a matter of reasoning about complete states of the world. Katsuno and Mendelzon emphasize a similar point in their influential paper "On the Difference Between Updating a Knowledge Base and Revising It" [KM91]. We will allude to this point several times in this dissertation, since it also helps explain technical difficulties encountered in some formalizations of actions.

1.3 State Constraints and Static Causal Laws

State constraints are formulas of classical logic that are said to hold in every possible state of an action domain. Traditionally, state constraints have been used to derive indirect effects, or "ramifications," of actions. Adapting a widely familiar example (from [Bak91]), let's assume that you can make Fred stop walking by making him not alive. So if shooting Fred kills him, you can make him stop walking by shooting him: not walking is a ramification of shooting. Traditionally, the background knowledge used in this example has been expressed by the state constraint

$$\neg \mathit{Alive} \supset \neg \mathit{Walking}\,. \tag{1.1}$$

This state constraint is equivalent in classical logic to its contrapositive

$$\mathit{Walking} \supset \mathit{Alive}\,.$$

This is troublesome because it is clear that the causal relation itself is not contrapositive. That is, roughly speaking, although you can make Fred stop walking by killing him, it does not follow from this that you can bring Fred back to life by making him walk.

Recently, a number of researchers have argued that state constraints are inadequate for representing background knowledge in action domains, because they do not adequately represent causal relations [Gef90, Elk92, BH93, Bar95, Lin95, MT95b, Thi95a, Gus96]. In this dissertation we explore the use of "static causal laws" of the kind introduced by McCain and Turner in [MT95b]: if a fluent formula ϕ is caused to be true, then a fluent formula ψ is also caused to be true. From such a causal law it follows that one can make ψ true by making ϕ true. It also follows that in every possible state, ψ is true if ϕ is. That is, the state constraint $\phi \supset \psi$

holds. On the other hand, it does not follow that if $\neg\psi$ is caused to be true, then $\neg\phi$ is also caused to be true. Thus, the static causal law is not contrapositive. In this dissertation we argue for the usefulness of propositions that represent such static causal laws, and we show that such propositions are more expressive than traditional state constraints.

In the example involving Fred, we can express the relevant causal background knowledge by means of a static causal law: if Fred is caused to be not alive, he is also caused to be not walking. In the treatment of static causal laws investigated in the first part of this dissertation (as in [MT95b, Tur97]), the logical properties of static causal laws are captured mathematically through the use of inference rules. In particular, the failure of contraposition in static causal laws is reflected in the noncontrapositive nature of inference rules. Thus, in the first part of the dissertation, we (essentially) replace the state constraint (1.1) with the inference rule

$$\frac{\neg \mathit{Alive}}{\neg \mathit{Walking}} \tag{1.2}$$

which says that from $\neg \mathit{Alive}$ you can derive $\neg \mathit{Walking}$. The inference rule is noncontrapositive: it does not say that you can derive *Alive* from *Walking*.

1.4 A Causal Account of Commonsense Inertia

In the presence of state constraints it is still possible to solve the frame problem by applying the principle of minimal change. In fact, Winslett's classic definition of possible next states already allows for state constraints, requiring that resulting states differ as little as possible from the initial state, while satisfying both the direct effects of the action and all state constraints. But in the presence of static causal laws, we need an understanding of the commonsense law of inertia that takes causal notions into account more explicitly.

Recently, Marek and Truszczyński [MT94, MT95a, MT98a] introduced a

formalism they call "revision programming," which defines possible next states on the basis of an essentially causal understanding of the commonsense law of inertia: they call it a principle of "justified change." Moreover, in revision programming the causal knowledge is expressed by means of what are essentially inference rules $\frac{\phi}{\psi}$, under the restriction that both ϕ and ψ are simple conjunctions (conjunctions of literals). Thus, in our terminology, they solve the frame problem in the presence of a restricted subclass of static causal laws. As evidence of the reasonableness of their definition, Marek and Truszczyński show that it agrees with Winslett's where the two overlap—taking a conjunction ϕ of literals to correspond to the inference rule $\frac{\mathit{True}}{\phi}$.

In the first part of this dissertation, we employ a causal definition of possible next states that is applicable in the presence of arbitrary static causal laws, expressed by means of (arbitrary) inference rules, following [MT95b, PT95, PT97, Tur97]. As evidence of the reasonableness of our definition, we note that it extends, and unifies, the definitions of Winslett and of Marek and Truszczyński.

Our solution to the frame problem is based on a fixpoint condition that makes mathematically precise a causal understanding of the commonsense law of inertia. Intuitively speaking, we capture the idea that things change only when they're made to. But we capture this idea indirectly, roughly as follows. We first impose the requirement that every fact in the resulting situation have a cause according to our description. In particular then, since we describe the changes caused by actions, those changes will have a cause according to our description. On the other hand, we build in the assumption that every fact that persists is caused. Therefore, facts that persist need no (additional) explanation. As a result, in effect, it is precisely the facts that change that must be explained by our description. That is, things don't change unless (our description tells us that) they're made to.

Also in the first part of this dissertation, we show how to express the com-

monsense law of inertia in the rule-based nonmonotonic formalisms of default logic [Rei80] and logic programming. Here we confront the technical difficulty, previously discussed, of honoring our causal definition of possible next states while working with descriptions that encompass more than an initial situation, an action and a resulting situation. And true to form, we end up expending considerable mathematical effort verifying that our default theories and logic programs indeed are correct with respect to our definition of possible next states. Nonetheless, the descriptions themselves are relatively straightforward. In particular, the default rules expressing the commonsense law of inertia have a remarkably simple form. Essentially, we write default rules of the form

$$\frac{F_t : F_{t+1}}{F_{t+1}} \tag{1.3}$$

where F_t says that a fluent F is true at time t and F_{t+1} says that F is true at time $t+1$.[3] The default rule (1.3) can be understood to say that if F is caused to be true at time t and remains true at time $t+1$, then it is caused to be true at time $t+1$.

1.5 Causally Possible Worlds and Universal Causation

The second part of the dissertation discusses a new modal nonmonotonic logic of "universal causation," called UCL, designed specifically for formalizing commonsense knowledge about actions. This logic was introduced in [Tur98]. UCL extends the recently introduced causal theories formalism of McCain and Turner [MT97], which shares its underlying motivations. The fundamental distinction in UCL—

[3]Inertia rules of essentially this form have been entertained previously in the literature [Rei80, MS88], but without substantial success. The specific proposals are not complete enough to analyze in any detail. We do discuss throughout the first part of the dissertation several interacting factors that contribute to the success of our approach. One such factor is that, while default logic is designed to deal with incompleteness—its models ("extensions") are logically closed sets of formulas, not classical interpretations—it is important in reasoning about action that we focus on complete worlds, as previously mentioned. Our default theories guarantee this kind of completeness, in contrast to previous published proposals in default logic.

between facts that are caused and facts that are merely true—is expressed by means of the modal operator C, read as "caused." For example, one can write $\phi \supset \mathsf{C}\psi$ to say that ψ is caused whenever ϕ is true. These simple linguistic resources make it possible for a UCL theory to express the conditions under which facts are caused. It is in this sense that UCL is a logic of causation.

In UCL, we can express the commonsense law of inertia by writing, for instance

$$\mathsf{C}F_t \wedge F_{t+1} \supset \mathsf{C}F_{t+1} \tag{1.4}$$

which corresponds closely to the default rule (1.3) for inertia discussed previously. Formual (1.4) stipulates that F is caused at time $t+1$ whenever it is caused at time t and persists from time t to time $t+1$.

Typical features of action domain descriptions are easily expressed in UCL. For instance, in a variant of the Yale Shooting domain [HM87] as extended by Baker [Bak91], one can write

$$\mathit{Shoot}_t \supset \mathsf{C}\neg \mathit{Alive}_{t+1} \wedge \mathsf{C}\neg \mathit{Loaded}_{t+1} \tag{1.5}$$

to describe the direct effects of shooting: whenever *Shoot* occurs, both $\neg \mathit{Alive}$ and $\neg \mathit{Loaded}$ are caused to hold subsequently. One can write

$$\mathit{Shoot}_t \supset \mathit{Loaded}_t \tag{1.6}$$

to express a precondition of the shoot action: shoot can occur only when the gun is loaded. To say that Fred is caused to be not walking whenever he is caused to be not alive, one can write the UCL formula

$$\mathsf{C}\neg \mathit{Alive}_t \supset \mathsf{C}\neg \mathit{Walking}_t\,. \tag{1.7}$$

This formula corresponds closely to the inference rule (1.2) considered previously. Notice that from (1.5) and (1.7) it follows by propositional logic that whenever

Shoot occurs, $\neg Walking$ is caused to hold subsequently. Thus (1.7) correctly yields $\neg Walking$ as an indirect effect, or ramification, of the *Shoot* action.

In accordance with common sense, (1.7) does not imply $\mathsf{C}\,Walking_t \supset \mathsf{C}\,Alive_t$. Intuitively, you cannot bring Fred back to life by getting him to walk. Instead, he simply can't walk unless he's alive. To put it another way, in any causally possible world, Fred is alive if he is walking. Accordingly, if (1.7) is an axiom of a UCL theory T, then $Walking_t \supset Alive_t$ is true in every interpretation that is causally explained by T.

1.6 Automated Reasoning about Actions and Satisfiability Planning

One of the purposes of formalizing commonsense knowledge about actions is to enable the automation of reasoning about actions. There is a subset of propositional UCL theories extremely well-suited to this purpose, in which every formula has either the form

$$\phi \supset \mathsf{C}L$$

or

$$\phi$$

where ϕ is a formula in which the modal operator C does not occur and L is a literal. Such UCL theories are called "definite." There is a concise translation of definite UCL theories into classical propositional logic. Thus, standard automated reasoning tools for classical logic can be applied to definite UCL theories. This possibility is notable, but does not constitute a primary contribution of this dissertation. We focus instead on the problem of automated planning.

Planning is an automated reasoning task associated with action domains that has been especially well-studied, due to its potential practical utility and its

fundamental computational difficulty, even for action domains that can be described quite simply, as in STRIPS [FN71, Lif87b]. In this dissertation, following [MT98b], we describe an implemented approach to satisfiability planning [KS92, KS96], in which a plan is obtained by "extracting" the action occurrences from a suitable model of a classical propositional theory describing the planning domain. In our approach, the description in classical logic is obtained by translation from a definite UCL theory.

This approach to planning is noteworthy for two reasons. First, it is based on a formalism for describing action domains that is more expressive than the STRIPS-based formalisms traditionally used in automated planning. Secondly, our experiments suggest that the additional expressiveness of causal theories comes with no performance penalty in satisfiability planning. Specifically, we show that the large blocks world and logistics planning problems used by Kautz and Selman [KS96] to demonstrate the effectiveness of satisfiability planning can be conveniently represented as UCL theories and solved in times comparable to those that they have obtained.

1.7 Outline of Dissertation

Chapter 2 consists of a review of the literature, and extends some of the themes raised thus far.

The first part of the dissertation (Chapters 3 and 4) is devoted to demonstrating the extent to which inference rules, and rule-based formalisms such as default logic, can be convenient for formalizing causal aspects of commonsense knowledge about actions. We introduce, and investigate in detail, an approach to representing actions in which inference rules play a central role. More specifically, we define a high-level action language, called $\mathcal{AC}$, and we show how to translate $\mathcal{AC}$ into the rule-based nonmonotonic formalisms of default logic and logic programming. This

line of investigation is motivated by the simple observation that the noncontrapositive nature of causal relationships is nicely reflected in the noncontrapositive nature of inference rules. More precisely, we show that an inference rule

$$\frac{\phi}{\psi} \tag{1.8}$$

can be understood to represent the corresponding static causal law: if ϕ is caused then ψ is caused. In this way we obtain causal theories of action based on familiar and well-understood mathematical tools.

In the second part of the dissertation (Chapter 5) we introduce a new modal nonmonotonic logic—called UCL— in which the appropriate (mathematically simple) causal notions can be represented by means of a modal operator C (read as "caused") along with standard truth-functional connectives. In this logic, we say that a formula ϕ is caused to be true by writing the modal formula $\mathsf{C}\phi$. Thus, for instance, instead of the inference rule (1.8), we can write the UCL formula

$$\mathsf{C}\phi \supset \mathsf{C}\psi\,. \tag{1.9}$$

We discuss the fact that UCL can be understood to provide a more general, alternative account of the causal notions that underlie the work presented in the first part of the dissertation. We also introduce a different, but closely related, approach to formalizing commonsense knowledge about actions, directly in UCL. In order to help clarify the relationship between this new nonmonotonic logic and well-known general-purpose nonmonotonic formalisms, we present several theorems concerning translations back and forth from default logic [Rei80], autoepistemic logic [Moo85], and circumscription [McC80]. We also relate action formalizations in UCL to the circumscriptive causal action theories of Lin [Lin95, Lin96].

In the third and last part of the dissertation (Chapter 6), we develop mathematical results that justify planning on the basis of a restricted class of UCL action

descriptions, and show that this approach to planning is relatively effective on large classical planning problems.

Chapter 7 consists of a brief summary of the dissertation and a few remarks concerning possible directions of future work.

Chapter 2

Literature Survey

> The common knowledge about the world that is possessed by every schoolchild and the methods for making obvious inferences from this knowledge are called common sense.
>
> – Ernest Davis [Dav90]

2.1 The Situation Calculus

As mentioned in the previous chapter, McCarthy and Hayes recognized and named the frame problem in [MH69]. In that paper they also introduced the "situation calculus," which can be understood as both a simple ontology for models of action domains and a family of convenient notational conventions for representing theories about action. The essential elements and assumptions of the situation calculus ontology can be described roughly as follows.

- There are properties of the world that vary with time, called "fluents."
- A world at a moment in time is called a "situation."
- Each situation maps each fluent to a value.

- For each situation S and action A, there is a unique situation, $Result(A, S)$, that would result if the action A were performed in situation S.

In addition, it is common to assume that there is a distinguished "initial situation." In fact, in situation calculus settings, the model structures of action domains can be understood essentially as trees in which nodes correspond to situations and edges correspond to actions, with the initial situation at the root. We can complete our description of such a situation calculus model structure by associating with each situation an interpretation of the set of fluents. That is, each situation is mapped to a state. Notice that in such a model, each situation can be uniquely specified by the sequence of actions that, when performed in the initial situation, would lead to it. This lends itself naturally to the following notational conventions. First, we denote the initial situation by a constant S_0. We then denote the situation that would result from doing, for instance, action A_1 followed by action A_2, in the initial situation, by the term $Result(A_2, Result(A_1, S_0))$. Given this, we say, for instance, that a fluent F is true in that situation by writing $Holds(F, Result(A_2, Result(A_1, S_0)))$, and we say that it is false in that situation by writing $\neg Holds(F, Result(A_2, Result(A_1, S_0)))$.

In the first part of this dissertation, we propose a high-level language for representing actions that reflects a situation calculus ontology. Furthermore, our translations of this high-level action language into default logic and logic programming utilize standard syntactic conventions for representing situation calculus theories in a many-sorted, first-order language. These are convenient choices for several interrelated reasons. First, the situation calculus ontology is appealing in its simplicity. For instance, it does not address the problem of representing the duration of actions. In fact, the passage of time is reflected only in the state changes associated with discrete action occurrences. Second, the syntactic conventions of the situation calculus are themselves simple and convenient. Third, in light of the first two observations, it is not surprising that there has been a great deal of previous work on

representing actions in the situation calculus. Thus, the situation calculus is widely familiar.

There have been, of course, many other proposals of ontologies and notational conventions for theories of action. A discussion of such proposals is beyond the scope of this literature survey. However, because this dissertation includes a proposal for representing actions in logic programming, we mention that there is a considerable body of work on formalizing actions in logic programming that is based on the "event calculus" of Kowalski and Sergot [KS86]. As with the situation calculus, the event calculus can be understood as both a simple ontology for models of action domains and a family of convenient notational conventions for representing theories about action. Recently there have been a number of papers investigating relationships between the event calculus and situation calculus [PR93, KS94, Mil95, vBDS95].

In the second and third parts of this dissertation, we employ a simple alternative ontology and notation, in which time has the structure of the natural numbers, and in which the occurrence of an action at a time becomes a proposition. One advantage of this approach is that it makes it more convenient to represent and reason about concurrent actions. Another advantage of this alternative approach is related to a complication in the situation calculus that has been suppressed to this point in our discussion—in a situation calculus model, there may be situations that are, intuitively speaking, unreachable. That is, as the tree-like model structure "unfolds" we may reach a situation in which some action simply cannot be performed, given what is true in that situation. This renders "unreachable" the situation that would result from performing the action. Nonetheless, our model will satisfy sentences that describe the state of the world in that unreachable situation. For instance, even if S names such an unreachable situation (about which it intuitively makes no sense to say that a fluent holds or does not hold), the sentence $\forall f(Holds(f, S) \vee \neg Holds(f, S))$ is logically valid. This technical, and conceptual,

difficulty can be dealt with, at the cost of additional complexity.[1] By contrast, in the alternative approach utilized in the latter parts of the dissertation, the fact that a certain action cannot be performed in a certain situation is, roughly speaking, reflected in the fact that there is simply no model in which that action occurs in such a situation.

2.2 Nonmonotonic Formalisms

Representational difficulties in commonsense reasoning—such as the need to represent the nonmonotonicity inherent in the commonsense law of inertia—led in the 1980's to the introduction of several nonmonotonic formalisms. In 1980, default logic was defined by Reiter [Rei80], McCarthy introduced cirumscription [McC80], and a modal nonmonotonic logic was proposed by McDermott and Doyle [MD80]. In 1985, Moore introduced a particularly influential modal nonmonotonic logic, called autoepistemic logic [Moo85]. Circumscription has undergone a great deal of refinement and extension, some of which is reflected in [Lif85, McC86, Lif91, Lif95]. Autoepistemic logic received a particularly elegant, model-theoretic characterization in [LS93].

Because of the prominent role played by default logic in the first part of this dissertation, we provide at this point an informal introduction to it.[2] A *default rule* is an expression of the form

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma}$$

where all of $\alpha, \beta_1, \ldots, \beta_n, \gamma$ are formulas ($n \geq 0$). Such a rule says, intuitively speaking, if you know α and each of $\beta_1, \ldots, \beta_n$ are consistent with what you know, then conclude γ. A *default theory* is a set of default rules. Its meaning is formally

[1] For instance, a portion of the technical challenge in Chapter 4—where we prove the correctness of a family of situation calculus theories in default logic—can be attributed to this feature of the situation calculus.

[2] The precise definition appears in Section 3.5.1.

determined by the set of its "extensions," which are fixpoints of an equation that reflects the intuition described above. Each extension is a logically closed set of formulas which is also "closed under" the rules of the default theory. A formula is a consequence of a default theory if it belongs to all of its extensions. Default logic is nonmonotonic: we may lose consequences when we add rules to a default theory.

Logic programming is another nonmonotonic formalism which, along with default logic, is of special interest in this dissertation. Logic programming was inspired by Kowalski's 1974 paper "Predicate Logic as a Programming Language" [Kow74]. Because of the "closed world assumption" and the "negation as failure" operator, logic programming has long been recognized as a nonmonotonic formalism, but the appropriate semantics has been a contentious issue. An early proposal by Clark [Cla78], known as "Clark's completion," remains influential, although it suffers from well-known anomalies (as a semantics for logic programs). Other much-studied proposals appear in [Fit85, Kun87, Prz88, VGRS90]. One fruitful line of research investigated connections between the semantics of logic programming and other nonmonotonic formalisms. Of particular interest in relation to this dissertation are the connections to autoepistemic logic [Gel87] and default logic [BF87], which in 1988 led to the proposal by Gelfond and Lifschitz of the "stable model" semantics for logic programs [GL88], later renamed the "answer set" semantics and extended to apply to logic programs with "classical" negation and "epistemic" disjunction [GL90, GL91].

In the first part of this dissertation, we are concerned with logic programs with classical negation under the answer set semantics. Such programs correspond to a simple subset of default logic, as observed in [GL90]. Thus, we will essentially view a logic program rule of the form

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

(where all the L_i's are literals) as an abbreviation for the corresponding default rule

$$\frac{L_1 \wedge \cdots \wedge L_m : \overline{L_{m+1}}, \ldots, \overline{L_n}}{L_0}$$

where for each literal L_i, $\overline{L_i}$ stands for the literal complementary to L_i.[3] The symbol *not* that appears in the bodies of some logic program rules stands for "negation as failure." Roughly speaking, the expression $not\, L$ can be read as "L is not known."

We introduce in the second part of this dissertation a new nonmonotonic formalism, called UCL. The syntax and some of the motivations of UCL are anticipated in a more ambitious formalism introduced by Geffner [Gef89, Gef90, Gef92]. Geffner employs a modal language with a single modal operator C, read as "explained," and defines "default theories which explicitly accomodate a distinction between 'explained' and 'unexplained' propositions" [Gef90]. His proposal is meant to enhance "the appeal of preferential entailment as a unifying framework for non-monotonic inference" by contributing to the development of "a general domain-independent criterion for inferring preferences from theories" [Gef90]. The mathematical complexity of Geffner's definitions may reflect the generality of his goal. By comparison, in UCL both aim and means are modest. It appears that UCL can be embedded in Geffner's formalism, perhaps with some minor technical modifications, but we do not pursue this possibility in this dissertation. The rewards would be minimal, given the differences in emphasis, and in mathematical machinery.

2.3 Action Theories in Logical Formalisms

In 1986 McCarthy published a proposal for reasoning about actions in circumscriptive theories [McC86], discussed in the previous chapter. Partly in response to this proposal, Hanks and McDermott wrote the landmark paper in which they not only exposed a difficulty with McCarthy's proposal, but also showed that an analagous

[3] The precise definition appears in Section 3.6.1.

difficulty could arise in default logic. As we have described, their counterexamples were based on the "Yale Shooting" domain [HM87].[4] Hanks and McDermott were not content to reject McCarthy's specific proposal, but instead went on to argue more generally against the use of nonmonotonic formalisms for reasoning about action. Two of their claims are especially relevant to this dissertation: (i) nonmonotonic formalizations will be too difficult to understand and evaluate, and (ii) nonmonotonic formalisms will not directly capture the appropriate commonsense concepts and patterns of reasoning.

Despite, or perhaps because of, the warnings of Hanks and McDermott, there is a sizable body of work published in the last decade on logic-based, nonmonotonic approaches to representing actions. One such line of work involves proposals for representing actions in classical (monotonic) logic. There are a number of proposals that specify a standard method for generating first-order frame axioms [Ped89, Sch90, Rei91]. In these proposals, there is a standard form for first-order axioms describing the effects of actions and other features specific to a given action domain. The resulting first-order theory is then augmented by additional standard axioms. Finally, there is a procedure for generating frame axioms, based on the domain specific axioms. Because such a procedure depends on the domain specific axioms, the method as a whole is nonmonotonic, in spite of the fact that the resulting action theories are expressed in monotonic logic. The inescapable fact here is the underlying, fundamental nonmonotonicity of the frame problem itself. Any correct solution, taken as a whole, will be nonmonotonic, even if the end product is expressed in a monotonic formalism. Of course this holds, presumptively, for less systematic proposals, which discuss frame axioms for some examples without specifying a general method for generating them [Haa87, Elk92]. Thus the so-called monotonic approaches are conceptually similar to the more clearly nonmonotonic

[4] In Chapter 3.5 we discuss in some detail their default theory for the Yale Shooting domain.

approaches which employ variants of circumscription, autoepistemic logic, default logic and logic programming. Undoubtedly the greatest concentration of such work has been in circumscription [Hau87, Lif87a, LR89, Lif90, Bak91, GLR91, Lif91, LS91, CE92, LR94, KL95, Lin95, Gus96, Lin96]. There were also early published solutions to the Yale Shooting problem in autoepistemic logic [Gel88], default logic [Mor88] and logic programming [EK89, Eva89, AB90].

A primary methodological weakness of much of the work cited above is the fact that it is motivated by, and validated for, only a small set of examples. In many cases, there is no clear claim about what kinds of action domains can be represented correctly by a given approach. Moreover, it is often unclear how the technical means employed are related to the intuitions they are meant to capture.

2.4 High-Level Action Languages

Recently, Gelfond and Lifschitz [GL93] proposed an influential research methodology, which involves the introduction of special-purpose, high-level languages for representing action domains. The advantage of starting with a high-level action language is that it can utilize a restricted syntax and a relatively simple semantics which is nonetheless adequate to capture our commonsense intuitions about the whole family of action domains expressible in the language. Of particular significance in obtaining a simple semantics is the fact that such high-level languages can isolate the problem of defining possible next states, and then deploy such a definition explicitly, in straightforward fashion, to constrain more general model structures (encompassing more than an initial situation, action, and resulting situation).

In [GL93], the high-level action language $\mathcal{A}$ was introduced. Many of the subsequent action languages [BG93, KL94, Thi94, BGP95, GKL95, GL95, Thi95b, Tur96a] are essentially extensions of $\mathcal{A}$. These languages share a situation calculus ontology. Next we briefly describe some of their main features and differences. In

$\mathcal{A}$ actions are deterministic, always executable, and cannot be performed concurrently. Furthermore, fluents are propositional (that is, boolean-valued), and there is no way to represent background knowledge of any kind. In $\mathcal{A}_C$ [BG93], concurrent actions are allowed and actions are no longer required to be always executable. In $\mathcal{A}_{ND}$ [Thi94], actions may be nondeterministic. In the languages $\mathcal{AR}_0$ and $\mathcal{AR}$ [KL94, GKL95, GKL97], actions may be nondeterministic and are not required to be always executable. More importantly, these languages allow the use of state constraints to express background knowledge. Also, these languages employ the "frame/nonframe" distinction introduced in [Lif90], which is discussed briefly in Chapter 3.4 of this dissertation. In addition, $\mathcal{AR}$ allows non-boolean fluents. The language $\mathcal{ARD}$ [GL95] extends $\mathcal{AR}$ by adding the notion of "dependent" fluents. The language $\mathcal{AC}$ [Tur96a, Tur97] that is included in the dissertation essentially extends the propositional portion of $\mathcal{AR}$, but restricts the use of the frame/nonframe distinction, allowing it only for the purpose of introducing "explicit definitions." The main improvement of $\mathcal{AC}$ over propositional $\mathcal{AR}$ is its adoption of the method introduced in [MT95b] for representing causal background knowledge. The language $\mathcal{L}_0$ [BGP95] is an extension of a subset of $\mathcal{A}$, incorporating the notion of "observations" about an actual past, which affords $\mathcal{L}_0$ some of the distinctive expressiveness of the event calculus. The formalism of "dynamic systems" [Thi95b] is an extension of $\mathcal{A}$, incorporating concurrent actions and events, "momentary" fluents and delayed effects of actions.

High-level action languages can be of help in the evaluation of existing proposals for representing commonsense knowledge about actions in general-purpose formalisms. One can demonstrate the (partial) correctness of such a proposal by specifying a correct translation of (some portion of) a high-level language into a general-purpose formalism, using the methods and ideas of the proposal. In [Kar93], Kartha shows that the "monotonic" proposals due to Pednault [Ped89] and Reiter

[Rei91] are in fact correct for action domains expressible in $\mathcal{A}$. He also shows that $\mathcal{A}$ can be embedded in Baker's [Bak91] circumscriptive approach to representing actions. This is an instructive case. Baker's proposal is considerably more expressive than $\mathcal{A}$, allowing the use of state constraints to represent background knowledge. Moreover, Baker's approach has been widely admired, although its correctness has been difficult to assess due to its technical complexity. It turns out, as Kartha demonstrates in [Kar94], that Baker's approach can yield intuitively incorrect results when applied to action domains in which there are nondeterministic actions.

High-level action languages also help generate new proposals for representing actions in general-purpose formalisms, as new translations are developed. For instance, the original paper on $\mathcal{A}$ [GL93] included a sound translation of a portion of $\mathcal{A}$ into logic programming. Now there are sound and complete translations of $\mathcal{A}$ into abductive logic programming [DD93, Dun93], equational logic programming [Thi94] and disjunctive logic programming [Tur94]. In [LT95], we show that the translation from [Tur94] can be transformed into the translation from [DD93] by a series of simple syntactic transformations, obtaining in the process a family of seven translations of $\mathcal{A}$ into logic programming. In [BG93], the language $\mathcal{A}_C$ is given a sound but incomplete translation into logic programming. In [Tur97], we specify a sound and complete translation into logic programming of a portion of the language $\mathcal{AC}$. (This translation is included in the first part of this dissertation.) Subramanian [Sub93] embeds $\mathcal{A}$ in the logic of the Boyer-Moore theorem prover [BM88]. The languages $\mathcal{AR}_0$ and $\mathcal{AR}$ are embedded in circumscriptive theories in [KL94, GKL95]. Finally, in [Tur96a, Tur97] a portion of the language $\mathcal{AC}$ is embedded in default logic. (Again, this translation is included in the first part of this dissertation.)

2.5 Possible Next States and Theory Update

In addition to all of the previously discussed work on representing action domains, there is a body of relevant work in the simpler settings of possible next states and "theory update." As described previously, work on defining possible next states is still explicitly directed toward the problem of reasoning about action, but in a setting where some of the complexities of action domains are ignored in order to focus more directly on the frame problem itself [GS88, MS88, Win88, BH93, Bar95, MT95b, Thi95a]. The frame problem is also confronted in the still more abstract setting of theory update [KM91, MT93b, Bar94, MT94, MT95a, PT95, PT97, MT98a]. Recall that one example is the formalism of revision programming [MT94, MT98a], which can be understood as a precurser to the causal approaches presented in this dissertation. As previously mentioned, an extension of the proposal of [MT95b, PT95] is used to define possible next states in the action language $\mathcal{AC}$ in the first part of this dissertation.

2.6 Causal Theories of Action

We mention separately the ongoing line of recent work on causal theories of action [Gef90, Elk92, BH93, Bar95, Lin95, MT95b, Thi95a, Gus96, Lin96, San96, Lif97, MT97, Thi97, Tur97, GL98, MT98b, Tur98], most of which has already been cited in other contexts.

Recall that Hector Geffner in [Gef89, Gef90] introduced an ambitious nonmonotonic formalism for causal and default reasoning, and discussed how to apply it to a number of problems in commonsense knowledge representation and nonmonotonic reasoning, including the problem of reasoning about action. Although puzzling in some details, his proposal anticipates several ideas central to the work presented in the second part of this dissertation. For one thing, the syntax of UCL is essentially that of his nonmonotonic formalism.[5] Moreover, the intuitive reading of the modal operator is really very close in the two formalisms: Geffner reads "explained," where we read, essentially, "has a cause." In fact, Geffner was interested in a similar notion of causal explanation, and, except for some minor technical complications, it seems that UCL can be embedded in his formalism. Since Geffner's formalism is considerably more ambitious, one way to characterize the relative contribution of UCL is to say that while its aims are simpler, so are its means.

There is another striking fact. In his example of an action theory, Geffner includes formulas of the form

$$\phi \supset \mathsf{C}\psi \tag{2.1}$$

which is exactly the form of formulas in the subset of UCL that corresponds to the causal theories formalism of McCain and Turner introduced in [MT97]. Such formulas have one of the crucial properties of static causal laws: they are noncontrapositive. That is, (2.1) does not entail the formula $\neg\psi \supset \mathsf{C}\neg\phi$. Such causal laws are of considerable interest for a number of reasons. For one thing, as discussed at length in the third part of this dissertation, they can lead to effective methods for automated reasoning about actions when the formula ψ in the consequent of (2.1) is restricted to be a literal. They also played at least an inspirational role in the circumscriptive proposal of Lin [Lin95], in which *Caused* is introduced as a predicate that is minimized with respect to "causal laws" of roughly the form

$$\mathit{Holds}(\phi, s) \supset \mathit{Caused}(F, V, s)$$

where V is a "truth value" and F is the name of a fluent. Again, intuitively at least, such a sentence corresponds to the special case of (2.1) when ψ is a literal.

[5] More accurately, Geffner stops short of introducing a modal logic, but considers instead atomic symbols of the form $\mathsf{C}\phi$, where ϕ is a formula of propositional logic. He then imposes closure conditions on his propositional models which serve to approximate the semantics of propositional S5 modal logic.

Elkan [Elk92] argues for the importance of causality in describing the indirect effects of actions, but his proposal is not fully specified. Instead he illustrates his ideas by considering examples. It is clear though that his approach involves "precomputing" the indirect effects of actions as a preliminary step to formalizing the action domain. This is counter to the spirit of the work in this dissertation. Elkan also rejects the notion of static causal laws, arguing that cause always temporally precedes effect.

Brewka and Hertzberg [BH93] attempt to represent causal background knowledge using inference rules, much as is done in the definition of possible next states used in this dissertation. Nevertheless, their definition is based on the principle of minimal change. Rather than attempt to reason directly about what is caused, they use inference rules to alter the measure of change. This can lead to unsatisfactory results, as discussed in Chapter 3.4 of this dissertation, and also makes their definition somewhat unwieldy.

As we have said, the action language $\mathcal{AC}$ discussed in Chapter 3.4 incorporates the causal approach introduced in [MT95b]. Further discussion of the recent work on causality in theories of action in [Bar95, Lin95, Thi95a, Thi97] is postponed until Chapter 3.4, where brief comparisons with $\mathcal{AC}$ appear. Lin's proposal [Lin95, Lin96] is also considered again, in much greater detail, in Chapter 5.

Chapter 3

Inference Rules in Causal Action Theories

3.1 Introduction

In the first part of this dissertation, we employ the methodology proposed by Gelfond and Lifschitz (discussed in Chapter 2) which involves first defining a high-level language for representing commonsense knowledge about actions, and then specifying translations from the high-level action language into general-purpose formalisms. Accordingly, we define a high-level action language $\mathcal{AC}$, and specify sound and complete translations of portions of $\mathcal{AC}$ into default logic and logic programming.

Before defining the action language $\mathcal{AC}$, we introduce a definition of possible next states that reflects a causal understanding of the commonsense law of inertia, as previously discussed, and allows us to take into account static causal laws, characterized mathematically by means of inference rules.

Our translations of $\mathcal{AC}$ take advantage of the fact that default logic and logic programming are rule-based formalisms. This of course simplifies the translation of static causal laws, but, as we demonstrate, it also allows convenient representations

of other causal aspects of action domains. In particular, as previously described, we represent the commonsense law of inertia with rules of a very simple form.

This chapter is organized as follows. In Section 3.2 we illustrate the range of applicability of the definitions introduced in this chapter, by considering four example action domains. We provide for each a brief informal description, a formalization in the action language $\mathcal{AC}$, and a corresponding formalization in logic programming, default logic, or both. In Section 3.3 we introduce the causal definition of possible next states that is used in $\mathcal{AC}$, and briefly investigate its mathematical properties. We then define $\mathcal{AC}$ in Section 3.4, and compare it to some other recent proposals for causal theories of action. In Section 3.5 we specify a translation from a subset of $\mathcal{AC}$ into default logic, and state the relevant soundness and completeness theorems. We also specify a second, simpler translation which is sound and complete for a smaller subset of $\mathcal{AC}$. Section 3.5 also includes a comparison between the default theory we obtain for the classic Yale Shooting domain and the default theories considered by Hanks and McDermott and by Morris [Mor88]. In Section 3.6 we show that, under simple syntactic restrictions on the form of $\mathcal{AC}$ domain descriptions, the translations into default logic can be adapted to generate logic programs that correctly represent commonsense knowledge about actions. We defer until Chapter 4 most proofs of theorems.

3.2 Four Examples

In order to illustrate the range of applicability of the definitions introduced in this chapter, we next consider four example action domains, providing for each an informal description, a formalization in the high-level action language $\mathcal{AC}$, and a sound and complete translation into default logic, logic programming, or both.

Example 1

We begin with yet another variant of the Yale Shooting domain. There is a pilgrim and a turkey. The pilgrim has two guns. If the pilgrim fires a loaded gun, the turkey will be caused to be not alive in the resulting situation. Furthermore, one can make the turkey be not trotting by making it not alive, because whenever there is a cause for the turkey being not alive there is also a cause for the turkey not trotting. Initially the turkey is trotting and at least one of the two guns is loaded.

Based on this informal description, we can conclude, for instance, that the turkey is not trotting in the situation that would result if the pilgrim were to shoot his two guns, one after the other, in the initial situation.

This is an example of a "temporal projection" action domain, in which we are told only about the values of fluents in the initial situation. Furthermore, this is an "incomplete" temporal projection domain, since the information we are given about the initial situation does not completely describe it.

This action domain includes a static causal law: whenever not alive is caused, not trotting is also caused. It follows from this static causal law that one can make the turkey be not trotting by making it be not alive. Therefore, shooting a loaded gun when the turkey is trotting has not only the direct effect of killing the turkey, but also the indirect effect, or ramification, of making it stop trotting.

In the action language $\mathcal{AC}$, this action domain can be formalized as follows.[1]

initially $Trotting$

initially $Loaded(Gun_1) \vee Loaded(Gun_2)$

$\neg Alive$ **suffices for** $\neg Trotting$

$Shoot(x)$ **causes** $\neg Alive$ **if** $Loaded(x)$

[1] Although $\mathcal{AC}$ domain descriptions do not include variables, we sometimes use metavariables in our representations of them. For instance, the metavariable x in the fourth expression in the domain description ranges over $\{Gun_1, Gun_2\}$.

This $\mathcal{AC}$ domain description entails, for instance, the $\mathcal{AC}$ proposition

$$\neg Trotting \textbf{ after } Shoot(Gun_1); Shoot(Gun_2)$$

which says that $\neg Trotting$ holds in the situation that would result from performing the action sequence $Shoot(Gun_1); Shoot(Gun_2)$ in the initial situation.

The domain description includes the proposition

$$\neg Alive \textbf{ suffices for } \neg Trotting$$

which describes the static causal law: it says that, in the action domain we are describing, whenever $\neg Alive$ is caused, $\neg Trotting$ is also caused. Because of this static causal law, it is impossible in this action domain for *Trotting* to be true when *Alive* is false. Intuitively, this can be explained as follows. In every situation, (we require that) every fact is caused. In particular then, whenever *Alive* is false in a situation, the fact that *Alive* is false must be caused. And since $\neg Alive$ is caused, it follows by the static causal law that $\neg Trotting$ is also caused; and consequently $\neg Trotting$ must be true as well. Accordingly, the semantics of $\mathcal{AC}$ guarantees that no model of the domain description includes a situation in which both *Trotting* and $\neg Alive$ hold. On the other hand, we emphasize that in the semantics of $\mathcal{AC}$ it does not follow from this proposition that *Alive* can be made true by making *Trotting* true! This failure of contraposition reflects the fact that one cannot make a turkey be alive just by making it trot. (On the contrary, common sense tells us that a turkey simply cannot trot unless it is alive.)

We display in Figure 3.1 a correct formalization in default logic of this example. It can be obtained from the above $\mathcal{AC}$ domain description by a translation defined in Section 3.5 of this dissertation.

The first rule in this default theory reflects the assertion that the turkey is initially trotting, by ensuring that there can be no consistent extension of the default theory in which the turkey is initially not trotting. In a similar fashion, the second

$$\frac{\neg Holds(Trotting, S_0)}{False} \quad \frac{\neg(Holds(Loaded(Gun_1), S_0) \vee Holds(Loaded(Gun_2), S_0))}{False}$$

$$\frac{\neg Holds(Alive, s)}{\neg Holds(Trotting, s)} \quad \frac{Holds(Loaded(x), s)}{\neg Holds(Alive, Result(Shoot(x), s))}$$

$$\frac{: Holds(f, S_0)}{Holds(f, S_0)} \quad \frac{: \neg Holds(f, S_0)}{\neg Holds(f, S_0)}$$

$$\frac{Holds(f, s) : Holds(f, Result(a, s))}{Holds(f, Result(a, s))} \quad \frac{\neg Holds(f, s) : \neg Holds(f, Result(a, s))}{\neg Holds(f, Result(a, s))}$$

Figure 3.1: Default theory for Example 1.

rule says that at least one of the pilgrim's guns is initially loaded. The form of these two rules may be surprising. For instance, one may wonder why the first rule is not

$$\frac{True}{Holds(Trotting, S_0)}$$

instead. This can be explained as follows.

Consider the following $\mathcal{AC}$ domain description, obtained by deleting the first two propositions from the above domain description.

$$\neg Alive \textbf{ suffices for } \neg Trotting$$

$$Shoot(x) \textbf{ causes } \neg Alive \textbf{ if } Loaded(x)$$

This reduced domain description has exactly twelve models, one for each possible initial situation. (Recall that the $\mathcal{AC}$ proposition $\neg Alive$ **suffices for** $\neg Trotting$ rules out any situation in which *Alive* is false and *Trotting* is true.) In the semantics of $\mathcal{AC}$, the role of the proposition

$$\textbf{initially } Trotting$$

is simply to eliminate those models in which trotting is initially false. Similarly, the $\mathcal{AC}$ proposition

$$\textbf{initially } Loaded(Gun_1) \vee Loaded(Gun_2)$$

simply eliminates those models in which both guns are initially unloaded. Thus the full domain description has exactly three models.

Now, the translation into default logic has the property that there is a one-to-one correspondence between $\mathcal{AC}$ models of the domain description and consistent extensions of the corresponding default theory. Thus, the default theory for the reduced domain description has twelve consistent extensions. In general, adding a default rule of the form

$$\frac{\phi}{False}$$

simply eliminates all consistent extensions to which ϕ belongs. Therefore, adding the rule

$$\frac{\neg Holds(Trotting, S_0)}{False}$$

simply eliminates those extensions that include the literal $\neg Holds(Trotting, S_0)$. Similarly, adding the rule

$$\frac{\neg(Holds(Loaded(Gun_1), S_0) \vee Holds(Loaded(Gun_2), S_0))}{False}$$

simply eliminates those extensions in which, roughly speaking, both guns are initially unloaded. Adding both of these rules eliminates nine extensions in all, and leaves us with exactly the three extensions that correspond to the three models of the original domain description. Because of the simple, monotonic behavior of such default rules, the correctness of this aspect of the translation is relatively transparent.

The third rule in the default theory can be understood to say that the turkey can be made to stop trotting by making it not alive. Notice that this default rule does not allow one to derive $Holds(Alive, S_0)$ from $Holds(Trotting, S_0)$, for instance. This reflects the fact that the static causal law is noncontrapositive. Nonetheless, in the context of the default theory as a whole, this default rule guarantees that no consistent extension includes both $Holds(Trotting, S_0)$ and $\neg Holds(Alive, S_0)$.

The fourth rule in the default theory can be understood to say that the

1. $False \leftarrow not\ Holds(Trotting, S_0)$
2. $False \leftarrow not\ Holds(Loaded(Gun_1), S_0), not\ Holds(Loaded(Gun_2), S_0)$
3. $\neg Holds(Trotting, s) \leftarrow \neg Holds(Alive, s)$
4. $\neg Holds(Alive, Result(Shoot(x), s)) \leftarrow Holds(Loaded(x), s)$
5. $Holds(f, S_0) \leftarrow not\ \neg Holds(f, S_0)$
6. $\neg Holds(f, S_0) \leftarrow not\ Holds(f, S_0)$
7. $Holds(f, Result(a, s)) \leftarrow Holds(f, s), not\ \neg Holds(f, Result(a, s))$
8. $\neg Holds(f, Result(a, s)) \leftarrow \neg Holds(f, s), not\ Holds(f, Result(a, s))$

Figure 3.2: Logic program for Example 1.

turkey is not alive after the pilgrim fires a gun, if that gun is loaded.[2] Notice that this default rule does not allow one to derive the literal $\neg Holds(Loaded(Gun_1), S_0)$ from the literal $Holds(Alive, Result(Shoot(Gun_1), S_0))$, for instance. This reflects the commonsense intuition that facts in the future cannot cause facts in the past.[3]

The remaining rules in the default theory are standard elements of the translation we are considering. The fifth and sixth rules reflect the obvious, and crucial, fact that each fluent is either true or false in the initial situation, by forcing each consistent extension of the default theory to include, for each fluent F, either the literal $Holds(F, S_0)$ or the literal $\neg Holds(F, S_0)$. Furthermore, these two rules interact to guarantee that the default theory takes into account every possible initial situation. The seventh and eighth rules express the commonsense law of inertia, as previously discussed. For instance, since we have the literal $Holds(Trotting, S_0)$, one of the inertia rules allows us to derive the literal $Holds(Trotting, Result(Shoot(Gun_1), S_0))$, so long as it is consistent to do so (which, roughly speaking, it will be if and only if the first gun is initially unloaded). Notice that these inertia rules again reflect the commonsense belief that facts in the future do not cause facts in the past.

[2] Here, as earlier, x appears as a metavariable ranging over $\{Gun_1, Gun_2\}$.

[3] This point is discussed further in Section 3.5, when we compare our translation for the Yale Shooting domain to previously published default logic formalizations.

This action domain can also be correctly formalized in logic programming, under the answer set semantics of Gelfond and Lifschitz. Because of the equivalence of logic programming under the answer set semantics and the appropriate subset of default logic, the logic program in Figure 3.2 can be understood as a direct translation of the previous default theory, except for the first and second rules, which are handled in a slightly more complex fashion (to be explained in Section 3.6 of this dissertation).

Recall that one consequence of the action domain in this example is that the turkey would not be trotting if the pilgrim were to shoot his two guns, one after the other, in the initial situation. Accordingly, the literal

$$\neg Holds(Trotting, Result(Shoot(Gun_2), Result(Shoot(Gun_1), S_0)))$$

is a consequence of both the default theory and the logic program.

Example 2

Let us consider a second action domain, adapted from [Lin95], in which there is a spring-loaded briefcase with two clasps. We have actions that unfasten the clasps, one at a time. If both clasps are unfastened, the briefcase pops open. Initially the briefcase is not open. We can conclude in this case that the briefcase would be open after we unfastened the first clasp in the initial situation if and only if the second clasp is initially not fastened.

As in the previous example, this is an incomplete temporal projection domain in which there is a static causal law. Once again, we are interested in ramifications.

This action domain can be described in $\mathcal{AC}$ as follows.

$$\textbf{initially } \neg Open$$

$$Unfasten(x) \textbf{ causes } \neg Fastened(x)$$

$$\neg Fastened(Clasp_1) \wedge \neg Fastened(Clasp_2) \textbf{ suffices for } Open$$

$$\frac{Holds(Open, S_0)}{False} \qquad \frac{True}{\neg Holds(Fastened(x), Result(Unfasten(x), s))}$$

$$\frac{\neg Holds(Fastened(Clasp_1), s) \wedge \neg Holds(Fastened(Clasp_2), s)}{Holds(Open, s)}$$

$$\frac{: Holds(f, S_0)}{Holds(f, S_0)} \qquad \frac{: \neg Holds(f, S_0)}{\neg Holds(f, S_0)}$$

$$\frac{Holds(f, s) : Holds(f, Result(a, s))}{Holds(f, Result(a, s))} \qquad \frac{\neg Holds(f, s) : \neg Holds(f, Result(a, s))}{\neg Holds(f, Result(a, s))}$$

Figure 3.3: Default theory for Example 2.

The corresponding default theory is shown in Figure 3.3. The first three rules of the default theory correspond to the three propositions in the domain description. The last four rules again encode the completeness of the initial situation and the commonsense law of inertia. (As in the previous example, this domain description can also be translated into a logic program.)

The domain description entails the $\mathcal{AC}$ proposition

$$(\ Open \textbf{ after } Unfasten(Clasp_1)\ ) \equiv (\ \textbf{initially } \neg Fastened(Clasp_2)\ )$$

and, accordingly, the formula

$$Holds(Open, Result(Unfasten(Clasp_1), S_0)) \equiv \neg Holds(Fastened(Clasp_2), S_0)$$

is a consequence of the default theory. In contrast with the previous example, we are concerned in this case with a consequence of a more complex kind, relating fluent values at two different time points.

As we have said, background knowledge in action domains has traditionally been represented in the form of state constraints, which are, intuitively speaking, formulas of classical logic that are said to hold in every possible state of the world. Thus, for example, in a more traditional description of this action domain, one might write a state constraint

$$\textbf{always } \neg Fastened(Clasp_1) \wedge \neg Fastened(Clasp_2) \supset Open$$

in place of the proposition

$$\neg Fastened(Clasp_1) \wedge \neg Fastened(Clasp_2) \textbf{ suffices for } Open$$

which represents a static causal law. It would in fact be a mistake to do this. While both propositions correctly rule out states in which the briefcase is closed and yet neither clasp is fastened, the two propositions do not agree on the indirect effects, or ramifications, of actions. For instance, consider a situation in which the first clasp is fastened, the second one isn't, and the briefcase is closed. According to the state constraint, it is possible that, after unfastening the first clasp, the briefcase would remain closed and the second clasp would (mysteriously) become fastened.[4] This outcome—sanctioned by the state constraint—is contrary to expectation, and is in fact not sanctioned by the static causal law.

In general, static causal laws are more expressive than state constraints, as the previous example suggests. In fact, as we show in Section 3.3, state constraints, as they have been traditionally understood, constitute a simple special case of static causal laws. In Section 3.4 we will discuss these issues at greater length in light of this and other examples.

Example 3

A third action domain, loosely adapted from [KL94, GKL95], involves flipping a coin and betting on the outcome.[5] After each toss the coin either lies heads or it doesn't. (Intuitively, the outcome of the toss action is nondeterministic.) If you bet heads when the coin lies heads, you become a winner. If you bet heads when the coin doesn't lie heads, you cease being a winner. Now, suppose that you toss and bet heads, after which you are a winner. In this case we can conclude that the coin

[4] The reader may notice the similarity to the "Two Ducts" domain of Ginsberg and Smith [GS88], as well as to the "Two Switches" domain of [Lif90]. As we'll explain in Section 3.4, the current example is more telling for our purposes.

[5] This action domain also resembles Sandewall's "Russian Turkey Shoot" domain [San94].

1. $False \leftarrow not\ Holds(Winner, Result(BetHeads, Result(Toss, S_0)))$
2. $Holds(Heads, Result(Toss, s)) \leftarrow not\ \neg Holds(Heads, Result(Toss, s))$
3. $\neg Holds(Heads, Result(Toss, s)) \leftarrow not\ Holds(Heads, Result(Toss, s))$
4. $Holds(Winner, Result(BetHeads, s)) \leftarrow Holds(Heads, s)$
5. $\neg Holds(Winner, Result(BetHeads, s)) \leftarrow \neg Holds(Heads, s)$
6. $Holds(f, S_0) \leftarrow not\ \neg Holds(f, S_0)$
7. $\neg Holds(f, S_0) \leftarrow not\ Holds(f, S_0)$
8. $Holds(f, Result(a, s)) \leftarrow Holds(f, s), not\ \neg Holds(f, Result(a, s))$
9. $\neg Holds(f, Result(a, s)) \leftarrow \neg Holds(f, s), not\ Holds(f, Result(a, s))$

Figure 3.4: Logic program for Example 3.

was heads after the toss.

This is an action domain in which there is a nondeterministic action. Notice also that this is not a temporal projection domain, since we are told about the value of a fluent in a non-initial situation. In this case, we are interested in reasoning from a later to an earlier time.

This action domain can be formalized in $\mathcal{AC}$ as follows.

$$Winner \textbf{ after } Toss; BetHeads$$

$$Toss \textbf{ possibly changes } Heads$$

$$BetHeads \textbf{ causes } Winner \textbf{ if } Heads$$

$$BetHeads \textbf{ causes } \neg Winner \textbf{ if } \neg Heads$$

This domain description entails the $\mathcal{AC}$ proposition

$$Heads \textbf{ after } Toss$$

and, accordingly, the literal

$$Holds(Heads, Result(Toss, S_0))$$

is entailed by the corresponding logic program, listed in Figure 3.4.

The first rule of this program corresponds to the first proposition in the domain description. The next two rules correspond to the second proposition in the domain description: the nondeterministic effect of the *Toss* action is captured through the interaction of these rules. The fourth and fifth rules correspond to the third and fourth propositions in the domain description. Again, the last four rules encode the completeness of the initial situation and the commonsense law of inertia.

Example 4

Finally, consider a fourth action domain, adapted from [KL94]. The door to your hotel room is closed. It can be opened by inserting the keycard, but that is not possible when you do not have the keycard.

In $\mathcal{AC}$ we write the following.

$$\textbf{initially } \neg DoorOpen$$

$$InsertCard \textbf{ causes } DoorOpen$$

$$\textbf{impossible } InsertCard \textbf{ if } \neg HasCard$$

Since it is not known whether or not you initially have your keycard, this domain description does not entail the $\mathcal{AC}$ proposition

$$DoorOpen \textbf{ after } InsertCard$$

but it does entail the following weaker $\mathcal{AC}$ proposition.

$$(\ DoorOpen \textbf{ after } InsertCard\ ) \equiv (\ \textbf{initially } HasCard\ )$$

Accordingly, the corresponding logic program (Figure 3.5) does not entail the literal

$$Holds(DoorOpen, Result(InsertCard, S_0))$$

but each answer set for the program includes exactly one of the following two literals:

$$Holds(DoorOpen, Result(InsertCard, S_0))\,,\ \neg Holds(HasCard, S_0)\,.$$

1. $False \leftarrow not\ \neg Holds(DoorOpen, S_0)$
2. $Holds(DoorOpen, Result(InsertCard, s)) \leftarrow Reachable(Result(InsertCard, s))$
3. $\neg Reachable(Result(InsertCard, s)) \leftarrow \neg Holds(HasCard, s)$
4. $Reachable(s) \leftarrow not\ \neg Reachable(s)$
5. $\neg Reachable(Result(a, s)) \leftarrow \neg Reachable(s)$
6. $Holds(f, S_0) \leftarrow not\ \neg Holds(f, S_0)$
7. $\neg Holds(f, S_0) \leftarrow not\ Holds(f, S_0)$
8. $Holds(f, Result(a, s)) \leftarrow Holds(f, s), Reachable(Result(a, s)),$
   $not\ \neg Holds(f, Result(a, s))$
9. $\neg Holds(f, Result(a, s)) \leftarrow \neg Holds(f, s), Reachable(Result(a, s)),$
   $not\ Holds(f, Result(a, s))$

Figure 3.5: Logic program for Example 4.

This domain description, unlike those considered in the previous examples, describes an "action precondition" for one of its actions: the action *InsertCard* can be performed only in situations where *HasCard* holds. Thus, for instance, the domain description fails to entail the proposition

$$True \textbf{ after } InsertCard$$

which says, roughly speaking, that the action of inserting the keycard can be performed in the initial situation.

The action language $\mathcal{AC}$ handles action preconditions in a flexible and robust manner. By contrast we note that the sole restriction placed in this dissertation on the $\mathcal{AC}$ domain descriptions translated into default logic will be a requirement that action preconditions be expressed in a particular explicit form. The domain description above satisfies this requirement. There are additional, syntactic restrictions on the domains that we translate into logic programming. This domain satisfies these additional restrictions as well, and therefore we are able to formalize it in logic programming, as shown in Figure 3.5, using a translation defined in Section 3.6 of this

dissertation.

The first three rules of this program correspond to the three propositions in the domain description. Notice that the translation in this case is complicated by the fact that in this action domain, unlike the domains considered previously, there is an action that is sometimes impossible to perform. This additional difficulty is accommodated in the translation through the use of an additional predicate *Reachable*, which says of a sequence of actions that it can be performed in the initial situation. (Recall the related discussion in Chapter 2.) For instance, the third rule says, roughly speaking, that if you are in a situation in which you do not have your keycard, there is no "reachable" situation that can result from inserting your keycard—since you in fact cannot insert it. Rule 2 says that if it is indeed possible to insert your keycard in the current situation, then the door will be open after you have done so. Rule 4 expresses the assumption that situations are reachable unless we say otherwise. This assumption is based on the more fundamental assumption in $\mathcal{AC}$ that actions are performable unless we say otherwise (either explicitly or implicitly). Rule 5 says that if a given situation is not reachable, then it does not have any reachable successors. Notice that in this translation the assumption of inertia (in the last two rules) is also predicated on reachability.

3.3 A Causal Definition of Possible Next States

In this section we introduce and briefly investigate the causal definition of possible next states that is used in the action language $\mathcal{AC}$. As mentioned previously, (essentially) this definition was first introduced in [MT95b, PT95].

3.3.1 Preliminary Definitions

Given a set U of propositional symbols, we denote by $\mathcal{L}(U)$ the language of propositional logic with exactly the atoms U. We assume here and throughout the dis-

sertation that the language includes a zero-place logical connective *True* such that *True* is a tautology. *False* stands for $\neg$*True*. (Notice that U can be empty.) For any literal L, let $\overline{L}$ denote the literal complementary to L. For any set X of literals, let $\overline{X} = \{\overline{L} : L \in X\}$. By $Lit(U)$ we denote the set consisting of all literals in the language $\mathcal{L}(U)$. In this description we say nothing about the form of the atoms in U, but of course an important special case is when U is the set of all ground atoms of a many-sorted first-order language.

We identify each interpretation of $\mathcal{L}(U)$ with the set of literals from $Lit(U)$ that are true in it. We say that a set of formulas from $\mathcal{L}(U)$ is *logically closed* if it is closed under propositional logic (with respect to the language $\mathcal{L}(U)$). Inference rules over $\mathcal{L}(U)$ will be written as expressions of the form

$$\frac{\phi}{\psi}$$

where ϕ and ψ are formulas from $\mathcal{L}(U)$.

Let R be a set of inference rules, and let Γ be a set of formulas. We say that Γ is *closed under* R if for every rule $\frac{\phi}{\psi}$ in R, if ϕ belongs to Γ then ψ does too. By $Cn_U(R)$ we denote the least logically closed set of formulas from $\mathcal{L}(U)$ that is closed under R. We often find it convenient to identify a formula ϕ with the inference rule

$$\frac{True}{\phi}.$$

Under this convention, $Cn_U(\Gamma)$ denotes the least logically closed set of formulas from $\mathcal{L}(U)$ that contains Γ. Similarly, $Cn_U(\Gamma \cup R)$ is the least logically closed set of formulas from $\mathcal{L}(U)$ that contains Γ and is also closed under R. We usually omit the subscript to Cn when there is no danger of confusion.

Although the definitions in this section are stated for the propositional case, they are taken, in the standard way, to apply in the (quantifier-free) non-propositional case as well, by taking each non-ground expression to stand for all of its ground instances.

3.3.2 Possible Next States: Rule Update

We are now ready to introduce the causal definition of possible next states from [MT95b, PT95], which is applicable in the presence of arbitrary inference rules. Following [PT95], we call this definition "rule update." In the last subsection of this section we will discuss a slight extension of this definition that corresponds precisely to the definition of possible next states used in the action language $\mathcal{AC}$.

Let $\mathcal{R}$ be a set of inference rules over $\mathcal{L}(U)$.[6] Let I and I' be interpretations of $\mathcal{L}(U)$. We say I' is a *rule update of I by $\mathcal{R}$* if

$$Cn_U(I') = Cn_U\left((I \cap I') \cup \mathcal{R}\right) .$$

The literals in $I \cap I'$ can be understood as the facts that are "preserved by inertia" as we move from interpretation I to interpretation I'. In accordance with a causal understanding of the commonsense law of inertia, the definition of rule update does not require any additional "causal explanation" for the truth of these literals in I'. The definition does require though that all new facts in I'—that is, the literals in $I' \setminus I$—be "causally explained" by the rules in $\mathcal{R}$, along with the literals in $I \cap I'$. Accordingly, it follows from the definition of rule update that I' is a rule update of I by $\mathcal{R}$ if and only if the following two conditions are met:

- $Cn(I')$ is closed under $\mathcal{R}$;
- for all literals L in $I' \setminus I$, $L \in Cn\left((I \cap I') \cup \mathcal{R}\right)$.

That is, roughly speaking, in order for I' to be a rule update of I by $\mathcal{R}$, I' must be "consistent with" the rules in $\mathcal{R}$, and furthermore every literal in I' must be causally explained—either it held in I or it was forced to become true according to $\mathcal{R}$.

[6] In applications to reasoning about action, the set $\mathcal{R}$ will normally consist of two parts—a set E of formulas whose truth is "directly" caused by an action, and a set R of inference rules that represent static causal laws. For present purposes, it is convenient to suppress these details.

Consider the following example.

$$I_1 = \{a, b, c\} \qquad \mathcal{R}_1 = \left\{\frac{a}{\neg b \vee \neg c}\right\} \qquad I_2 = \{a, \neg b, c\}$$

First we will show that I_2 is a rule update of I_1 by $\mathcal{R}_1$. Notice that

$$I_1 \cap I_2 = \{a, c\}$$

and that

$$\neg b \in Cn\left((I_1 \cap I_2) \cup \mathcal{R}_1\right) .$$

So for all literals $L \in I_2 \setminus I_1$, $L \in Cn\left((I_1 \cap I_2) \cup \mathcal{R}_1\right)$. And since $Cn(I_2)$ is closed under $\mathcal{R}_1$, we have shown that I_2 is an update of I_1 by $\mathcal{R}_1$. A symmetric argument shows that the interpretation $\{a, b, \neg c\}$ is also a rule update of I_1 by $\mathcal{R}_1$. On the other hand, if we take $I_3 = \{\neg a, b, c\}$, then $I_1 \cap I_3 = \{b, c\}$; and we see that $\neg a \notin Cn\left((I_1 \cap I_3) \cup \mathcal{R}_1\right)$. So I_3 is not a rule update of I_1 by $\mathcal{R}_1$. One can similarly show that $\{a, \neg b, \neg c\}$ is not a rule update of I_1 by $\mathcal{R}_1$.

3.3.3 Rule Update and Minimal Change

Next we briefly investigate mathematical properties of rule update. For instance, we show that rule update does not violate the principle of minimal change, even though it is based on a causal understanding of the commonsense law of inertia. We also show that rule update includes as a special case Winslett's classic minimal-change definition of update by means of formulas. We do not include a proof that rule update also generalizes Marek and Truszczyński's revision programming [MT94, MT98a], as mentioned previously in Chapter 1. This fact is proved in [PT95, PT97].

Given interpretations I, I' and I'', we say that I' is *closer to I than I'' is* if $I'' \cap I$ is a proper subset of $I' \cap I$.

Let Γ be a set of formulas. Let I and I' be interpretations. We say that I' is a *formula update of I by* Γ if I' is a model of Γ such that no model of Γ is closer

to I than I' is.[7]

In order to compare formula update and rule update, we introduce the following additional definition. Given a set $\mathcal{R}$ of inference rules, we define a corresponding set of formulas $Theory(\mathcal{R})$ as follows.

$$Theory(\mathcal{R}) = \left\{ \phi \supset \psi : \frac{\phi}{\psi} \in \mathcal{R} \right\}$$

Thus, for example, $Theory(\mathcal{R}_1) = \{a \supset \neg b \vee \neg c\}$.

Let $\mathcal{R}$ be a set of inference rules and I an interpretation. Notice that $Cn(I)$ is closed under $\mathcal{R}$ if and only if I is a model of $Theory(\mathcal{R})$. Thus, every rule update of I by $\mathcal{R}$ is a model of $Theory(\mathcal{R})$. In fact, we have the following stronger result, which shows that rule update satisfies the principle of minimal change.

Proposition 3.1 *Let $\mathcal{R}$ be a set of inference rules and I an interpretation. Every rule update of I by $\mathcal{R}$ is a formula update of I by $Theory(\mathcal{R})$.*

Proof. Assume that I' is a rule update of I by $\mathcal{R}$. So I' is a model of $Theory(\mathcal{R})$. Let I'' be a model of $Theory(\mathcal{R})$ such that $I' \cap I \subseteq I'' \cap I$. We need to show that $I'' = I'$. Since I' and I'' are both interpretations, it's enough to show that $Cn(I') \subseteq Cn(I'')$.

$$\begin{array}{rcll} Cn(I') & = & Cn\,((I \cap I') \cup \mathcal{R}) & (\ I' \text{ is an update of } I \text{ by } \mathcal{R}\ ) \\ & \subseteq & Cn\,((I \cap I'') \cup \mathcal{R}) & (\ I' \cap I \subseteq I'' \cap I\ ) \\ & \subseteq & Cn\,(I'' \cup \mathcal{R}) & (\ I'' \cap I \subseteq I''\ ) \\ & = & Cn(I'') & (\ Cn(I'') \text{ is closed under } \mathcal{R}\ ) \end{array}$$

□

The converse of Proposition 3.1 doesn't hold in general. For instance, we have seen in the example in the previous section that I_3 is not a rule update of I_1 by $\mathcal{R}_1$, and yet it is easy to verify that I_3 is a formula update of I_1 by $Theory(\mathcal{R}_1)$.

[7] The definition given here is equivalent, and almost identical, to the definition in [Win88].

On the other hand, the following proposition shows that if every inference rule in $\mathcal{R}$ has the form $\frac{True}{\phi}$ then the rule updates of I by $\mathcal{R}$ will be exactly the formula updates of I by $Theory(\mathcal{R})$. Thus, rule update includes formula update as a special case.

Proposition 3.2 *Let $\mathcal{R}$ be a set of inference rules, each of the form $\frac{True}{\phi}$. Any formula update of an interpretation I by $Theory(\mathcal{R})$ is a rule update of I by $\mathcal{R}$.*

Proof. Assume I' is a formula update of I by $Theory(\mathcal{R})$. Let I'' be a model of $(I \cap I') \cup Theory(\mathcal{R})$. So I'' is a model of $Theory(\mathcal{R})$. Also $I' \cap I \subseteq I''$, so $I' \cap I \subseteq I'' \cap I$. Since no model of $Theory(\mathcal{R})$ is closer to I than I' is, we can conclude that $I'' = I'$. Thus, I' is the only model of $(I \cap I') \cup Theory(\mathcal{R})$. It follows that $Cn(I') = Cn((I \cap I') \cup Theory(\mathcal{R}))$. Due to the special form of the rules in $\mathcal{R}$, $Cn((I \cap I') \cup Theory(\mathcal{R})) = Cn((I \cap I') \cup \mathcal{R})$. So I' is a rule update of I by $\mathcal{R}$. □

We will find that inference rules of the form $\frac{\phi}{False}$ constitute another simple case of special interest. Adding such a rule simply eliminates all rule updates that satisfy ϕ.

Proposition 3.3 *Let $\mathcal{R}$ be a set of inference rules, and I an interpretation. An intepretation I' is a rule update of I by $\mathcal{R} \cup \left\{ \frac{\phi}{False} \right\}$ if and only if I' is a rule update of I by $\mathcal{R}$ and $I \not\models \phi$.*

The proof is straightforward.

3.3.4 Explicit Definitions in Rule Update

The definition of possible next states in $\mathcal{AC}$ is actually a slight extension of rule update, in which "explicit definitions" are accomodated. This will require a little explanation.

In classical propositional logic, given a theory Γ in a language $\mathcal{L}(U \setminus \{p\})$, we can obtain a definitional extension Γ' of Γ, in the language $\mathcal{L}(U)$, by adding to Γ an explicit definition of the form $p \equiv \phi$, where ϕ is a formula of $\mathcal{L}(U \setminus \{p\})$. There is a one-to-one correspondence between models of Γ and models of Γ', which can be characterized as follows. For any interpretation I of $\mathcal{L}(U \setminus \{p\})$, let $p(I)$ denote the interpretation of $\mathcal{L}(U)$ such that $I \subset p(I)$ and $p(I) \models p \equiv \phi$. Every model of Γ' can be written in the form $p(I)$ for some interpretation I of $\mathcal{L}(U \setminus \{p\})$. Moreover, for all interpretations I of $\mathcal{L}(U \setminus \{p\})$, $I \models \Gamma$ if and only if $p(I) \models \Gamma'$. Notice that it follows that Γ' is a conservative extension of Γ. Finally, it is clear that we can then replace with p any occurrence of ϕ in any formula of Γ', except in the explicit definition of p, and obtain an equivalent theory.

We wish to obtain a similar "definitional extension" result for rule update. Here explicit definitions will be inference rules of the form $\frac{True}{p \equiv \phi}$. (Recall that we often identify such a rule with the formula $p \equiv \phi$.)

Let $\mathcal{R}$ be a set of inference rules in a language $\mathcal{L}(U \setminus \{p\})$. Let $\mathcal{R}'$ be the set of inference rules over $\mathcal{L}(U)$ obtained by adding to $\mathcal{R}$ the explicit definition $\frac{True}{p \equiv \phi}$, where ϕ is a formula of $\mathcal{L}(U \setminus \{p\})$. In this case what we can say is that there is a one-to-one correspondence between models of $Cn_{U \setminus \{p\}}(\mathcal{R})$ and models of $Cn_U(\mathcal{R}')$. The characterization is the same as before. Every model of $Cn_U(\mathcal{R}')$ can be written in the form $p(I)$ for some interpretation I of $\mathcal{L}(U \setminus \{p\})$. Moreover, for all interpretations I of $\mathcal{L}(U \setminus p)$, $I \models Cn_{U \setminus \{p\}}(\mathcal{R})$ if and only if $p(I) \models Cn_U(\mathcal{R}')$. Finally, it is clear that if $\mathcal{R}''$ can be obtained from $\mathcal{R}'$ by replacing with p any or all occurrences of ϕ in any or all rules in $\mathcal{R}'$, except in the explicit definition of p, then $Cn_U(\mathcal{R}') = Cn_U(R'')$.

This is almost the result we want, except that it does not refer to rule update. So, is there a one-to-one correspondence between updates by $\mathcal{R}$ and updates by $\mathcal{R}'$? More precisely, is it the case that, for any interpretations I and I' of $\mathcal{L}(U \setminus p)$, I' is an update of I by $\mathcal{R}$ if and only if $p(I')$ is an update of $p(I)$ by $\mathcal{R}'$? The answer is

no. To see why this may be so, observe that our previous observations imply that

$$Cn_{U \setminus \{p\}}(I') = Cn_{U \setminus \{p\}}\left((I \cap I') \cup \mathcal{R}\right) \quad \text{iff} \quad Cn_U(p(I')) = Cn_U((I \cap I') \cup \mathcal{R}').$$

Therefore, I' is an update of I by $\mathcal{R}$ if and only if

$$Cn_U(p(I')) = Cn_U\left((I \cap I') \cup \mathcal{R}'\right).$$

But $p(I')$ is an update of $p(I)$ by $\mathcal{R}'$ if and only if

$$Cn_U(p(I')) = Cn_U\left((p(I) \cap p(I')) \cup \mathcal{R}'\right).$$

We see that update by $\mathcal{R}$ and update by $\mathcal{R}'$ may diverge because, in general, $I \cap I'$ can be a proper subset of $p(I) \cap p(I')$.

Here's such an example. Take $U = \{p, q, r\}$, $\mathcal{R} = \left\{\frac{True}{\neg q}\right\}$, and consider

$$\mathcal{R}' = \mathcal{R} \cup \left\{\frac{True}{p \equiv (q \vee r)}\right\}.$$

The interpretation $I' = \{\neg q, \neg r\}$ is the only rule update of $I = \{q, \neg r\}$ by $\mathcal{R}$. As expected, $p(I') = \{\neg p, \neg q, \neg r\}$ is a rule update of $p(I) = \{p, q, \neg r\}$ by $\mathcal{R}'$. But there is a second, unintended rule update of $p(I)$ by $\mathcal{R}'$. The interpretation $I'' = \{\neg q, r\}$ is not a rule update of I by $\mathcal{R}$, but $p(I'') = \{p, \neg q, r\}$ is a rule update of $p(I)$ by $\mathcal{R}'$. This second rule update of $p(I)$ by $\mathcal{R}'$ makes no sense if we are to understand the inference rule $\frac{True}{p \equiv (q \vee r)}$ as a definition of p. Intuitively, the problem here is that in computing rule updates by $\mathcal{R}'$, we inappropriately take the fluent p to be inertial. Thus $I \cap I'' = \emptyset$ while $p(I) \cap p(I'') = \{p\}$. Since we mean for p to be a defined fluent, it should not be inertial in itself—instead its inertial characteristics should be obtained indirectly from the fluents in terms of which it is defined.

The following proposition justifies the definition of possible next states in the presence of explicit definitions that is used in the following section in defining the semantics of $\mathcal{AC}$.

Proposition 3.4 *Let $\mathcal{R}'$ be a set of inference rules over $\mathcal{L}(U)$ that includes a rule $\frac{True}{p \equiv \phi}$, where ϕ is a formula of $\mathcal{L}(U \setminus \{p\})$. Let $\mathcal{R}$ be the set of inference rules over $\mathcal{L}(U \setminus \{p\})$ obtained from $\mathcal{R}' \setminus \left\{ \frac{True}{p \equiv \phi} \right\}$ by replacing all occurrences of p with ϕ. Every interpretation of $\mathcal{L}(U \setminus \{p\})$ can be written in the form $I \cap Lit(U \setminus \{p\})$, where I is an interpretation of $\mathcal{L}(U)$ that satisfies $p \equiv \phi$. Moreover, for all interpretations I and I' of $\mathcal{L}(U)$ that satisfy $p \equiv \phi$, $I' \cap Lit(U \setminus \{p\})$ is a rule update of $I \cap Lit(U \setminus \{p\})$ by $\mathcal{R}$ if and only if*

$$Cn_U(I') = Cn_U\left[\, (I \cap I' \cap Lit(U \setminus \{p\})) \cup \mathcal{R}' \,\right] .$$

The proof of this proposition is straightforward based on the observations we've already made.

The formulation used here brings us close to the precise statement of the fixpoint condition in the definition of possible next states in $\mathcal{AC}$.

3.4 The Action Language $\mathcal{AC}$

In the high-level action language $\mathcal{AC}$, a description of an action domain is a set of propositions of the following five kinds:

1. value propositions, which restrict the values of fluents in situations that would result from the performance of sequences of actions;
2. sufficiency propositions, which say that whenever one fluent formula is caused to be true, a second fluent formula is also caused to be true;
3. effect propositions, which say that under certain conditions a fluent formula would be caused to hold as a result of the performance of an action;
4. influence propositions, which say that under certain conditions the performance of an action would "nondeterministically" change the value of a fluent;

5. executability propositions, which say that under certain conditions an action would be impossible to perform.

In this section, we specify the syntax and semantics of $\mathcal{AC}$, and illustrate the definitions with an example. We then discuss properties of $\mathcal{AC}$ and consider some related work.

3.4.1 Syntax of $\mathcal{AC}$

We begin with two disjoint nonempty sets of symbols, a set $\mathbf{F}$ of *fluent names* and a set $\mathbf{A}$ of *action names*. We designate a subset $\mathbf{F}_f$ of $\mathbf{F}$ as the *frame fluents* and we call the members of $\mathbf{F} \setminus \mathbf{F}_f$ the *nonframe fluents*. A *fluent formula* is a formula from $\mathcal{L}(\mathbf{F})$. A *frame fluent formula* is a formula from $\mathcal{L}(\mathbf{F}_f)$.

An *atomic value proposition* is an expression of the form

$$\phi \textbf{ after } \overline{A}$$

where ϕ is a fluent formula, and $\overline{A}$ is a string of action names. Such an expression says that the actions $\overline{A}$ can be performed in sequence, beginning in the initial situation, and if they were, the fluent formula ϕ would hold in the resulting situation. If $\overline{A}$ is the empty string, we may write instead

$$\textbf{initially } \phi .$$

A *value proposition* is a propositional combination of atomic value propositions.

A *sufficiency proposition* is an expression of the form

$$\phi \textbf{ suffices for } \psi$$

where ϕ and ψ are fluent formulas. Sufficiency propositions represent static causal laws. Thus, such a proposition says that, in the action domain being described, whenever ϕ is caused, ψ is caused. We write

$$\textbf{always } \phi$$

as an abbreviation for the proposition *True* **suffices for** ϕ and we write

never ϕ

as an abbreviation for the proposition ϕ **suffices for** *False*. Given a nonframe fluent F, an expression of the form

always $F \equiv \phi$

where ϕ is a frame fluent formula, is called an *explicit definition of* F. We require that $\mathcal{AC}$ domain descriptions include an explicit definition of every nonframe fluent.

An *effect proposition* is an expression of the form

A **causes** ϕ **if** ψ

where A is an action name, and ϕ and ψ are fluent formulas. Such an expression says that, if the action A were to be performed in a situation in which ψ holds, the fluent formula ϕ would be caused to hold in the resulting situation. If ψ is the formula *True*, we may simply write A **causes** ϕ.

An *influence proposition* is an expression of the form

A **possibly changes** F **if** ψ

where A is an action name, F is a fluent name, and ψ is a fluent formula. Such an expression says that, if the action A were to be performed in a situation in which ψ holds, the fluent F would be caused to be true or caused to be false in the resulting situation. If ψ is the formula *True*, we may simply write A **possibly changes** F.

An *executability proposition* is an expression of the form

impossible A **if** ψ

where A is an action name and ψ is a fluent formula. Such an expression says that the action A cannot be performed in any situation in which ψ holds. One easily

checks that, in the semantics of $\mathcal{AC}$, such a proposition has essentially the same meaning as the effect proposition A **causes** *False* **if** ψ, but the syntactic distinction becomes convenient in Sections 3.5 and 3.6 when we specify translations of $\mathcal{AC}$ domain descriptions into default logic and logic programming.

An $\mathcal{AC}$ *domain description* is a set of $\mathcal{AC}$ propositions that includes an explicit definition for each nonframe fluent.

3.4.2 Semantics of $\mathcal{AC}$

Let D be an $\mathcal{AC}$ domain description, with fluents $\mathbf{F}$ and frame fluents $\mathbf{F}_f$. A *structure for* D is a partial function from action strings to interpretations of $\mathcal{L}(\mathbf{F})$, whose domain is nonempty and prefix-closed.[8] By $Dom(\Psi)$ we denote the domain of a structure Ψ. Notice that for every structure Ψ, $Dom(\Psi)$ includes the empty string (denoted by ϵ).

Let R be the set of inference rules $\frac{\phi}{\psi}$ such that the sufficiency proposition

ϕ **suffices for** ψ

is in D. An interpretation S of $\mathcal{L}(\mathbf{F})$ is called a *state* if $Cn_{\mathbf{F}}(S)$ is closed under R.

Let A be action name and S a state. We say that A is *prohibited in* S if there is an executability proposition

impossible A **if** ψ

in D such that S satisfies ψ. Let $E(A, S)$ be the set of all fluent formulas ϕ for which there is an effect proposition

A **causes** ϕ **if** ψ

in D such that S satisfies ψ. Similarly, let $F(A, S)$ be the set of all fluent names F for which there is an influence proposition

A **possibly changes** F **if** ψ

[8] A set Σ of strings is *prefix-closed* if, for every string $\sigma \in \Sigma$, every prefix of σ is also in Σ.

in D such that S satisfies ψ.

A set E of fluent formulas is called an *explicit effect of A in S* if:

1. A is not prohibited in S, and
2. there is an interpretation I of $\mathcal{L}(F(A,S))$ such that $E = I \cup E(A,S)$.

We define possible next states for domain description D as follows, using the fixpoint condition described in Section 3.3.4. We say that a state S' *may result from doing A in S* if there is an explicit effect E of A in S such that

$$Cn_{\mathbf{F}}(S') = Cn_{\mathbf{F}}\left[\,(S \cap S' \cap Lit(\mathbf{F}_f)) \cup E \cup R\,\right] .$$

As discussed in the previous section, this definition guarantees that S' may result from doing A in S if and only if the value of every frame fluent in S' is suitably explained—either it held the same value in S and was not made to change, or its value was changed (directly or indirectly) by the action. Let $Res(A,S)$ denote the set of states that may result from doing A in S.

Given a structure Ψ, we say that an atomic value proposition ϕ **after** $\overline{A}$ is *true in* Ψ if $\overline{A} \in Dom(\Psi)$ and $\Psi(\overline{A})$ satisfies ϕ. The general truth definition for value propositions is then given by the standard recursion over the logical connectives.

A structure Ψ for D is a *model of D* if it satisfies the following four conditions.

1. $\Psi(\epsilon)$ is a state.
2. For all $\overline{A} \in Dom(\Psi)$ and all action names A, if $Res(A, \Psi(\overline{A}))$ is nonempty then $\overline{A}; A \in Dom(\Psi)$.
3. For all $\overline{A}; A \in Dom(\Psi)$, $\Psi(\overline{A}; A) \in Res(A, \Psi(\overline{A}))$.
4. Every value proposition in D is true in Ψ.

A value proposition is *entailed* by D if it is true in every model of D.

Let us briefly, and somewhat informally, describe two easily verified properties of such models. First, all "reachable" situations are mapped to states. That is, for all $\overline{A} \in Dom(\Psi)$, $\Psi(\overline{A})$ is a state. Second, if an action string $\overline{A}$ corresponds to a reachable situation, then according to our definition of possible next states it is possible to achieve the state $\Psi(\overline{A})$ by performing the actions $\overline{A}$ in sequence starting in the initial state $\Psi(\epsilon)$.

3.4.3 An Example $\mathcal{AC}$ Domain Description

As an example illustrating the use of the preceding definitions, consider the following $\mathcal{AC}$ domain description D_1—another variant of the Yale Shooting domain. In this domain description, *Dead* is the only nonframe fluent.

always $Dead \equiv \neg Alive$

initially $Walking$

$\neg Walking$ **after** $Shoot$

$\neg Alive$ **suffices for** $\neg Walking$

$Shoot$ **causes** $Dead \wedge \neg Loaded$

impossible $Shoot$ **if** $\neg Loaded$

Notice that we are describing here a different shoot action than in Example 1 (Section 3.2), where shooting was always possible. There, the direct effect $\neg Alive$ of the shoot action had a "fluent precondition" *Loaded*. Here, *Loaded* becomes instead an action precondition of *Shoot*.

Domain description D_1 has a unique model Ψ_1, as follows.

$$\begin{aligned} Dom(\Psi_1) &= \{\epsilon, Shoot\} \\ \Psi_1(\epsilon) &= \{Loaded, Alive, \neg Dead, Walking\} \\ \Psi_1(Shoot) &= \{\neg Loaded, \neg Alive, Dead, \neg Walking\} \end{aligned}$$

It is easy to check, for instance, that the following value proposition is true in Ψ_1.

$$(\textbf{initially } \mathit{Loaded}) \wedge (\mathit{Dead} \wedge \neg \mathit{Loaded} \textbf{ after } \mathit{Shoot})$$

To exercise the definitions, we will verify that Ψ_1 is the unique model of D_1. It is clear that Ψ_1 is a structure for D_1, so we begin by showing that Ψ_1 is a model.

First, we must check that $\Psi_1(\epsilon)$ is a state. We see that domain description D_1 includes the sufficiency propositions

$$\textbf{always } \mathit{Dead} \equiv \neg \mathit{Alive}$$

and

$$\neg \mathit{Alive} \textbf{ suffices for } \neg \mathit{Walking}$$

from which we obtain the associated set of inference rules

$$R = \left\{ \frac{\mathit{True}}{\mathit{Dead} \equiv \neg \mathit{Alive}}, \frac{\neg \mathit{Alive}}{\neg \mathit{Walking}} \right\}.$$

It follows that there are exactly six states in this action domain: namely, the six interpretations of $\mathcal{L}(\mathbf{F})$ that satisfy the fluent formulas

$$\mathit{Dead} \equiv \neg \mathit{Alive}$$

and

$$\neg \mathit{Alive} \supset \neg \mathit{Walking}.$$

We see that $\Psi_1(\epsilon)$ is indeed one of these six states.

Second, we must check that $\mathit{Res}(\mathit{Shoot}, \Psi_1(\mathit{Shoot}))$ is empty. Since D_1 includes the executability proposition

$$\textbf{impossible } \mathit{Shoot} \textbf{ if } \neg \mathit{Loaded}$$

we see that *Shoot* is prohibited in $\Psi_1(\mathit{Shoot})$. Therefore there can be no explicit effect E of *Shoot* in $\Psi_1(\mathit{Shoot})$, which shows that $\mathit{Res}(\mathit{Shoot}, \Psi_1(\mathit{Shoot})) = \emptyset$.

Third, we must verify that $\Psi_1(\mathit{Shoot})$ belongs to $\mathit{Res}(\mathit{Shoot}, \Psi_1(\epsilon))$. That is, we must show that $\Psi_1(\mathit{Shoot})$ may result from doing *Shoot* in $\Psi_1(\epsilon)$. This requires that we check that

$$\mathit{Cn}(\Psi_1(\mathit{Shoot})) = \mathit{Cn}((\Psi_1(\epsilon) \cap \Psi_1(\mathit{Shoot}) \cap \mathcal{L}(\mathbf{F}_f)) \cup E \cup R)$$

where E is an explicit effect of *Shoot* in $\Psi_1(\epsilon)$. We first observe that *Shoot* is not prohibited in $\Psi_1(\epsilon)$. Since D_1 includes no influence propositions, we have $F(\mathit{Shoot}, \Psi_1(\epsilon)) = \emptyset$. Thus the only interpretation of $\mathcal{L}(F(\mathit{Shoot}, \Psi_1(\epsilon)))$ is also $\emptyset$. Since D_1 includes the effect proposition

$$\mathit{Shoot} \textbf{ causes } \mathit{Dead} \wedge \neg \mathit{Loaded}$$

we have

$$E(\mathit{Shoot}, \Psi_1(\epsilon)) = \{\mathit{Dead} \wedge \neg \mathit{Loaded}\}.$$

Given these observations, we can conclude that the unique explicit effect E of *Shoot* in $\Psi_1(\epsilon)$ is $\{\mathit{Dead} \wedge \neg \mathit{Loaded}\}$. It remains to observe that $\Psi_1(\epsilon) \cap \Psi_1(\mathit{Shoot})$ is empty, so $\Psi_1(\epsilon) \cap \Psi_1(\mathit{Shoot}) \cap \mathcal{L}(\mathbf{F}_f)$ is also. Thus what we are to verify is that

$$\mathit{Cn}(\Psi_1(\mathit{Shoot})) = \mathit{Cn}(\{\mathit{Dead} \wedge \neg \mathit{Loaded}\} \cup R)$$

which is clearly true. In fact, what we have shown is that

$$\mathit{Res}(\mathit{Shoot}, \Psi_1(\epsilon)) = \{\Psi_1(\mathit{Shoot})\}$$

since $\Psi_1(\mathit{Shoot})$ is the only state that satisfies $\mathit{Dead} \wedge \neg \mathit{Loaded}$.

Fourth, we must check that Ψ_1 satisfies the two value propositions in D_1, which it clearly does.

So we've shown that Ψ_1 is indeed a model of domain description D_1. Now let us verify that it is the only model.

Assume that Ψ is a model of D_1. By model condition 1 we know that $\Psi(\epsilon)$ is a state, and by model condition 4, we know that the value proposition

$$\textbf{initially } \mathit{Walking}$$

is true in Ψ. That is, $\Psi(\epsilon)$ must satisfy the fluent formula *Walking*. It follows that $\Psi(\epsilon)$ also satisfies *Alive* and $\neg$*Dead*. Thus at this point we know everything about $\Psi(\epsilon)$ except whether or not it satisfies *Loaded*, so there are two states to consider.

Consider the state $S = \{\neg Loaded, Alive, \neg Dead, Walking\}$. We will show that $\Psi(\epsilon)$ cannot be S, which will be sufficient to show that $\Psi(\epsilon) = \Psi_1(\epsilon)$. Since D_1 includes the executability proposition **impossible** *Shoot* **if** $\neg$*Loaded* we know that *Shoot* is prohibited in S. It follows that there can be no explicit effect E of *Shoot* in S, which allows us to conclude that $Res(Shoot, S)$ is empty. Now, by model condition 4 we know that D_1 must satisfy the value proposition

$$\neg Walking \textbf{ after } Shoot$$

so we can conclude that $Shoot \in Dom(\Psi)$. It follows by model condition 3 that $\Psi(Shoot) \in Res(Shoot, \Psi(\epsilon))$. Since $Res(Shoot, S) = \emptyset$, we have $\Psi(\epsilon) \neq S$. So $\Psi(\epsilon) = \Psi_1(\epsilon)$. And since we've already seen that $Res(Shoot, \Psi_1(\epsilon)) = \{\Psi_1(Shoot)\}$, we can conclude by model conditions 2 and 3 that $\Psi(Shoot) = \Psi_1(Shoot)$, which is sufficient to establish the fact that $\Psi = \Psi_1$. So Ψ_1 is the unique model of D_1.

3.4.4 Remarks on the Action Language $\mathcal{AC}$

As we have said, the action language $\mathcal{AC}$ closely resembles the language $\mathcal{AR}$ of Giunchiglia, Kartha and Lifschitz [GKL95, GKL97] and its predecessor $\mathcal{AR}_0$ [KL94]. Unlike the language $\mathcal{AC}$, $\mathcal{AR}$ allows non-boolean fluents; but if we consider only the propositional portion of $\mathcal{AR}$, we find that the model structures for the languages are essentially identical.

Syntactically, the languages $\mathcal{AC}$ and $\mathcal{AR}$ are also very similar. One difference is that $\mathcal{AR}$ does not include sufficiency propositions for representing background knowledge, which is instead represented by state constraints of the form **always** ϕ, where ϕ is a fluent formula. In $\mathcal{AC}$ we understand such an expression as an abbreviation of the corresponding sufficiency proposition *True* **suffices for** ϕ. Thus

$\mathcal{AR}$ state constraints are well-formed $\mathcal{AC}$ propositions. Another significant syntactic difference between $\mathcal{AC}$ and $\mathcal{AR}$ is that $\mathcal{AR}$ includes only atomic value propositions, whereas $\mathcal{AC}$ allows propositional combinations of atomic value propositions. A third difference is that in $\mathcal{AR}$ the expression **impossible** A **if** ψ is simply an abbreviation for the effect proposition A **causes** *False* **if** ψ whereas in $\mathcal{AC}$ these are distinct propositions.[9]

As the preceding observations suggest, the set of well-formed propositional $\mathcal{AR}$ expressions is a proper subset of the set of well-formed $\mathcal{AC}$ expressions. Given this, the relationship between high-level action languages $\mathcal{AR}$ and $\mathcal{AC}$ is captured in the following theorem.[10]

Theorem 3.5 ($\mathcal{AR}$ Theorem) *Let D be a propositional $\mathcal{AR}$ domain description such that every nonframe fluent in D has an explicit definition in terms of frame fluents. D is an $\mathcal{AC}$ domain description, and the $\mathcal{AC}$ models of D are exactly the $\mathcal{AR}$ models of D.*

The statement of the $\mathcal{AR}$ Theorem reflects the fact that some propositional $\mathcal{AR}$ domain descriptions are not $\mathcal{AC}$ domain descriptions. These are the propositional $\mathcal{AR}$ domain descriptions in which there is a nonframe fluent that is not explicitly defined in terms of frame fluents. On the other hand, we have observed that some $\mathcal{AC}$ domain descriptions are not $\mathcal{AR}$ domain descriptions. For example, consider the following $\mathcal{AC}$ formalization of the Two-Switches domain, adapted from

[9] As noted earlier, we will see that this distinction becomes convenient when we specify the translations from $\mathcal{AC}$ into default logic and logic programming in Sections 3.5 and 3.6. Otherwise the distinction is unnecessary.

[10] We omit the proof of this theorem, which would be long and mostly unilluminating, involving the full definition of both $\mathcal{AR}$ and $\mathcal{AC}$. The idea of the main lemma though is interesting: it shows that, under the restrictions in the statement of the theorem, the two high-level languages have equivalent definitions of possible next states. We have already seen two closely related results: Proposition 3.2, which shows that rule update subsumes Winslett's classic minimal-change definition; and Proposition 3.4, which shows that our causal definition of possible next states is a suitable extension of rule update in the presence of explicit definitions.

[KL94] (and originally introduced in [Lif90]).

$$Up(Switch_1) \equiv Up(Switch_2) \textbf{ suffices for } On$$

$$Up(Switch_1) \not\equiv Up(Switch_2) \textbf{ suffices for } \neg On$$

$$Toggle(x) \textbf{ causes } Up(x) \textbf{ if } \neg Up(x)$$

$$Toggle(x) \textbf{ causes } \neg Up(x) \textbf{ if } Up(x)$$

The Two-Switches domain can be formalized in $\mathcal{AR}$ by declaring the fluent On to be nonframe and replacing the two sufficiency propositions by a single state constraint

$$\textbf{always } On \equiv (\, Up(Switch_1) \equiv Up(Switch_2)\,).$$

In modifying the domain description in this manner, we seem to be replacing causal information—the fact that the state of the switches causally determines the state of the light—with a "non-causal" explicit definition. But in doing so, we do not change the set of models.[11]

Let us consider a slight elaboration of the $\mathcal{AC}$ domain description from Example 2 (Section 3.2), adapted from [Lin95], which demonstrates that it is not always possible to obtain intuitively correct results using state constraints augmented by the frame/nonframe distinction. Recall that in this action domain, there is a spring-loaded briefcase with two clasps. We have actions that unfasten the clasps, one at a time. If both clasps are unfastened, the briefcase pops open. We will assume that initially the briefcase is not open and the second clasp is not fastened.

$$\textbf{initially } \neg Open \wedge \neg Fastened(Clasp_2)$$

$$Unfasten(x) \textbf{ causes } \neg Fastened(x)$$

$$\neg Fastened(Clasp_1) \wedge \neg Fastened(Clasp_2) \textbf{ suffices for } Open$$

[11] Notice that in this case, the domain description we obtain is in fact a "legal" $\mathcal{AC}$ domain description, since the nonframe fluent On is explicitly defined in terms of the frame fluent formula $Up(Switch_1) \equiv Up(Switch_2)$.

This domain description entails the value proposition

$$Open \textbf{ after } Unfasten(Clasp_1).$$

As discussed in Section 3.2, one might think of writing the state constraint

$$\textbf{always } (\neg Fastened(Clasp_1) \wedge \neg Fastened(Clasp_2)) \supset Open$$

in place of

$$\neg Fastened(Clasp_1) \wedge \neg Fastened(Clasp_2) \textbf{ suffices for } Open\,.$$

But it seems that there is no way of designating frame and nonframe fluents that will allow the resulting $\mathcal{AR}$ domain description to capture the intended models of the domain. For instance, if we declare $Open$ nonframe, then the briefcase can open spontaneously, as it were, at any time. On the other hand, if we leave all fluents "in the frame," we find that unfastening the first clasp can sometimes have the unintended ramification of fastening the second clasp.

Lin and Reiter [LR94] have suggested the name "ramification constraints" for state constraints that are used to derive indirect effects. One thing the $\mathcal{AR}$ Theorem shows is that $\mathcal{AC}$ expressions of the form

$$\textbf{always } \phi$$

correspond precisely to state constraints in $\mathcal{AR}$, assuming that all nonframe fluents are explicitly defined in terms of frame fluents. Recall that in $\mathcal{AC}$ such an expression stands for the sufficiency proposition

$$True \textbf{ suffices for } \phi\,.$$

It is natural to call such $\mathcal{AC}$ propositions *ramification constraints*.

Lin and Reiter [LR94] describe another use of state constraints: as so-called "qualification constraints." Qualification constraints are state constraints that simply restrict the state space; they do not themselves lead to any indirect effects.

Qualification constraints are so-named because they can lead to "derived action preconditions," or "qualifications."[12] It is straightforward to verify that $\mathcal{AC}$ sufficiency propositions of the form

$$\phi \textbf{ suffices for } \mathit{False}$$

in fact function as qualification constraints, since such propositions simply rule out any state in which ϕ holds, without leading to any indirect effects.[13] Recall that we abbreviate such sufficiency propositions as

$$\textbf{never } \phi\,.$$

It is natural to call such $\mathcal{AC}$ propositions *qualification constraints*.

As an example of an $\mathcal{AC}$ domain description involving a qualification constraint, consider the following formalization of the Emperor Domain of Lin and Reiter [LR94], in which, so the story goes, at most one block at a time can be yellow, by decree of the emperor.

$$\textbf{never } \mathit{Yellow}(\mathit{Block}_1) \wedge \mathit{Yellow}(\mathit{Block}_2)$$
$$\mathit{Paint}(x) \textbf{ causes } \mathit{Yellow}(x)$$

This domain description does not entail the $\mathcal{AC}$ value proposition

$$\mathit{Yellow}(\mathit{Block}_2) \textbf{ after } \mathit{Paint}(\mathit{Block}_2)$$

but it does entail the following weaker proposition.

$$(\,\mathit{Yellow}(\mathit{Block}_2) \textbf{ after } \mathit{Paint}(\mathit{Block}_2)\,) \equiv (\,\textbf{initially } \neg\mathit{Yellow}(\mathit{Block}_1)\,)$$

This reflects the fact that it is possible to paint the second block yellow if and only if the first block is not already yellow. Observe that in this case, in order to obtain

[12] This idea was anticipated by Ginsberg and Smith [GS88].
[13] This is essentially what Proposition 3.3 showed.

an equivalent $\mathcal{AR}$ domain description we replace the sufficiency proposition with the state constraint

$$\textbf{always } \neg(\,\mathit{Yellow}(\mathit{Block}_1) \wedge \mathit{Yellow}(\mathit{Block}_2)\,)$$

and also explictly describe the action preconditions, as follows.

$$\textbf{impossible } \mathit{Paint}(\mathit{Block}_1) \textbf{ if } \mathit{Yellow}(\mathit{Block}_2)$$
$$\textbf{impossible } \mathit{Paint}(\mathit{Block}_2) \textbf{ if } \mathit{Yellow}(\mathit{Block}_1)$$

Up to now we have not presented an example in which it is natural to use a ramification constraint (except to introduce an explicit definition). So consider a blocks world in which there are two blocks (A,B) and four locations (1,2,3,4). Each block is always in exactly one location. There are never two blocks in the same location. For each block, there is a move action that changes its location. We can describe this action domain in $\mathcal{AC}$ as follows.

$$\textbf{always } \mathit{Loc}(x,1) \vee \mathit{Loc}(x,2) \vee \mathit{Loc}(x,3) \vee \mathit{Loc}(x,4)$$
$$\textbf{always } \neg\mathit{Loc}(x,m) \vee \neg\mathit{Loc}(x,n) \qquad (m \neq n)$$
$$\textbf{never } \mathit{Loc}(A,n) \wedge \mathit{Loc}(B,n)$$
$$\mathit{Move}(x) \textbf{ causes } \neg\mathit{Loc}(x,n) \textbf{ if } \mathit{Loc}(x,n)$$

This domain description entails, for instance, the value propositions

$$(\,\textbf{initially } \mathit{Loc}(A,1) \wedge \mathit{Loc}(B,2)\,) \supset (\,\mathit{Loc}(A,3) \not\equiv \mathit{Loc}(A,4) \textbf{ after } \mathit{Move}(A)\,)$$

and

$$(\,\mathit{Loc}(A,1) \textbf{ after } \mathit{Move}(A)\,) \supset \textbf{ initially } \neg\mathit{Loc}(B,1)\,.$$

Sufficiency propositions are closely related to inference rules, as is apparent from the definition of *Res* in the semantics of $\mathcal{AC}$. As we mentioned in Chapter 2, Brewka and Hertzberg [BH93] also use inference rules to encode causal background

knowledge for reasoning about action. Their definition differs markedly from ours though. For instance, as we point out in [MT95b], their approach cannot capture the notion of qualification constraints. In fact, it sometimes yields different results even when derived action preconditions are not involved. For example, consider the following $\mathcal{AC}$ domain description.

$$\textbf{initially } \neg \mathit{HaveWine} \wedge \neg \mathit{WineOnTable} \wedge \neg \mathit{WaterOnTable}$$

$$\neg \mathit{HaveWine} \textbf{ suffices for } \neg \mathit{WineOnTable}$$

$$\mathit{ServeBeverage} \textbf{ causes } \mathit{WaterOnTable} \vee \mathit{WineOnTable}$$

This domain entails

$$\mathit{WaterOnTable} \wedge \neg \mathit{HaveWine} \wedge \neg \mathit{WineOnTable} \textbf{ after } \mathit{ServeBeverage}$$

while the corresponding question is resolved differently under the definition of Brewka and Hertzberg, according to which it is possible that wine will appear on the table, and, as an indirect effect, you will—somewhat miraculously—have wine. Intuitively, the weakness of their definition is that it still relies on a principle of minimal change, and thus fails to capture adequately the causal nature of the commonsense law of inertia. Consequently, in some cases, intuitively uncaused changes are sanctioned simply because they are minimal.

In other recent related work, Baral [Bar95] proposes an action description language based closely upon revision programming, which, as we have already mentioned, can be seen as a special case of the definition of possible next states used in $\mathcal{AC}$. Unfortunately, the semantics of Baral's action language is given directly in terms of a translation into disjunctive logic programs, which in general are relatively difficult to reason about. Nonetheless, it is possible to show that where his proposal overlaps with ours, it agrees.

Lin [Lin95] introduces a circumscriptive approach to causal theories of action that is closely related to his previous work with Reiter [LR94]. Lin shows that for

a special class of action descriptions—those he calls "stratified"—the meaning of a description can be obtained by a straightforward completion process. In the general case though, the semantics of Lin's action descriptions is given in terms of a multi-step minimization process. In the special case of Lin's "stratified" action descriptions, it is again possible to show that his proposal will agree with ours.[14]

Thielscher [Thi95a, Thi97] extends previous work, by himself and his colleagues, on reasoning about action in the formalism of equational logic programming. His proposal involves the use of state constraints accompanied by auxiliary information about directional, causal relationships between pairs of fluent atoms. The semantics of his action description language is given by a definition that is essentially procedural, and in fact seems motivated by computational (rather than declarative) concerns. It is unclear to what extent his proposal is related to ours.[15]

One clear advantage of the action description language $\mathcal{AC}$ over those of [Bar95, Lin95, Thi95a] is that it allows the use of arbitrary propositional formulas in the description of static causal laws and effects of actions. This makes it possible to express traditional ramification constraints, for instance. Also, recall that such formulas are used when explicit definitions are introduced. Another advantage is that $\mathcal{AC}$ has a relatively transparent semantics, specially tailored for action domains, in which there is a simple definition of possible next states that is used in a straightforward manner to constrain a situation calculus model structure.

We conclude this section with three results concerning general mathematical properties of $\mathcal{AC}$, modeled on similar results for the language $\mathcal{AR}$ in [GKL95].

Theorem 3.6 (Replacement Theorem) *Let D be an $\mathcal{AC}$ domain description. Let T be a subset of the sufficiency propositions in D. Take*

$$R_T = \left\{ \frac{\phi}{\psi} : \phi \textbf{ suffices for } \psi \in T \right\}.$$

[14] We will consider Lin's work more closely in the second part of this dissertation.

[15] A weak result along these lines appears in [Thi97].

Let ϕ, ϕ' be fluent formulas such that $(\phi \equiv \phi') \in Cn(R_T)$. Let D' be an $\mathcal{AC}$ domain description obtained from D by replacing some or all occurrences of ϕ with ϕ' in some or all propositions that do not belong to T. Domain descriptions D and D' have the same models.

Proof Sketch. Let R be the set of inference rules corresponding to the sufficiency propositions in domain description D, and let R' be the analagous set for D'. The key to this theorem is the observation that for any sets Γ, Γ' of formulas, if

$$Cn(\Gamma \cup R_T) = Cn(\Gamma' \cup R_T)$$

then

$$Cn(\Gamma \cup R) = Cn(\Gamma' \cup R')$$

which is not hard to prove. This implies, for instance, that the two domains have the same set of states. Furthermore, this observation can be used to show that the two domains agree on possible next states (that is, on *Res*). From these facts it follows that a structure Ψ satisfies the first three model conditions for domain D if and only if it satisfies them for domain D'. Consider such a structure Ψ. Since all states in the two domains satisfy $\phi \equiv \phi'$, it is clear that all of the value propositions in D are true in Ψ if and only if all of the value propositions in D' are true in Ψ. So D and D' indeed have the same models. □

Recall that in Section 3.3.4 we showed that (a slight simplification of) the definition of possible next states in $\mathcal{AC}$ handles explicit definitions in an appropriate manner (Proposition 3.4). Here we present a similar result for the language $\mathcal{AC}$ as a whole.

Theorem 3.7 (Explicit Definitions Theorem) *Let D be an $\mathcal{AC}$ domain description, with fluents $\mathbf{F}$, in which there is an explicit definition* **always** $F \equiv \phi$ *and furthermore there is no influence proposition A* **possibly changes** F **if** ψ. *Let D'*

be the domain description with fluents $\mathbf{F} \setminus \{F\}$ that can obtained from D by deleting the explicit definition **always** $F \equiv \phi$ *and replacing all remaining occurrences of F with ϕ. For every value proposition V in which F does not occur, V is true in D if and only if V is true in D'.*

Proof Sketch. Let D'' be the domain description, with fluents $\mathbf{F}$, obtained from D' by adding only the explicit definition **always** $F \equiv \phi$. We know from the Replacement Theorem that domains D and D'' have the same models. The proof can be completed by showing that the desired result holds between domains D'' and D', as follows.

For any interpretation I of $\mathcal{L}(\mathbf{F} \setminus \{F\})$, let $F(I)$ be the interpretation of $\mathcal{L}(\mathbf{F})$ such that $I \subseteq F(I)$ and $F(I) \models F \equiv \phi$. It is clear that this establishes a one-to-one correspondence between the states of D' and D''. Moreover, it is straightforward to show, along the lines of Proposition 3.4, that there is a similar one-to-one correspondence between the definitions of *Res* in the two domains. From these facts it follows that there is also a similar one-to-one correspondence between the models of D' and D'', which is sufficient to show that for every value proposition V in which F does not occur, V is true in D'' if and only if V is true in D'. □

Theorem 3.8 (Restricted Monotonicity Theorem)[16] *Let D be an $\mathcal{AC}$ domain description. If D' can be obtained by adding value propositions to D, then every value proposition entailed by D is also entailed by D'.*

Proof. Immediate, since adding value propositions can only rule out models. □

3.5 Representing Actions in Default Logic

We begin this section by reviewing the definition of default logic. Next, as a preliminary step, we show how to embed the $\mathcal{AC}$ definition of possible next states—that is,

[16] See [Lif93b] for a general account of restricted monotonicity.

the function *Res*—in default logic. We then define a class of $\mathcal{AC}$ domain descriptions called "qualification-free." Roughly speaking, in qualification-free domain descriptions, all action preconditions are stated "explicitly," in the form of executability propositions. We specify a sound and complete translation of qualification-free $\mathcal{AC}$ domain descriptions into default theories. We also specify a second, simpler translation for those $\mathcal{AC}$ domain descriptions in which there are no action preconditions whatsoever. Finally, we compare the formalization of the Yale Shooting domain obtained by our translation with the default theories discussed by Hanks and McDermott and by Morris.

3.5.1 Review of Default Logic

A *default rule* over $\mathcal{L}(U)$ is an expression of the form

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma} \tag{3.1}$$

where all of $\alpha, \beta_1, \ldots, \beta_n, \gamma$ are formulas from $\mathcal{L}(U)$ ($n \geq 0$). Let r be a default rule of the form (3.1). We call α the *prerequisite* of r, and denote it by $pre(r)$. We call the formulas $\beta_1, \ldots, \beta_n$ the *justifications* of r, and write $just(r)$ to denote the set $\{\beta_1, \ldots, \beta_n\}$. We call γ the *consequent* of r, and denote it by $cons(r)$. If $just(r)$ is empty, we say r is *justification-free*. If r is justification-free, we often identify r with the corresponding inference rule

$$\frac{\alpha}{\gamma}.$$

If $pre(r) = True$ we often omit it and write $\frac{: \beta_1, \ldots, \beta_n}{\gamma}$ instead. If $just(r) = \{cons(r)\}$, we say that r is *normal*.

A *default theory* over $\mathcal{L}(U)$ is a set of default rules over $\mathcal{L}(U)$. Let D be a default theory over $\mathcal{L}(U)$ and let E be a set of formulas from $\mathcal{L}(U)$. We define the *reduct of D by E*, denoted by D^E, as follows.

$$D^E = \left\{ \frac{pre(r)}{cons(r)} \ : \ r \in D \ and \ for \ all \ \ \beta \in just(r) \ , \ \neg\beta \notin E \right\}$$

We say that E is an *extension* of D if

$$E = Cn_U(D^E).$$

We say D is *consistent* if it has at least one consistent extension. We say that a formula is a *consequence* of D if it belongs to every extension of D. Default logic is due to Reiter [Rei80]. The definition of an extension given above follows [GLPT91], and is equivalent to Reiter's definition.

3.5.2 Embedding Possible Next States in Default Logic

The embedding in this section is not directly used in specifying the more general embedding of $\mathcal{AC}$ into default logic in the next section, and thus this section can safely be skipped. On the other hand, embedding in default logic the function *Res*—which defines the possible next states in $\mathcal{AC}$—is a clearly related, smaller problem, and some of the ideas used here are also applied in the next section. Moreover, the fact that this embedding is correct is a fundamental theorem used in the proof (presented in Chapter 4) of the correctness of the $\mathcal{AC}$ embedding .

Let D be an $\mathcal{AC}$ domain description. For any state S and action name A, let $\Delta(A, S)$ be the default theory obtained by taking the union of the following four sets of rules. (Recall that, roughly speaking, $E(A, S)$ is the set of direct, deterministic effects of action A in state S. Similarly, $F(A, S)$ is the set of fluents that may be nondeterministically affected by action A in state S. R is the set of inference rules corresponding to the static causal laws of the domain.)

1. All rules of the form $\frac{: L}{L}$ where L is a frame fluent literal in S.
2. $E(A, S)$
3. All rules of the forms $\frac{: F}{F}$ and $\frac{: \neg F}{\neg F}$ where $F \in F(A, S)$.
4. R

Notice that $\Delta(A,S)$ is a default theory over $\mathcal{L}(\mathbf{F})$. The following theorem shows that $\Delta(A,S)$ characterizes the possible next states as defined in $Res(A,S)$.

Theorem 3.9 *Let S be a state and A an action that is not prohibited in S. The following hold.*

1. *A state S' belongs to $Res(A,S)$ if and only if $Cn_{\mathbf{F}}(S')$ is a consistent extension of $\Delta(A,S)$.*
2. *Every consistent extension of $\Delta(A,S)$ can be written in the form $Cn_{\mathbf{F}}(S')$, where S' is a state.*

Proof. For the first part, let S' be a state, and let

$$E = E(A,S) \cup (S' \cap Lit(F(A,S))).$$

Observe that E is an explicit effect of A in S. It is not difficult to verify that

$$\Delta(A,S)^{Cn(S')} = (S \cap S' \cap Lit(\mathbf{F}_f)) \cup E \cup R.$$

Thus we see that $Cn(S')$ is a consistent extension of $\Delta(A,S)$ if and only if $Cn(S') = Cn[(S \cap S' \cap Lit(\mathbf{F}_f)) \cup E \cup R]$.

Since E is an explicit effect of A in S, we have shown that if $Cn(S')$ is a consistent extension of $\Delta(A,S)$ then $S' \in Res(A,S)$. To see the other direction, assume that $S' \in Res(A,S)$. Thus there is an explicit effect E' of A in S such that $Cn(S') = Cn[(S \cap S' \cap Lit(\mathbf{F}_f)) \cup E' \cup R]$. It is clear that $E' = E(A,S) \cup (S' \cap Lit(F(A,S)))$, which is to say that $E' = E$. Thus we can conclude that $Cn(S')$ is a consistent extension of $\Delta(A,S)$.

For the second part, assume that X is a consistent extension of $\Delta(A,S)$. Suppose there is a fluent name F such that $F \notin X$ and $\neg F \notin X$. Since every nonframe fluent in an $\mathcal{AC}$ domain description must have a definition in terms of

frame fluents, we can assume without loss of generality that F is a frame fluent. But in this case, since S is a state, $\Delta(A,S)$ includes one of the following two rules.

$$\frac{:F}{F} \quad \frac{:\neg F}{\neg F}$$

From this we can conclude that $Cn(\Delta(A,S)^X)$ includes either F or $\neg F$. This shows that $Cn(\Delta(A,S)^X) \neq X$. Contradiction. So we have shown that for every fluent name F, either $F \in X$ or $\neg F \in X$. And since X is consistent, it follows that there is an interpretation S' of $\mathcal{L}(\mathbf{F})$ such that $X = Cn(S')$. Now, since $\Delta(A,S)$ contains the inference rules R, we know that $Cn(S')$ is closed under R. So S' is a state. □

This embedding of possible next states in default logic is closely related to the embeddings of rule update in default logic in [PT95, PT97]. As previously mentioned, this theorem is useful in proving the $\mathcal{AC}$ embedding correct, and it also demonstrates (in slightly different form) some of the ideas behind that embedding.

3.5.3 Embedding $\mathcal{AC}$ in Default Logic

We say that an $\mathcal{AC}$ domain description is *qualification-free* if for all action names A and states S, A is prohibited in S whenever $Res(A,S)$ is empty.

Our default theories for reasoning about action use the situation calculus. For any fluent formula ϕ, we write $Holds(\phi, S)$ to stand for the formula obtained by replacing each fluent atom F in ϕ by $Holds(F,S)$. Given an action string $A_1;\cdots;A_m$, we write

$$[A_1;\cdots;A_m]$$

to stand for the term

$$Result(A_m, Result(A_{m-1}, \ldots Result(A_1, S_0)\cdots)).$$

Given an atomic value proposition ϕ **after** $\overline{A}$, we write

$$[\phi \textbf{ after } \overline{A}]$$

to stand for the formula

$$(\, Holds(\phi, [\overline{A}]) \wedge Reachable([\overline{A}])\,).$$

Given a (non-atomic) value proposition V, we write $[V]$ to stand for the formula obtained by simultaneously replacing each atomic value proposition V' that occurs in V by the formula $[V']$.

The translation δ takes an $\mathcal{AC}$ domain description D to a default theory $\delta(D)$ over the language $\mathcal{L}(U)$, where U is the least set of atoms such that, for every action string $\overline{A}$: (i) $Reachable([\overline{A}]) \in U$; (ii) for every fluent name F, $Holds(F, [\overline{A}]) \in U$.

For each value proposition V in D, $\delta(D)$ includes the rule

$$\frac{\neg[V]}{False}.$$

For each sufficiency proposition ϕ **suffices for** ψ in D, $\delta(D)$ includes the rule

$$\frac{Holds(\phi, s) \wedge Reachable(s)}{Holds(\psi, s)}.$$

For each effect proposition A **causes** ϕ **if** ψ in D, $\delta(D)$ includes the rule

$$\frac{Holds(\psi, s) \wedge Reachable(Result(A, s))}{Holds(\phi, Result(A, s))}.$$

For each influence proposition A **possibly changes** F **if** ψ in D, $\delta(D)$ includes the pair of rules

$$\frac{Holds(\psi, s) \wedge Reachable(Result(A, s)) : Holds(F, Result(A, s))}{Holds(F, Result(A, s))}$$

and

$$\frac{Holds(\psi, s) \wedge Reachable(Result(A, s)) : \neg Holds(F, Result(A, s))}{\neg Holds(F, Result(A, s))}.$$

For each executability proposition **impossible** A **if** ψ in D, $\delta(D)$ includes the rule

$$\frac{Holds(\psi, s)}{\neg Reachable(Result(A, s))}.$$

Default theory $\delta(D)$ also includes the additional, standard rules shown in Figure 3.6.

The following theorem shows that the translation δ is indeed sound and complete for qualification-free $\mathcal{AC}$ domain descriptions.

Reachability axioms. Default theory $\delta(D)$ includes the rules

$$\frac{: Reachable(s)}{Reachable(s)} \quad \text{and} \quad \frac{\neg Reachable(s)}{\neg Reachable(Result(a, s))}.$$

Initial situation axioms. For each fluent literal L, $\delta(D)$ includes the rule

$$\frac{: Holds(L, S_0)}{Holds(L, S_0)}.$$

Inertia axioms. For each frame fluent literal L, $\delta(D)$ includes the rule

$$\frac{Holds(L, s) \wedge Reachable(Result(a, s)) : Holds(L, Result(a, s))}{Holds(L, Result(a, s))}.$$

Figure 3.6: Standard elements of the translation δ.

Theorem 3.10 (Embedding Theorem) *Let D be a qualification-free $\mathcal{AC}$ domain description. A value proposition V is entailed by D if and only if the formula $[V]$ is entailed by the default theory $\delta(D)$.*

The Embedding Theorem is an immediate consequence of the following stronger theorem, which is proved in Chapter 4.

Theorem 3.11 (Correspondence Theorem) *Let D be a qualification-free $\mathcal{AC}$ domain description. There is a one-to-one correspondence between models of D and consistent extensions of $\delta(D)$ such that, for every model Ψ of D and its corresponding extension E, a value proposition V is true in Ψ if and only if $[V] \in E$.*

For example, recall domain description D_1 from Section 3.4, which is reproduced here in Figure 3.7. In this $\mathcal{AC}$ domain description, *Dead* is the only nonframe fluent. Domain description D_1 entails, for instance, the value proposition

initially *Loaded*

always $Dead \equiv \neg Alive$
initially $Walking$
$\neg Walking$ **after** $Shoot$
$\neg Alive$ **suffices for** $\neg Walking$
$Shoot$ **causes** $Dead \wedge \neg Loaded$
impossible $Shoot$ **if** $\neg Loaded$

Figure 3.7: $\mathcal{AC}$ domain description D_1.

and the Embedding Theorem guarantees that the corresponding formula

$$Holds(Loaded, S_0) \wedge Reachable(S_0)$$

is entailed by the corresponding default theory $\delta(D_1)$, listed in Figure 3.8.

If a domain description includes no executability propositions, we can eliminate the *Reachable* atoms in the corresponding default theory, thus obtaining a simpler translation, as follows. Let D be an $\mathcal{AC}$ domain description. By $\delta'(D)$ we denote the default theory obtained from $\delta(D)$ by first eliminating the reachability axioms and then replacing each *Reachable* atom in the remaining rules by the special atom *True*. Of course it is then straightforward to eliminate the resulting occurrences of *True* in the resulting default theory. Notice that the default theories in Examples 1 and 2 (Section 3.2) can be obtained by translation δ'.

For each atomic value proposition ϕ **after** $\overline{A}$, let

$$[\![\phi \textbf{ after } \overline{A}]\!]$$

denote the formula

$$Holds(\phi, [\overline{A}])$$

and for each (non-atomic) value proposition V, let $[\![V]\!]$ be the formula obtained from V by simultaneously replacing each atomic value proposition V' that occurs in V by the formula $[\![V']\!]$.

$$\frac{True \wedge Reachable(s)}{Holds(Dead, s) \equiv \neg Holds(Alive, s)} \quad \frac{\neg(Holds(Walking, S_0) \wedge Reachable(S_0))}{False}$$

$$\frac{\neg(\neg Holds(Walking, Result(Shoot, S_0)) \wedge Reachable(Result(Shoot, S_0)))}{False}$$

$$\frac{\neg Holds(Alive, s) \wedge Reachable(s)}{\neg Holds(Walking, s)} \quad \frac{\neg Holds(Loaded, s)}{\neg Reachable(Result(Shoot, s))}$$

$$\frac{True \wedge Reachable(Result(Shoot, s))}{Holds(Dead, Result(Shoot, s)) \wedge \neg Holds(Loaded, Result(Shoot, s))}$$

$$\frac{: Holds(f, S_0)}{Holds(f, S_0)} \quad \frac{: \neg Holds(f, S_0)}{\neg Holds(f, S_0)} \quad \frac{: Reachable(s)}{Reachable(s)} \quad \frac{\neg Reachable(s)}{\neg Reachable(Result(a, s))}$$

$$\frac{Holds(Alive, s) \wedge Reachable(Result(a, s)) : Holds(Alive, Result(a, s))}{Holds(Alive, Result(a, s))}$$

$$\frac{\neg Holds(Alive, s) \wedge Reachable(Result(a, s)) : \neg Holds(Alive, Result(a, s))}{\neg Holds(Alive, Result(a, s))}$$

$$\frac{Holds(Loaded, s) \wedge Reachable(Result(a, s)) : Holds(Loaded, Result(a, s))}{Holds(Loaded, Result(a, s))}$$

$$\frac{\neg Holds(Loaded, s) \wedge Reachable(Result(a, s)) : \neg Holds(Loaded, Result(a, s))}{\neg Holds(Loaded, Result(a, s))}$$

$$\frac{Holds(Walking, s) \wedge Reachable(Result(a, s)) : Holds(Walking, Result(a, s))}{Holds(Walking, Result(a, s))}$$

$$\frac{\neg Holds(Walking, s) \wedge Reachable(Result(a, s)) : \neg Holds(Walking, Result(a, s))}{\neg Holds(Walking, Result(a, s))}$$

Figure 3.8: Translation $\delta(D_1)$ of $\mathcal{AC}$ domain description D_1.

Corollary 3.12 (Reachability Corollary) *Let D be a qualification-free $\mathcal{AC}$ domain description with no executability propositions. There is a one-to-one correspondence between models of D and consistent extensions of $\delta'(D)$ such that, for every model Ψ of D and its corresponding extension E, a value proposition V is true in Ψ if and only if $[\![V]\!] \in E$.*

3.5.4 The Yale Shooting Problem in Default Logic

At this point it may be interesting to briefly consider some of the ways in which our default theory for the Yale Shooting domain differs from the one proposed and found inadequate by Hanks and McDermott [HM87], and from the more adequate solution later proposed by Morris [Mor88].

We can represent the origial Yale Shooting domain in $\mathcal{AC}$ as follows.[17]

initially *Alive*

Load **causes** *Loaded*

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

Of course this domain description entails the $\mathcal{AC}$ value proposition

$$\neg Alive \textbf{ after } Load; Wait; Shoot$$

and accordingly, we know by the Reachability Corollary that the corresponding literal

$$\neg Holds(Alive, Result(Shoot, Result(Wait, Result(Load, S_0))))$$

is a consequence of the corresponding default theory Y_1, which appears in Figure 3.9.

By comparison, the default theory that was introduced and rejected by Hanks and McDermott is (essentially) the default theory Y_2 that appears in Figure 3.10.

[17] This version of the Yale Shooting domain, which is more elaborate than the one discussed in Chapter 1, is faithful to the description given by Hanks and McDermott.

$$\frac{\neg Holds(Alive, S_0)}{False}$$

$$\frac{True}{Holds(Loaded, Result(Load, s))} \qquad \frac{Holds(Loaded, s)}{\neg Holds(Alive, Result(Shoot, s))}$$

$$\frac{: Holds(f, S_0)}{Holds(f, S_0)} \qquad \frac{: \neg Holds(f, S_0)}{\neg Holds(f, S_0)}$$

$$\frac{Holds(f, s) : Holds(f, Result(a, s))}{Holds(f, Result(a, s))} \qquad \frac{\neg Holds(f, s) : \neg Holds(f, Result(a, s))}{\neg Holds(f, Result(a, s))}$$

Figure 3.9: Default theory Y_1.

$$\frac{True}{Holds(Alive, S_0)} \qquad \frac{True}{Ab(Loaded, Load, s) \wedge Holds(Loaded, Result(Load, s))}$$

$$\frac{True}{Holds(Loaded, s) \supset (Ab(Alive, Shoot, s) \wedge \neg Holds(Alive, Result(Shoot, s)))}$$

$$\frac{True}{(Holds(f, s) \wedge \neg Ab(f, a, s)) \supset Holds(f, Result(a, s))}$$

$$\frac{True}{(\neg Holds(f, s) \wedge \neg Ab(f, a, s)) \supset \neg Holds(f, Result(a, s))}$$

$$\frac{: \neg Ab(f, a, s)}{\neg Ab(f, a, s)}$$

Figure 3.10: Default theory Y_2.

As suggested by the discussion in Section 1.2.1, the well-known difficulty in default theory Y_2 can be attributed to the fact that the default rule $\frac{:\neg Ab(f,a,s)}{\neg Ab(f,a,s)}$ effectively minimizes the extent of Ab. We can observe that the two inertia rules together guarantee that all ground instances of the standard circumscriptive frame axiom

$$\neg Ab(f,a,s) \supset (Holds(f,s) \equiv Holds(f,Result(a,s))$$

belong to every extension of Y_2. These three default rules together then can be understood to minimize the instances of

$$Holds(f,s) \not\equiv Holds(f,Result(a,s))$$

in the extensions of Y_2. In this way Y_2 minimizes global change.

But there is another way to describe what goes wrong here, which brings into relief a second point of interest (from our point of view). The default rule that minimizes Ab allows the default "supposition"

$$\neg Ab(Alive, Shoot, Result(Wait, Result(Load, S_0)))\,.$$

We can then reason "backward in time," using the rule describing the effect of *Shoot*, to derive

$$\neg Holds(Loaded, Result(Wait, Result(Load, S_0)))\,.$$

And from this fact, we can again reason "backward," using the inertia rule, to obtain

$$Ab(Loaded,\ Wait, Result(Load, S_0))\,.$$

It is this combination of "supposing" that certain changes do not occur and reasoning backward in time that allows us to obtain the extensions of Y_2 that correspond to the famous anomaly—according to which the gun may become mysteriously unloaded during the wait action.

There is another peculiarity to be noted here, related to the fact that in the Yale Shooting domain (as originally described by Hanks and McDermott) we

$$\frac{True}{Holds(Alive,S_0)} \qquad \frac{True}{Ab(Loaded,Load,s) \wedge Holds(Loaded,Result(Load,s))}$$

$$\frac{True}{Holds(Loaded,s) \supset (Ab(Alive,Shoot,s) \wedge \neg Holds(Alive,Result(Shoot,s)))}$$

$$\frac{Holds(f,s) : \neg Ab(f,a,s)}{Holds(f,Result(a,s))} \qquad \frac{\neg Holds(f,s) : \neg Ab(f,a,s)}{\neg Holds(f,Result(a,s))}$$

Figure 3.11: Default theory Y_3.

are not told whether or not the gun is initially loaded. Accordingly, the $\mathcal{AC}$ domain description entails neither **initially** *Loaded* nor **initially** $\neg$*Loaded*. From the Reachability Corollary it follows, for instance, that our default theory Y_1 does not entail the literal $\neg Holds(Loaded, S_0)$.

By comparison, in the default theory Y_2 of Hanks and McDermott, we can "suppose"

$$\neg Ab(Alive, Shoot, S_0)$$

and from this default supposition we can derive

$$\neg Holds(Loaded, S_0)$$

again by reasoning "backwards in time," using the rule describing the effect of the shoot action. This observation suggests that $\neg Holds(Loaded, S_0)$ may belong to some extension of the default theory. This is not unexpected, since the $\mathcal{AC}$ domain description itself has a model in which *Loaded* is initially false. But as it turns out, $\neg Holds(Loaded, S_0)$ belongs to every extension of the default theory of Hanks and McDermott, which is therefore not only incomplete for the Yale Shooting domain, but also unsound.

The default theory proposed by Morris for the Yale Shooting domain is (essentially) the default theory Y_3 shown in Figure 3.11. In Morris' default theory we can again reason "backwards in time," using the rule describing the effect of the action *Shoot*. But notice that there is now no default rule allowing us to "suppose"

literals of the form $\neg Ab(f, a, s)$. Moreover, there is no opportunity for "inappropriately" deriving atoms of the form $Ab(f, a, s)$ by reasoning backwards in time. Thus the famous anomaly is eliminated. On the other hand, it turns out that the formula $\neg Holds(Loaded, S_0)$ is once again inappropriately entailed. To see this, notice first that we cannot derive the literal

$$Ab(Alive, Shoot, S_0)$$

in default theory Y_3. Because of this, we are able to derive

$$Holds(Alive, Result(Shoot, S_0))$$

using one of the default rules expressing the commonsense law of inertia. And, from this, we can derive

$$\neg Holds(Loaded, S_0)$$

by reasoning backwards in time, using the rule describing the effect of the shoot action. Thus, Morris' default theory for the Yale Shooting domain is, apparently, complete but unsound.

In our translations of $\mathcal{AC}$ into default logic, as emphasized in the discussion in Section 3.2, we never write default rules making past facts derivable from future facts. Such rules would not, in general, make sense for us, since we think of these rules as expressing a simple kind of causal relation. Moreover, our discussion of the Hanks and McDermott and Morris default theories (correctly) suggests that our practice is technically useful. In fact, our use of the directional properties of default rules is essential to the proof of the Correspondence Theorem. Roughly speaking, it allows us to show that our general default theories correctly embed our causal definition of possible next states, by guaranteeing that in our default theories future facts cannot affect past facts.

It may or may not be helpful to point out also that the observed unsoundness of default theories Y_2 and Y_3 can be overcome simply by adding to them the following

rules enforcing completeness of the initial situation.

$$\frac{: Holds(f, S_0)}{Holds(f, S_0)} \quad \frac{: \neg Holds(f, S_0)}{\neg Holds(f, S_0)}$$

Recall that these rules are standard elements of our translations from $\mathcal{AC}$ into default logic. In a manner of speaking, they interact to force a default theory to take into account every possible (complete!) initial situation.

3.6 Logic Programs for Representing Actions

We begin this section by reviewing the relevant definitions in logic programming. We then identify a syntactically restricted class of $\mathcal{AC}$ domain descriptions for which the translation into logic programming is particularly convenient. We call such domain descriptions "LP-simple." After specifying a sound and complete translation of LP-simple, qualification-free domain descriptions into logic programming, we introduce the somewhat broader class of "vivid" $\mathcal{AC}$ domain descriptions, and show how vivid, qualification-free domain descriptions can be transformed into equivalent LP-simple, qualification-free domain descriptions. Thus we obtain a correct embedding in logic programming for all vivid, qualification-free $\mathcal{AC}$ domain descriptions.

In the case of value propositions, the translation into logic program rules is more complicated than the translation into default rules specified in the previous section. For all other $\mathcal{AC}$ propositions, the translation is essentially the same.

3.6.1 Review of Logic Programming

For the purposes of this dissertation, a *logic program rule* over $\mathcal{L}(U)$ is an expression of the form

$$L_0 \quad \leftarrow \quad L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n \tag{3.2}$$

with $0 \leq m \leq n$, where $L_0 \in Lit(U) \cup \{False\}$ and for all i $(1 \leq i \leq n)$, $L_i \in Lit(U)$. A *logic program* over $\mathcal{L}(U)$ is a set of logic program rules over $\mathcal{L}(U)$.

Under the answer set semantics of Gelfond and Lifschitz [GL90], logic programming corresponds to a subset of default logic.[18] Because it is convenient for the purposes of this dissertation, we will define the notions of "answer sets" and "entailment" for logic programs indirectly, in terms of the related notions for default logic.

For each logic program P there is a corresponding default theory $dt(P)$, defined as follows. For each logic program rule $r \in P$ of form (3.2), $dt(P)$ includes the corresponding default rule

$$\frac{L_1 \wedge \cdots \wedge L_m : \overline{L_{m+1}}, \ldots, \overline{L_n}}{L_0}.$$

A subset X of $Lit(U)$ is an *answer set* for P if there is an extension E of $dt(P)$ such that $X = E \cap Lit(U)$. It follows that X is a consistent answer set for P if and only if $Cn(X)$ is a consistent extension of $dt(P)$. For any $L \in Lit(U)$, we say that P *entails* L if L belongs to all answer sets for P. It follows that P entails L if and only if $dt(P)$ does.

3.6.2 LP-Simple $\mathcal{AC}$ Domain Descriptions

An atomic value proposition is *LP-simple* if it has the form

$$L \textbf{ after } \overline{A}$$

where L is a fluent literal, and a (non-atomic) value proposition is *LP-simple* if it has the form

$$V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n \qquad (0 \leq m \leq n, n > 0)$$

where each V_i $(1 \leq i \leq n)$ is an LP-simple atomic value proposition.

A sufficiency proposition is *LP-simple* if it has either the form

[18] Logic programs of this kind are also reducible (as shown in [GL90]) to normal logic programs under the stable model semantics [GL88].

$$L_1 \wedge \cdots \wedge L_n \textbf{ suffices for } L_0 \qquad (n > 0)$$

where each L_i $(0 \leq i \leq n)$ is a fluent literal, or the form

$$\textbf{always } L$$

where L is a fluent literal, or the form

$$\textbf{never } L_1 \wedge \cdots \wedge L_n \qquad (n > 0)$$

where each L_i $(1 \leq i \leq n)$ is again a fluent literal.

An effect proposition is *LP-simple* if it has either the form

$$A \textbf{ causes } L_0 \textbf{ if } L_1 \wedge \cdots \wedge L_n \qquad (n > 0)$$

where each L_i $(0 \leq i \leq n)$ is a fluent literal, or the form

$$A \textbf{ causes } L$$

where L is a fluent literal.

An influence proposition is *LP-simple* if it has either the form

$$A \textbf{ possibly changes } F \textbf{ if } L_1 \wedge \cdots \wedge L_n \qquad (n > 0)$$

where each L_i $(1 \leq i \leq n)$ is a fluent literal, or the form

$$A \textbf{ possibly changes } F.$$

Finally, an executability proposition is *LP-simple* if it has the form

$$\textbf{impossible } A \textbf{ if } L_1 \wedge \cdots \wedge L_n \qquad (n > 0)$$

where each L_i $(1 \leq i \leq n)$ is a fluent literal.

We say that an $\mathcal{AC}$ domain description is *LP-simple* if all of its propositions are. Perhaps the most severe restriction on LP-simple domain descriptions is that

they cannot include explicit definitions, due to the restricted form of sufficiency propositions. Notice that three of the four example domain descriptions considered in Section 3.2 are in fact LP-simple domain descriptions.

3.6.3 LP-Simple $\mathcal{AC}$ Domain Descriptions as Logic Programs

Let D be an LP-simple $\mathcal{AC}$ domain description. We define its translation into a logic program $\pi(D)$ as follows.

For each value proposition $V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$ in D, include the rule

$$False \quad \leftarrow \quad [\![V_{m+1}]\!], \ldots, [\![V_n]\!], not\, [\![V_1]\!], \ldots, not\, [\![V_m]\!] .$$

For each sufficiency proposition in D of the form $L_1 \wedge \cdots \wedge L_n$ **suffices for** L_0 include the rule

$$Holds(L_0, s) \quad \leftarrow \quad Holds(L_1, s), \ldots, Holds(L_n, s), Reachable(s) .$$

For each sufficiency proposition in D of the form **always** L include the rule

$$Holds(L, s) \quad \leftarrow \quad Reachable(s) .$$

For each sufficiency proposition in D of the form **never** $L_1 \wedge \cdots \wedge L_n$ include the rule

$$False \quad \leftarrow \quad Holds(L_1, s), \ldots, Holds(L_n, s), Reachable(s) .$$

For each effect proposition in D of the form A **causes** L_0 **if** $L_1 \wedge \cdots \wedge L_n$ include the rule

$$Holds(L_0, Result(A, s)) \quad \leftarrow \quad Holds(L_1, s), \ldots, Holds(L_n, s), Reachable(Result(A, s)) .$$

For each effect proposition in D of the form A **causes** L include the rule

$$Holds(L, Result(A, s)) \quad \leftarrow \quad Reachable(Result(A, s)) .$$

For each influence proposition in D of the form

$$A \textbf{ possibly changes } F \textbf{ if } L_1 \wedge \cdots \wedge L_n$$

include the following two rules.

$$\begin{aligned} Holds(F, Result(A, s)) \quad &\leftarrow \quad Holds(L_1, s), \ldots, Holds(L_n, s), \\ &\qquad Reachable(Result(A, s)), not\, \neg Holds(F, Result(A, s)) \\ \neg Holds(F, Result(A, s)) \quad &\leftarrow \quad Holds(L_1, s), \ldots, Holds(L_n, s), \\ &\qquad Reachable(Result(A, s)), not\, Holds(F, Result(A, s)) \end{aligned}$$

For each influence proposition in D of the form A **possibly changes** F include the following two rules.

$$\begin{aligned} Holds(F, Result(A, s)) \quad &\leftarrow \quad Reachable(Result(A, s)), not\, \neg Holds(F, Result(A, s)) \\ \neg Holds(F, Result(A, s)) \quad &\leftarrow \quad Reachable(Result(A, s)), not\, Holds(F, Result(A, s)) \end{aligned}$$

Finally, for each executability proposition **impossible** A **if** $L_1 \wedge \cdots \wedge L_n$ in D, include the rule

$$\neg Reachable(s) \quad \leftarrow \quad Holds(L_1, s), \ldots, Holds(L_n, s) .$$

Also include the following six standard rules for reachability, completeness of the initial situation, and the commonsense law of inertia.

$$\begin{aligned} Reachable(s) \quad &\leftarrow \quad not\, \neg Reachable(s) \\ \neg Reachable(Result(a, s)) \quad &\leftarrow \quad \neg Reachable(s) \\ Holds(f, S_0) \quad &\leftarrow \quad not\, \neg Holds(f, S_0) \end{aligned}$$

$$\begin{aligned}
\neg Holds(f, S_0) &\leftarrow not\ Holds(f, S_0) \\
Holds(f, Result(a,s)) &\leftarrow Holds(f,s), Reachable(Result(a,s)), \\
&\qquad not\ \neg Holds(f, Result(a,s)) \\
\neg Holds(f, Result(a,s)) &\leftarrow \neg Holds(f,s), Reachable(Result(a,s)), \\
&\qquad not\ Holds(f, Result(a,s))
\end{aligned}$$

Notice that the logic program in the fourth example in Section 3.2 can be obtained by the translation π.

Theorem 3.13 (LP Embedding Theorem).
Let D be an LP-simple, qualification-free $\mathcal{AC}$ domain description. an LP-simple atomic value proposition L **after** *$\overline{A}$ is entailed by D if and only if $Holds(L, [\overline{A}])$ is entailed by the logic program $\pi(D)$.*

The LP Embedding Theorem is an immediate consequence of the following stronger theorem, which is proved in Chapter 4 using the Correspondence Theorem for default logic.

Theorem 3.14 (LP Correspondence Theorem)
Let D be an LP-simple, qualification-free $\mathcal{AC}$ domain description. There is a one-to-one correspondence between models of D and consistent answer sets of $\pi(D)$ such that, for every model Ψ of D and corresponding answer set X, an LP-simple value proposition

$$V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$$

is true in Ψ if and only if at least one of the sets $\{ [\![V_1]\!], \ldots, [\![V_m]\!] \} \cap X$ and $\{ [\![V_{m+1}]\!], \ldots, [\![V_n]\!] \} \setminus X$ is nonempty.

If an LP-simple domain description includes no executability propositions, we can eliminate the *Reachable* atoms in the corresponding logic program, thus obtaining a simpler translation. So let D be an LP-simple $\mathcal{AC}$ domain description without

executability propositions. By $\pi'(D)$ we denote the logic program obtained from $\pi(D)$ by first eliminating the reachability axioms and then deleting all *Reachable* atoms from the remaining rules. Notice that the logic program in the third example in Section 3.2 can be obtained by the translation π'.

Corollary 3.15 (LP Reachability Corollary)
Let D be an LP-simple, qualification-free $\mathcal{AC}$ domain description without executability propositions. There is a one-to-one correspondence between models of D and consistent answer sets of $\pi'(D)$ such that, for every model Ψ of D and corresponding answer set X, an LP-simple value proposition $V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$ is true in Ψ if and only if the set $\{ [\![V_1]\!], \ldots, [\![V_m]\!], \overline{[\![V_{m+1}]\!]}, \ldots, \overline{[\![V_n]\!]} \} \cap X$ is nonempty.

3.6.4 Making Vivid $\mathcal{AC}$ Domain Descriptions LP-Simple

The syntactic restrictions which define the class of LP-simple domain descriptions are, fortunately, more strict than necessary. In this section we show that a much broader class of $\mathcal{AC}$ domain descriptions can be embedded into logic programming.

We say that a sufficiency proposition is *vivid* if it has the form

$$\phi \textbf{ suffices for } \psi$$

where ψ is a conjunction of fluent literals. Similarly, we say that an effect proposition is *vivid* if it has the form

$$A \textbf{ causes } \phi \textbf{ if } \psi$$

where ϕ is a nonempty conjunction of fluent literals.

We say that a domain description is *vivid* if all of its sufficiency propositions and effect propositions are. Any vivid domain description can be transformed into an equivalent LP-simple domain description, in the manner described below.

We begin by assuming a function *CNF* that takes every fluent formula ϕ to an equivalent fluent formula $CNF(\phi)$ in conjunctive normal form. We also assume

a function *DNF* that takes every fluent formula ϕ to an equivalent fluent formula $DNF(\phi)$ in disjunctive normal form.

For any atomic value proposition ϕ **after** $\overline{A}$, let $CNF(\phi$ **after** $\overline{A})$ be the result of simultaneously replacing each disjunct L of each conjunct of $CNF(\phi)$ with the LP-simple atomic value proposition L **after** $\overline{A}$. Notice that ϕ **after** $\overline{A}$ is true in a structure Ψ if and only if $CNF(\phi$ **after** $\overline{A})$ is.

Next we describe a three-step transformation that takes any value proposition V to a corresponding family of LP-simple value propositions.

1. Let V_s be the result of simultaneously replacing each atomic value proposition V' that occurs in V with the value proposition $CNF(V')$. Notice that V_s is a propositional combination of LP-simple atomic value propositions. Notice also that V is true in a structure Ψ if and only if V_s is.

2. Let C be the set of conjuncts of the conjunctive normal form of V_s. Notice that each member of C is a disjunction of LP-simple atomic value propositions or their negations. Notice also that V_s is true in a structure Ψ if and only if every member of C is.

3. Take the set of value propositions obtained by reordering the literals of each member of C so that each of the resulting expressions is an LP-simple value proposition.

Observe that V is true in a structure Ψ if and only if all of the corresponding LP-simple value propositions are.

For any vivid sufficiency proposition ϕ **suffices for** ψ, take the family of LP-simple sufficiency propositions ϕ' **suffices for** L such that ϕ' is a disjunct of $DNF(\phi)$ and L is a conjunct of ψ.

For any vivid effect proposition A **causes** ϕ **if** ψ, take the family of LP-simple effect propositions A **causes** L **if** ψ' such that L is a conjunct of ϕ and ψ' is

a disjunct of $DNF(\psi)$.

For any influence proposition A **possibly changes** F **if** ψ, take the family of LP-simple influence propositions A **possibly changes** F **if** ψ' such that ψ' is a disjunct of $DNF(\psi)$.

Finally, for each executability proposition **impossible** A **if** ψ, take the family of LP-simple executability propositions **impossible** A **if** ψ' such that ψ' is a disjunct of $DNF(\psi)$.

Let *LP-Simple* be a function that takes every vivid domain description to an LP-simple domain description that can be obtained by transforming each of its propositions in the manner described above.

Theorem 3.16 (Vivid Domains Theorem) *Let D be a vivid $\mathcal{AC}$ domain description. The domain descriptions D and LP-Simple(D) have the same models. Moreover, D is qualification-free if and only if LP-Simple(D) is.*

Since we have already specified a correct embedding of LP-simple, qualification-free domain descriptions into logic programming, the Vivid Domains Theorem establishes the more general fact that every vivid, qualification-free domain description can be correctly embedded in logic programming, by first transforming it into an equivalent LP-simple, qualification-free domain description. For instance, the logic program in the first example in Section 3.2 can be obtained in this manner.

Finally, it is clear from the previous discussion that any value proposition V can be transformed into a family Q of LP-simple value propositions such that V is true in a structure Ψ if and only if every member of Q is. Thus we have shown that the LP Correspondence Theorem can be applied to any value proposition, for any vivid qualification-free $\mathcal{AC}$ domain description. In this way we obtain correct formalizations of action domains that include non-atomic value propositions, nondeterministic actions, causal background information and action preconditions.

Chapter 4

Proofs for Preceding Chapter

We begin with the statement of the Splitting Set and Splitting Sequence Theorems [Tur96b], and their proofs.

We then prove the Correspondence Theorem and Reachability Corollary, showing that the translations from $\mathcal{AC}$ into default logic are sound and complete.

On the basis of these results, we go on to prove the LP Correspondence Theorem and LP Reachability Corollary, showing the correctness of our translations of LP-simple, qualification-free $\mathcal{AC}$ domain descriptions into logic programming. Finally we prove the Vivid Domains Theorem, which shows that every vivid $\mathcal{AC}$ domain description can be transformed into an equivalent LP-simple domain description.

4.1 Splitting a Default Theory

In this section we briefly turn our attention from the specific problem of representing actions in default logic and logic programming, in order to present technical results concerning default theories in general. These results—the Splitting Set Theorem and Splitting Sequence Theorem for default logic from [Tur96b]—are needed for the

proof of the Correspondence Theorem.

The Splitting Theorems for default logic can sometimes be used to simplify the task of reasoning about a default theory, by "splitting it into parts." These Splitting Theorems are related somewhat, in spirit, to "partial evaluation" in logic programming, in which results obtained from one part of a program are used to simplify the remainder of the program.[1] In fact, the Splitting Theorems for default logic closely resemble the Splitting Theorems for logic programming introduced in [LT94], despite complications due to the presence of arbitrary formulas in default theories.[2] Similar results for autoepistemic logic can be found in [GP92]. Very closely related, but independently obtained, results for default logic can be found in [Cho94, Cho95]. The relationship of that work to the splitting theorems presented here is examined in [Ant97].

4.1.1 Splitting Sets

Let D be a default theory over $\mathcal{L}(U)$ such that, for every rule $r \in D$, $pre(r)$ is in conjunctive normal form. (Of course any default theory can be easily transformed into an equivalent default theory, over the same language, satisfying this condition.) For any rule $r \in D$, a formula ϕ is a *constituent of* r if at least one of the following conditions holds: (i) ϕ is a conjunct of $pre(r)$; (ii) $\phi \in just(r)$; (iii) $\phi = cons(r)$.

A *splitting set* for D is a subset A of U such that for every rule $r \in D$ the following two conditions hold.

1. Every constituent of r belongs to $\mathcal{L}(A) \cup \mathcal{L}(U \setminus A)$.

2. If $cons(r)$ does not belong to $\mathcal{L}(U \setminus A)$, then r is a default rule over $\mathcal{L}(A)$.

[1] See, for example, [Kom90].

[2] In [LT94] we presented without proof Splitting Theorems for logic programs with classical negation and disjunction, under the answer set semantics [GL91]. The results for nondisjunctive logic programs follow from the Splitting Theorems for default logic. The definitions and proofs presented here can be adapted to the more general case of disjunctive default logic [GLPT91], from which the Splitting Theorems for disjunctive logic programs would follow as well.

If A is a splitting set for D, we say that A *splits* D. The *base of D relative to A*, denoted by $b_A(D)$, is the default theory over $\mathcal{L}(A)$ that consists of all members of D that are default rules over $\mathcal{L}(A)$.

Let $U_2 = \{a, b, c, d\}$. Consider the following default theory D_2 over $\mathcal{L}(U_2)$.

$$\frac{:\neg b}{a} \quad \frac{:\neg a}{b} \quad \frac{a \vee b : a, b}{c \vee d} \quad \frac{a \wedge (c \vee d) : \neg d}{\neg d} \quad \frac{b \wedge (c \vee d) : \neg c}{\neg c}$$

Take $A_2 = \{a, b\}$. It's easy to verify that A_2 splits D_2, with

$$b_{A_2}(D_2) = \left\{ \frac{:\neg b}{a}, \frac{:\neg a}{b} \right\}.$$

Notice that the default theory $b_{A_2}(D_2)$ over $\mathcal{L}(A_2)$ has two consistent extensions:

$$Cn_{A_2}(\{a\}) \quad \text{and} \quad Cn_{A_2}(\{b\}).$$

Given a splitting set A for D, and a set X of formulas from $\mathcal{L}(A)$, the *partial evaluation of D by X with respect to A*, denoted by $e_A(D, X)$, is the default theory over $\mathcal{L}(U \setminus A)$ obtained from D in the following manner. For each rule $r \in D \setminus b_A(D)$ such that

1. every conjunct of $pre(r)$ that belongs to $\mathcal{L}(A)$ also belongs to $Cn_A(X)$, and
2. no member of $just(r)$ has its complement in $Cn_A(X)$

there is a rule $r' \in e_A(D, X)$ such that

1. $pre(r')$ is obtained from $pre(r)$ by replacing each conjunct of $pre(r)$ that belongs to $\mathcal{L}(A)$ by $True$, and
2. $just(r') = just(r) \cap \mathcal{L}(U \setminus A)$, and
3. $cons(r') = cons(r)$.

For example, it is easy to verify that

$$e_{A_2}(D_2, Cn_{A_2}(\{a\})) = \left\{ \frac{True}{c \vee d}, \frac{True \wedge (c \vee d) : \neg d}{\neg d} \right\}$$

and that

$$e_{A_2}(D_2, Cn_{A_2}(\{b\})) = \left\{ \frac{True}{c \vee d}, \frac{True \wedge (c \vee d) : \neg c}{\neg c} \right\}.$$

Let A be a splitting set for D. A *solution to D with respect to A* is a pair $\langle X, Y \rangle$ of sets of formulas satisfying the following two properties.

1. X is a consistent extension of the default theory $b_A(D)$ over $\mathcal{L}(A)$.
2. Y is a consistent extension of the default theory $e_A(D, X)$ over $\mathcal{L}(U \setminus A)$.

For example, given our previous observations, it is easy to verify that D_2 has two solutions with respect to A_2:

$$\langle\, Cn_{A_2}(\{a\}),\ Cn_{U_2 \setminus A_2}(\{c, \neg d\}) \,\rangle \quad \text{and} \quad \langle\, Cn_{A_2}(\{b\}),\ Cn_{U_2 \setminus A_2}(\{\neg c, d\}) \,\rangle.$$

Theorem 4.1 (Splitting Set Theorem) *Let A be a splitting set for a default theory D over $\mathcal{L}(U)$. A set E of formulas is a consistent extension of D if and only if $E = Cn_U(X \cup Y)$ for some solution $\langle X, Y \rangle$ to D with respect to A.*

Thus, for example, it follows from the Splitting Set Theorem that the default theory D_2 has exactly two consistent extensions:

$$Cn_{U_2}(\{a, c, \neg d\}) \quad \text{and} \quad Cn_{U_2}(\{b, \neg c, d\}).$$

Corollary 4.2 (Splitting Set Corollary) *Let A be a splitting set for a default theory D over $\mathcal{L}(U)$. If E is a consistent extension of D, then the pair*

$$\langle\, E \cap \mathcal{L}(A),\ E \cap \mathcal{L}(U \setminus A) \,\rangle$$

is a solution to D with respect to A.

4.1.2 Splitting Sequences

A (transfinite) *sequence* is a family whose index set is an initial segment of ordinals $\{\alpha : \alpha < \mu\}$. We say that a sequence $\langle A_\alpha \rangle_{\alpha<\mu}$ of sets is *monotone* if $A_\alpha \subseteq A_\beta$ whenever $\alpha < \beta$, and *continuous* if, for each limit ordinal $\alpha < \mu$, $A_\alpha = \bigcup_{\gamma<\alpha} A_\gamma$.

A *splitting sequence* for a default theory D over $\mathcal{L}(U)$ is a nonempty, monotone, continuous sequence $\langle A_\alpha \rangle_{\alpha<\mu}$ of splitting sets for D such that $\bigcup_{\alpha<\mu} A_\alpha = U$.

The definition of a solution with respect to a splitting set is extended to splitting sequences as follows. Let $A = \langle A_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence for D. A *solution to D with respect to A* is a sequence $\langle E_\alpha \rangle_{\alpha<\mu}$ of sets of formulas that satisfies the following three conditions.

1. E_0 is a consistent extension of the default theory $b_{A_0}(D)$ over $\mathcal{L}(A_0)$.
2. For any α such that $\alpha + 1 < \mu$, $E_{\alpha+1}$ is a consistent extension of the default theory
$$e_{A_\alpha}\left(b_{A_{\alpha+1}}(D), \bigcup_{\gamma\leq\alpha} E_\gamma\right)$$
over $\mathcal{L}(A_{\alpha+1} \setminus A_\alpha)$.
3. For any limit ordinal $\alpha < \mu$, $E_\alpha = Cn_\emptyset(\emptyset)$.

We generalize the Splitting Set Theorem as follows.

Theorem 4.3 (Splitting Sequence Theorem) *Let $A = \langle A_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence for a default theory D over $\mathcal{L}(U)$. A set E of formulas is a consistent extension of D if and only if*

$$E = Cn_U\left(\bigcup_{\alpha<\mu} E_\alpha\right)$$

for some solution $\langle E_\alpha \rangle_{\alpha<\mu}$ to D with respect to A.

The proof of this theorem relies on the Splitting Set Theorem. We also have the following counterpart to the Splitting Set Corollary.

Corollary 4.4 (Splitting Sequence Corollary) *Let $A = \langle A_\alpha \rangle_{\alpha<\mu}$ be a splitting sequence for a default theory D over $\mathcal{L}(U)$. Let $\langle U_\alpha \rangle_{\alpha<\mu}$ be the sequence of pairwise disjoint subsets of U such that for all $\alpha < \mu$*

$$U_\alpha = A_\alpha \setminus \bigcup_{\gamma<\alpha} A_\gamma\,.$$

If E is a consistent extension of D, then the sequence $\langle E \cap \mathcal{L}(U_\alpha) \rangle_{\alpha<\mu}$ is a solution to D with respect to A.

4.2 Proof of Splitting Set Theorem

The proof begins with three auxiliary lemmas.

Lemma 4.5 *Let U, U' be disjoint sets of atoms. Let R be a set of inference rules over $\mathcal{L}(U)$. Let R' be a set of inference rules over $\mathcal{L}(U')$. Let $X = Cn_{U\cup U'}(R \cup R')$.*

- *If X is consistent, then $X \cap \mathcal{L}(U) = Cn_U(R)$.*
- *$X = Cn_{U\cup U'}(Cn_U(R) \cup R')$.*

Proof. Straightforward. □

Lemma 4.6 *Let U, U' be disjoint sets of atoms. Let D be a default theory over $\mathcal{L}(U)$, and let D' be a default theory over $\mathcal{L}(U')$. Let E be a consistent, logically closed set of formulas from $\mathcal{L}(U)$ and let E' be a consistent, logically closed set of formulas from $\mathcal{L}(U')$. Let $X = Cn_{U\cup U'}(E \cup E')$. $X = Cn_{U\cup U'}((D \cup D')^X)$ if and only if $E = Cn_U(D^E)$ and $E' = Cn_{U'}((D')^{E'})$.*

Proof. By Lemma 4.5, we have $X \cap \mathcal{L}(U) = Cn_U(E)$ and $X \cap \mathcal{L}(U') = Cn_{U'}(E')$. And because E and E' are logically closed, we have

$$X \cap \mathcal{L}(U) = E \quad \text{and} \quad X \cap \mathcal{L}(U') = E' . \tag{4.1}$$

The fact that $(D \cup D')^X = D^{X \cap \mathcal{L}(U)} \cup (D')^{X \cap \mathcal{L}(U')}$ is easily checked. Hence by (4.1) we have

$$(D \cup D')^X \quad = \quad D^E \cup (D')^{E'} . \tag{4.2}$$

($\Rightarrow$) Assume that $X = Cn_{U \cup U'}((D \cup D')^X)$. It follows by (4.2) that $X = Cn_{U \cup U'}(D^E \cup (D')^{E'})$. By Lemma 4.5 and (4.1), we have $E = Cn_U(D^E)$ and $E' = Cn_{U'}((D')^{E'})$.

($\Leftarrow$) Assume $E = Cn_U(D^E)$ and $E' = Cn_{U'}((D')^{E'})$. Thus we have

$$X = Cn_{U \cup U'}(Cn_U(D^E) \cup Cn_{U'}((D')^{E'})) .$$

By Lemma 4.5 we conclude that $X = Cn_{U \cup U'}(D^E \cup (D')^{E'})$. By (4.2) we have $X = Cn_{U \cup U'}((D \cup D')^X)$. □

Lemma 4.7 *Let D be a default theory over $\mathcal{L}(U)$, and let E be a set of formulas from $\mathcal{L}(U)$. Let D' be a default theory over $\mathcal{L}(U)$ such that every rule $r \in D'$ satisfies at least one of the following conditions: (i) $cons(r)$ is equivalent to True; (ii) $pre(r) \notin E$; (iii) some member of $just(r)$ has its complement in E. E is an extension of D if and only if E is an extension of $D \cup D'$.*

Proof. Straightforward. □

In proving the Splitting Set Theorem it is convenient to introduce a set of alternative definitions, differing very slightly from those used in stating the theorem. (We nonetheless prefer the original definitions, because they are more convenient in applications of the Splitting Theorems.)

Let D be a default theory over $\mathcal{L}(U)$ split by A. We define the following.

- $t^*_A(D) = \{\ r \in D\ :\ cons(r) \in \mathcal{L}(U \setminus A)\ \}$
- $b^*_A(D) = D \setminus t^*_A(D)$

The advantage of these alternative definitions is captured in the following key lemma, which fails to hold for their counterparts $b_A(D)$ and $D \setminus b_A(D)$.

Lemma 4.8 *Let D be a default theory over $\mathcal{L}(U)$ with splitting set A. For any set X of formulas from $\mathcal{L}(U)$, $b^*_A(D)^X = b^*_A(D^X)$ and $t^*_A(D)^X = t^*_A(D^X)$.*

Proof. It is enough to show that $t^*_A(D)^X = t^*_A(D^X)$.

($\Rightarrow$) Assume $r \in t^*_A(D)^X$. Clearly we have $r \in D^X$. We need to show that $cons(r) \in \mathcal{L}(U \setminus A)$. Since $r \in t^*_A(D)^X$, there must be an $r' \in t^*_A(D)$ such that $\{r'\}^X = \{r\}$. We know that $cons(r') \in \mathcal{L}(U \setminus A)$; and since $cons(r) = cons(r')$, we're done. Proof in the other direction is similar. □

The proof also makes use of the following additional alternative definitions.

Given a set X of formulas from $\mathcal{L}(A)$, let $e^*_A(D, X)$ be the default theory over $\mathcal{L}(U \setminus A)$ obtained from D in the following manner. For each rule $r \in t^*_A(D)$ such that

- every conjunct of $pre(r)$ that belongs to $\mathcal{L}(A)$ also belongs to $Cn_A(X)$, and
- no member of $just(r)$ has its complement in $Cn_A(X)$

there is a rule $r' \in e^*_A(D, X)$ such that

- $pre(r')$ is obtained from $pre(r)$ by replacing each conjunct of $pre(r)$ that belongs to $\mathcal{L}(A)$ by $True$, and
- $just(r') = just(r) \cap \mathcal{L}(U \setminus A)$, and
- $cons(r') = cons(r)$.

Notice that e^*_A differs from e_A only in starting with the rules in $t^*_A(D)$ instead of the rules in $D \setminus b_A(D)$.

Finally, let $s^*_A(D)$ be the set of all pairs $\langle X, Y\rangle$ such that

- X is a consistent extension of $b^*_A(D)$, and
- Y is a consistent extension of $e^*_A(D, X)$.

The following lemma shows that these alternative definitions are indeed suitable for our purpose.

Lemma 4.9 *If a default theory D over $\mathcal{L}(U)$ is split by A, then $s^*_A(D)$ is precisely the set of solutions to D with respect to A.*

Proof. ($\Leftarrow$) Assume that $\langle X, Y\rangle$ is a solution to D with respect to A. Thus X is a consistent extension of $b_A(D)$ and Y is a consistent extension of $e_A(D, X)$. Let r be a rule in $b_A(D) \setminus b^*_A(D)$. Notice that $b_A(D) \setminus b^*_A(D) \subseteq t^*_A(D)$. So $r \in b_A(D) \cap t^*_A(D)$. It follows that r is a rule over $\mathcal{L}(A)$ such that $cons(r) \in \mathcal{L}(U \setminus A)$. Thus, either $cons(r)$ is equivalent to *True* or $cons(r)$ is equivalent to *False*. Assume that $cons(r)$ is equivalent to *False*. Since $r \in b_A(D)$ and X is a consistent extension of $b_A(D)$, we can conclude that either $pre(r) \notin X$ or some member of $just(r)$ has its complement in X. So we've shown that for every rule $r \in b_A(D) \setminus b^*_A(D)$, either $cons(r)$ is equivalent to *True* or $pre(r) \notin X$ or some member of $just(r)$ has its complement in X. Since $b^*_A(D) \subseteq b_A(D)$, it follows by Lemma 4.7 that X is a consistent extension of $b^*_A(D)$. It remains to show that Y is a consistent extension of $e^*_A(D, X)$. The reasoning here is much the same. Since $D \setminus b_A(D) \subseteq t^*_A(D)$, we know that $e_A(D, X) \subseteq e^*_A(D, X)$. Furthermore, it is not difficult to show, by the definitions of e_A and e^*_A, along with some of the previous observations, that if a rule r belongs to $e^*_A(D, X) \setminus e_A(D, X)$, then $cons(r)$ is equivalent to *True*. From this it follows by Lemma 4.7 that Y is a consistent extension of $e^*_A(D, X)$.

Proof in the other direction is similar. □

The next three lemmas are used in the proof of Lemma 4.13, which is one of the main lemmas in the proof of the Splitting Set Theorem.

Lemma 4.10 *Let R be a set of inference rules over $\mathcal{L}(U)$ with splitting set A. Let E be a consistent set of formulas. If $E = Cn_U(R)$, then $E \cap \mathcal{L}(A) = Cn_A(b^*_A(R))$.*

Proof. Let r be a rule from $b^*_A(R)$. Since E is closed under R and r is a rule over $\mathcal{L}(A)$, we know that either $pre(r) \notin E \cap \mathcal{L}(A)$ or $cons(r) \in E \cap \mathcal{L}(A)$. This shows that $E \cap \mathcal{L}(A)$ is closed under $b^*_A(R)$. Since E is logically closed, so is $E \cap \mathcal{L}(A)$. It follows that $Cn_A(b^*_A(R)) \subseteq E \cap \mathcal{L}(A)$.

Let $E' = Cn_U[Cn_A(b^*_A(R)) \cup (E \cap \mathcal{L}(U \setminus A))]$. Notice that E' is a consistent, logically closed subset of E. Also notice that by Lemma 4.5 we can conclude that $E' \cap \mathcal{L}(U \setminus A) = E \cap \mathcal{L}(U \setminus A)$. Again by Lemma 4.5 we also know that $E' \cap \mathcal{L}(A) = Cn_A(b^*_A(R))$. So we will complete the proof by showing that $E' = E$.

Let r be a rule in $t^*_A(R)$. We know that either $pre(r) \notin E$ or $cons(r) \in E$. Since $E' \subseteq E$, we have $pre(r) \notin E'$ if $pre(r) \notin E$. On the other hand, since $E' \cap \mathcal{L}(U \setminus A) = E \cap \mathcal{L}(U \setminus A)$ and $cons(r) \in \mathcal{L}(U \setminus A)$, we have $cons(r) \in E'$ if $cons(r) \in E$. Thus, either $pre(r) \notin E'$ or $cons(r) \in E'$. That is, E' is closed under $t^*_A(R)$. Of course E' is also closed under $b^*_A(R)$. So we've shown that E' is a logically closed subset of E that is closed under R. And since E is the least logically closed set of formulas closed under R, $E' = E$. □

Lemma 4.11 *Let R be a set of inference rules over $\mathcal{L}(U)$ with splitting set A. Let E be a consistent set of formulas. If $E = Cn_U(R)$, then $E = Cn_U[(E \cap \mathcal{L}(A)) \cup t^*_A(R)]$.*

Proof. Let $E' = Cn_U[(E \cap \mathcal{L}(A)) \cup t^*_A(R)]$. We'll show that $E = E'$. By the previous lemma we can conclude that $E' \cap \mathcal{L}(A) = E \cap \mathcal{L}(A)$. Also by the previous lemma, we know that $E \cap \mathcal{L}(A) = Cn_A(b^*_A(R))$. It follows that $E' \cap \mathcal{L}(A)$ is closed

under $b^*_A(R)$, and since $b^*_A(R)$ is a set of rules over $\mathcal{L}(A)$, E' is closed under $b^*_A(R)$. Since E' is also closed under $t^*_A(R)$, E' is closed under R. Since E is the least logically closed set that is closed under R, $E \subseteq E'$. On the other hand, we can see that E is closed under $(E \cap \mathcal{L}(A)) \cup t^*_A(R)$. It follows that $E' \subseteq E$. So $E = E'$. □

Lemma 4.12 *Let R be a set of inference rules over $\mathcal{L}(U)$ with splitting set A. Let E be a consistent set of formulas. If $E \cap \mathcal{L}(A) = Cn_A(b^*_A(R))$ and $E = Cn_U[(E \cap \mathcal{L}(A)) \cup t^*_A(R)]$, then $E = Cn_U(R)$.*

Proof. Let $E' = Cn_U(R)$. It's easy to see that E is closed under R. Thus $E' \subseteq E$. Since E' is closed under $b^*_A(R)$ and $E \cap \mathcal{L}(A)$ is the least set closed under $b^*_A(R)$, we have $E \cap \mathcal{L}(A) \subseteq E'$. So E' is closed under $E \cap \mathcal{L}(A)$. Since E' is also closed under $t^*_A(R)$, E' is closed under $(E \cap \mathcal{L}(A)) \cup t^*_A(R)$. Since E' is a logically closed subset of E and E is the least logically closed set closed under $(E \cap \mathcal{L}(A)) \cup t^*_A(R)$, we have $E' = E$. □

Lemma 4.13 *Let D be a default theory over $\mathcal{L}(U)$ with splitting set A. Let E be a consistent set of formulas from $\mathcal{L}(U)$. $E = Cn_U(D^E)$ if and only if*

- $E \cap \mathcal{L}(A) = Cn_A[b^*_A(D)^{E \cap \mathcal{L}(A)}]$ *and*
- $E = Cn_U[(E \cap \mathcal{L}(A)) \cup t^*_A(D)^E]$.

Proof. ($\Rightarrow$) Assume that $E = Cn_U(D^E)$. Notice that A splits D^E. By Lemma 4.10, $E \cap \mathcal{L}(A) = Cn_A(b^*_A(D^E))$. By Lemma 4.8, $b^*_A(D^E) = b^*_A(D)^E$. Since $b^*_A(D)$ is a default theory over $\mathcal{L}(A)$, $b^*_A(D)^E = b^*_A(D)^{E \cap \mathcal{L}(A)}$. So $E \cap \mathcal{L}(A) = Cn_A(b^*_A(D)^{E \cap \mathcal{L}(A)})$. By Lemma 4.11, $E = Cn_U[(E \cap \mathcal{L}(A)) \cup t^*_A(D^E)]$. By Lemma 4.8, $t^*_A(D^E) = t^*_A(D)^E$. So $E = Cn_U[(E \cap \mathcal{L}(A)) \cup t^*_A(D)^E]$.

($\Leftarrow$) Assume that $E = Cn_U[(E \cap \mathcal{L}(A)) \cup t^*_A(D)^E]$. Recall that A splits D^E. Since $b^*_A(D)$ is a default theory over $\mathcal{L}(A)$, we have $b^*_A(D)^{E \cap \mathcal{L}(A)} = b^*_A(D)^E$. By Lemma 4.8, $t^*_A(D)^E = t^*_A(D^E)$ and $b^*_A(D)^E = b^*_A(D^E)$. Thus we have $E \cap \mathcal{L}(A) = Cn_A(b^*_A(D^E))$ and $E = Cn_U[(E \cap \mathcal{L}(A)) \cup t^*_A(D^E)]$. By Lemma 4.12, $E = Cn_U(D^E)$. □

The next three lemmas are used in the proof of Lemma 4.17, which is another of the main lemmas in the proof of the Splitting Set Theorem.

Lemma 4.14 *Let R,R' be sets of inference rules over $\mathcal{L}(U)$. If $Cn_U(R)$ is closed under R' and $Cn_U(R')$ is closed under R, then $Cn_U(R) = Cn_U(R')$.*

Proof. Straightforward. □

Lemma 4.15 *Let D be a default theory over $\mathcal{L}(U)$ split by A. Let E be a logically closed set of formulas from $\mathcal{L}(U)$, with $X = E \cap \mathcal{L}(A)$ and $Y = E \cap \mathcal{L}(U \setminus A)$. The set $Cn_U[X \cup e^*_A(D, X)^Y]$ is closed under $X \cup t^*_A(D)^E$.*

Proof. Let $E' = Cn_U[X \cup e^*_A(D, X)^Y]$. Of course E' is closed under X. We must show that E' is also closed under $t^*_A(D)^E$. Let r be a rule from $t^*_A(D)^E$ such that $pre(r) \in E'$. We must show that $cons(r) \in E'$. Let r' be a rule in $t^*_A(D)$ such that $\{r'\}^E = \{r\}$. It follows that no member of $just(r')$ has its complement in E. Hence no member of $just(r')$ has its complement in X. Notice also that $pre(r') = pre(r)$. Since $pre(r) \in E'$ and E' is logically closed, we can conclude that every conjunct of $pre(r')$ that belongs to $\mathcal{L}(A)$ also belongs to X. Given these observations, we know there is a rule r'_e such that $e^*_A(\{r'\}, X) = \{r'_e\}$. Notice that $just(r'_e) = just(r') \cap \mathcal{L}(U \setminus A)$. Since no member of $just(r')$ has its complement in E, it's clear that no member of $just(r'_e)$ has its complement in Y. Given this observation we know that there is a rule r_e such that $\{r'_e\}^Y = \{r_e\}$. Since $pre(r_e) = pre(r'_e)$, we know that $pre(r_e)$ is the result of replacing each conjunct of $pre(r')$ that belongs to $\mathcal{L}(A)$ by $True$. And since $pre(r') = pre(r)$, we can conclude that $pre(r_e) \in E'$. It follows that $cons(r_e) \in E'$. And since $cons(r_e) = cons(r'_e) = cons(r') = cons(r)$, we have $cons(r) \in E'$. □

Lemma 4.16 *Let D be a default theory over $\mathcal{L}(U)$ split by A. Let E be a logically closed set of formulas from $\mathcal{L}(U)$, with $X = E \cap \mathcal{L}(A)$ and $Y = E \cap \mathcal{L}(U \setminus A)$. The set $Cn_U(X \cup t^*_A(D)^E)$ is closed under $X \cup e^*_A(D,X)^Y$.*

Proof. Let $E' = Cn_U(X \cup t^*_A(D)^E)$. Of course E' is closed under X. We must show that E' is also closed under $e^*_A(D,X)^Y$. Let r_e be a rule from $e^*_A(D,X)^Y$ such that $pre(r_e) \in E'$. We must show that $cons(r_e) \in E'$. Let r' be a rule in $t^*_A(D)$ such that $e^*_A(\{r'\},X)^Y = \{r_e\}$. Notice that no member of $just(r')$ has its complement in X. Furthermore, no member of $just(r') \cap \mathcal{L}(U \setminus A)$ has its complement in Y. We can conclude that no member of $just(r')$ has its complement in E. Given this observation we know that there is a rule r in $t^*_A(D)^E$ such that $\{r'\}^E = \{r\}$. Notice that $pre(r) = pre(r')$. We know that every conjunct of $pre(r')$ that belongs to $\mathcal{L}(A)$ also belongs to X, which is a subset of E'. Moreover, since $pre(r_e) \in E'$, we know that every conjunct of $pre(r')$ that does not belong to $\mathcal{L}(A)$ belongs to E'. It follows that $pre(r') \in E'$. Thus $pre(r) \in E'$, from which it follows that $cons(r) \in E'$. And since $cons(r) = cons(r') = cons(r_e)$, $cons(r_e) \in E'$. □

Lemma 4.17 *Let D be a default theory over $\mathcal{L}(U)$ split by A. Let E be a logically closed set of formulas from $\mathcal{L}(U)$, with $X = E \cap \mathcal{L}(A)$ and $Y = E \cap \mathcal{L}(U \setminus A)$. We have $Cn_U[X \cup e^*_A(D,X)^Y] = Cn_U[X \cup t^*_A(D)^E]$.*

Proof. Immediate from the previous three lemmas. □

Proof of Splitting Set Theorem. Given a default theory D over $\mathcal{L}(U)$ with splitting set A, we know by Lemma 4.9 that $s^*_A(D)$ is precisely the set of solutions to D with respect to A. We will show that E is a consistent extension of D if and only if $E = Cn_U(X \cup Y)$ for some $\langle X, Y\rangle \in s^*_A(D)$.

($\Rightarrow$) Assume E is a consistent extension of D. Let $X = E \cap \mathcal{L}(A)$ and $Y = E \cap \mathcal{L}(U \setminus A)$. By Lemma 4.13, $X = Cn_A(b^*_A(D)^X)$ and $E = Cn_U(X \cup t^*_A(D)^E)$. By Lemma 4.17, $E = Cn_U(X \cup e^*_A(D,X)^Y)$. By Lemma 4.5, we can conclude that

$Y = Cn_{U\setminus A}(e^*_A(D,X)^Y)$. So we have established that $\langle X, Y\rangle \in s^*_A(D)$. We can also conclude by Lemma 4.5 that $E = Cn_U[X \cup Cn_{U\setminus A}(e^*_A(D,X)^Y)]$. And since $Y = Cn_{U\setminus A}(e^*_A(D,X)^Y)$, we have $E = Cn_U(X \cup Y)$.

($\Leftarrow$) Assume $E = Cn_U(X \cup Y)$ for some $\langle X, Y\rangle \in s^*_A(D)$. Since $\langle X, Y\rangle \in s^*_A(D)$, we have $Y = Cn_{U\setminus A}(e^*_A(D,X)^Y)$. Hence $E = Cn_U[X \cup Cn_{U\setminus A}(e^*_A(D,X)^Y)]$. By Lemma 4.5, we can conclude that $E = Cn_U(X \cup e^*_A(D,X)^Y)$. Thus, by Lemma 4.17, $E = Cn_U(X \cup t^*_A(D)^E)$. By Lemma 4.5, $E \cap \mathcal{L}(A) = Cn_A(X)$, and since X is logically closed, $Cn_A(X) = X$. So $E \cap \mathcal{L}(A) = X$. Since $\langle X, Y\rangle \in s^*_A(D)$, we have $X = Cn_A(b^*_A(D)^X)$. It follows by Lemma 4.13 that $E = Cn_U(D^E)$. □

Proof of Splitting Set Corollary. Assume that E is a consistent extension of D. By the Splitting Set Theorem, there is a solution $\langle X, Y\rangle$ to D with respect to A such that $E = Cn_U(X \cup Y)$. Since $X \subseteq \mathcal{L}(A)$ and $Y \subseteq \mathcal{L}(U \setminus A)$, we can conclude by Lemma 4.5 that $E \cap \mathcal{L}(A) = Cn_A(X)$. And since X is logically closed, $Cn_A(X) = X$. So $E \cap \mathcal{L}(A) = X$. A symmetric argument shows that $E \cap \mathcal{L}(U \setminus A) = Y$. □

4.3 Proof of Splitting Sequence Theorem

Lemma 4.18 *Let D be a default theory over $\mathcal{L}(U)$ with splitting sequence $A = \langle A_\alpha\rangle_{\alpha<\mu}$. Let E be a set of formulas from $\mathcal{L}(U)$. Let $X = \langle X_\alpha\rangle_{\alpha<\mu}$ be a sequence of sets of formulas from $\mathcal{L}(U)$ such that*

- *$X_0 = E \cap \mathcal{L}(A_0)$,*
- *for all α s.t. $\alpha + 1 < \mu$, $X_{\alpha+1} = E \cap \mathcal{L}(A_{\alpha+1} \setminus A_\alpha)$,*
- *for any limit ordinal $\alpha < \mu$, $X_\alpha = Cn_\emptyset(\emptyset)$.*

If E is a consistent extension of D, then X is a solution to D with respect to A.

Proof. There are three things to check.

First, by the Splitting Set Corollary, we can conclude that $E \cap \mathcal{L}(A_0)$ is a consistent extension of $b_{A_0}(D)$.

Second, choose α such that $\alpha + 1 < \mu$. We must show that $X_{\alpha+1}$ is a consistent extension of

$$e_{A_\alpha}\left(b_{A_{\alpha+1}}(D), \bigcup_{\gamma \leq \alpha} X_\gamma\right). \tag{4.3}$$

Let $\beta = \alpha + 1$. By the Splitting Set Corollary, $E \cap \mathcal{L}(A_\beta)$ is a consistent extension of $b_{A_\beta}(D)$. Let $D' = b_{A_\beta}(D)$ and let $E' = E \cap \mathcal{L}(A_\beta)$. By the Splitting Set Corollary, since A_α splits D', $E' \cap \mathcal{L}(A_\beta \setminus A_\alpha)$ is a consistent extension of $e_{A_\alpha}(D', E' \cap \mathcal{L}(A_\alpha))$. It is easy to verify that $X_{\alpha+1} = E' \cap \mathcal{L}(A_\beta \setminus A_\alpha)$. It is not difficult to verify also that $e_{A_\alpha}(D', E' \cap \mathcal{L}(A_\alpha))$ is the same as (4.3).

Third, for any limit ordinal $\alpha < \mu$, $X_\alpha = Cn_\emptyset(\emptyset)$. □

Lemma 4.19 *Let D be a default theory over $\mathcal{L}(U)$ with splitting sequence $A = \langle A_\alpha \rangle_{\alpha<\mu}$. Let $\langle E_\alpha \rangle_{\alpha<\mu}$ be a solution to D with respect to A. For all $\alpha < \mu$*

$$Cn_{A_\alpha}\left(\bigcup_{\gamma \leq \alpha} E_\gamma\right)$$

is a consistent extension of $b_{A_\alpha}(D)$.

Proof. For all $\alpha < \mu$, let

$$X_\alpha = Cn_{A_\alpha}\left(\bigcup_{\gamma \leq \alpha} E_\gamma\right).$$

Proof is by induction on α. Assume that for all $\gamma < \alpha$, X_γ is a consistent extension of $b_{A_\gamma}(D)$. We'll show that X_α is a consistent extension of $b_{A_\alpha}(D)$. There are two cases to consider.

Case 1: α is not a limit ordinal. Choose γ such that $\gamma + 1 = \alpha$. By the inductive hypothesis, X_γ is a consistent extension of $b_{A_\gamma}(D)$. We also know that E_α is a consistent extension of $e_{A_\gamma}(b_{A_\alpha}(D), \bigcup_{\beta \leq \gamma} E_\beta)$. Let $D' = b_{A_\alpha}(D)$. It is clear

that $b_{A_\gamma}(D) = b_{A_\gamma}(D')$. It is not difficult to verify that $e_{A_\gamma}(b_{A_\alpha}(D), \bigcup_{\beta \leq \gamma} E_\beta)$ is the same as $e_{A_\gamma}(D', X_\gamma)$. So we've shown that X_γ is a consistent extension of $b_{A_\gamma}(D')$ and that E_α is a consistent extension of $e_{A_\gamma}(D', X_\gamma)$. By the Splitting Set Theorem, it follows that $Cn_{A_\alpha}(X_\gamma \cup E_\alpha)$ is a consistent extension of D'. And since it's easy to check that $Cn_{A_\alpha}(X_\gamma \cup E_\alpha) = X_\alpha$, we're done with the first case.

Case 2: α is a limit ordinal. First we show that X_α is closed under $b_{A_\alpha}(D)^{X_\alpha}$. So suppose the contrary. Thus there is an $r \in b_{A_\alpha}(D)^{X_\alpha}$ such that $pre(r) \in X_\alpha$ and $cons(r) \notin X_\alpha$. Since A is continuous and α is a limit ordinal, we know there must be a $\gamma < \alpha$ such that $r \in b_{A_\gamma}(D)^{X_\alpha}$. Since $b_{A_\gamma}(D)$ is a default theory over $\mathcal{L}(A_\gamma)$, we have $b_{A_\gamma}(D)^{X_\alpha} = b_{A_\gamma}(D)^{X_\gamma}$. So $r \in b_{A_\gamma}(D)^{X_\gamma}$. Furthermore, it follows that $pre(r) \in X_\gamma$ and $cons(r) \notin X_\gamma$. This shows that X_γ is not closed under $b_{A_\gamma}(D)^{X_\gamma}$, which contradicts the fact that, by the inductive hypothesis, X_γ is a consistent extension of $b_{A_\gamma}(D)$. So we have shown that X_α is closed under $b_{A_\alpha}(D)^{X_\alpha}$.

Now, let $E = Cn_{A_\alpha}(b_{A_\alpha}(D)^{X_\alpha})$. We will show that $E = X_\alpha$, from which it follows that X_α is a consistent extension of $b_{A_\alpha}(D)$. Since X_α is logically closed and closed under $b_{A_\alpha}(D)^{X_\alpha}$, we know that $E \subseteq X_\alpha$. Suppose $E \neq X_\alpha$, and consider any formula $\phi \in X_\alpha \setminus E$. Since A is continuous and α is a limit ordinal, there must be a $\gamma < \alpha$ such that ϕ is from $\mathcal{L}(A_\gamma)$ and therefore $\phi \in X_\gamma$. Thus, X_γ is a proper superset of $E \cap \mathcal{L}(A_\gamma)$. By the inductive hypothesis, we know that X_γ is a consistent extension of $b_{A_\gamma}(D)$. Thus, $X_\gamma = Cn_{A_\gamma}(b_{A_\gamma}(D)^{X_\gamma})$. And since $b_{A_\gamma}(D)^{X_\gamma} = b_{A_\gamma}(D)^{X_\alpha}$, we have $X_\gamma = Cn_{A_\gamma}(b_{A_\gamma}(D)^{X_\alpha})$. Since $E = Cn_{A_\alpha}(b_{A_\alpha}(D)^{X_\alpha})$ and $b_{A_\gamma}(D)^{X_\alpha} \subseteq b_{A_\alpha}(D)^{X_\alpha}$, we know that E is closed under $b_{A_\gamma}(D)^{X_\alpha}$. Moreover, since $b_{A_\gamma}(D)^{X_\alpha}$ is a default theory over $\mathcal{L}(A_\gamma)$, $E \cap \mathcal{L}(A_\gamma)$ is closed under $b_{A_\gamma}(D)^{X_\alpha}$. But X_γ is the least logically closed set closed under $b_{A_\gamma}(D)^{X_\alpha}$, so $X_\gamma \subseteq E \cap \mathcal{L}(A_\gamma)$, which contradicts the fact that X_γ is a proper superset of $E \cap \mathcal{L}(A_\gamma)$. We can conclude that $E = X_\alpha$, which completes the second case. □

Let D be a default theory over $\mathcal{L}(U)$ with splitting sequence $A = \langle A_\alpha \rangle_{\alpha<\mu}$. The *standard extension* of A is the sequence $B = \langle B_\alpha \rangle_{\alpha<\mu+1}$ such that

- for all $\alpha < \mu$, $B_\alpha = A_\alpha$, and
- $B_\mu = U$.

Notice that the standard extension of A is itself a splitting sequence for D.

Lemma 4.20 *Let D be a default theory over $\mathcal{L}(U)$ with splitting sequence $A = \langle A_\alpha \rangle_{\alpha<\mu}$. Let $B = \langle B_\alpha \rangle_{\alpha<\mu+1}$ be the standard extension of A. Let $X = \langle X_\alpha \rangle_{\alpha<\mu}$ be a sequence of sets of formulas from $\mathcal{L}(U)$. Let $Y = \langle Y_\alpha \rangle_{\alpha<\mu+1}$ be defined as follows.*

- *For all $\alpha < \mu$, $Y_\alpha = X_\alpha$.*
- *$Y_\mu = Cn_\emptyset(\emptyset)$.*

If X is a solution to D with respect to A, then Y is a solution to D with respect to B.

Proof. First, it's clear that Y_0 is a consistent extension of $b_{B_0}(D)$, since $Y_0 = X_0$, $b_{B_0}(D) = b_{A_0}(D)$, and X_0 is a consistent extension of $b_{A_0}(D)$. Similarly, it's clear that for any α such that $\alpha + 1 < \mu$, $Y_{\alpha+1}$ is a consistent extension of

$$e_{B_\alpha}\left(b_{B_{\alpha+1}}(D), \bigcup_{\gamma\leq\alpha} Y_\gamma\right).$$

We also know that for any limit ordinal $\alpha < \mu$, $Y_\alpha = Cn_\emptyset(\emptyset)$. It remains to show that we handle μ correctly. There are two cases to consider.

Case 1: μ is a limit ordinal. In this case we must show that $Y_\mu = Cn_\emptyset(\emptyset)$, which it does.

Case 2: μ is not a limit ordinal. In this case, choose α such that $\alpha + 1 = \mu$. We must show that Y_μ is a consistent extension of the default theory

$$e_{B_\alpha}\left(b_{B_\mu}(D), \bigcup_{\gamma\leq\alpha} Y_\gamma\right) \tag{4.4}$$

over $\mathcal{L}(B_\mu \setminus B_\alpha)$. Since A is a splitting sequence for a default theory over $\mathcal{L}(U)$, we know that $\bigcup_{\gamma<\mu} A_\gamma = U$. Moreover, since A is monotone and μ is not a limit ordinal, it follows that $A_\alpha = U$. And since $B_\alpha = A_\alpha$, we know that $b_{B_\alpha}(D) = D$. It follows that default theory (4.4) is empty. It also follows that $B_\mu \setminus B_\alpha = \emptyset$, so the language of (4.4) is $\mathcal{L}(\emptyset)$. Since $Y_\mu = Cn_\emptyset(\emptyset)$, we have shown that Y_μ is a consistent extension of (4.4). □

Proof of Splitting Sequence Theorem. ($\Rightarrow$) Assume that E is a consistent extension of D. By Lemma 4.18, there is a solution $\langle E_\alpha \rangle_{\alpha<\mu}$ to D with respect to $\langle A_\alpha \rangle_{\alpha<\mu}$ for which it is not difficult to verify that

$$E = Cn_U\left(\bigcup_{\alpha<\mu} E_\alpha\right).$$

($\Leftarrow$) Assume that $X = \langle X_\alpha \rangle_{\alpha<\mu}$ is a solution to D with respect to $\langle A_\alpha \rangle_{\alpha<\mu}$. Let

$$E = Cn_U\left(\bigcup_{\alpha<\mu} X_\alpha\right).$$

Let $B = \langle B_\alpha \rangle_{\alpha<\mu+1}$ be the standard extension of $\langle A_\alpha \rangle_{\alpha<\mu}$. By Lemma 4.20, we know there is a solution $\langle Y_\alpha \rangle_{\alpha<\mu+1}$ to D with respect to B such that

$$E = Cn_U\left(\bigcup_{\alpha<\mu+1} Y_\alpha\right).$$

Moreover, we know there is an $\alpha < \mu + 1$ such that $B_\alpha = U$. Thus $b_{B_\alpha}(D) = D$ and

$$E = Cn_{B_\alpha}\left(\bigcup_{\gamma\leq\alpha} Y_\gamma\right).$$

It follows by Lemma 4.19 that E is a consistent extension of D. □

Proof of Splitting Sequence Corollary. Assume that E is a consistent extension of D. By the Splitting Sequence Theorem, there is a solution $\langle X_\alpha \rangle_{\alpha<\mu}$ to D with respect to A such that $E = Cn_U\left(\bigcup_{\alpha<\mu} X_\alpha\right)$. We will show that for all $\alpha < \mu$,

$E \cap \mathcal{L}(U_\alpha) = X_\alpha$. Let $X = \bigcup_{\alpha<\mu} X_\alpha$. Consider any $\alpha < \mu$. We have $X_\alpha \subseteq \mathcal{L}(U_\alpha)$, $X \setminus X_\alpha \subseteq \mathcal{L}(U \setminus U_\alpha)$, and $E = Cn_U(X_\alpha \cup X \setminus X_\alpha)$. Thus, by Lemma 4.5 we can conclude that $E \cap \mathcal{L}(U_\alpha) = Cn_{U_\alpha}(X_\alpha)$. And since X_α is logically closed, we have $Cn_{U_\alpha}(X_\alpha) = X_\alpha$. □

4.4 Proof of Correspondence Theorem and Reachability Corollary

Our primary task is to prove the special case of the Correspondence Theorem in which the domain description has no value propositions. We'll call this intermediate result the Correspondence Lemma. Most of the work in this section is devoted to its proof.

Let D be a qualification-free domain description without value propositions, with fluents $\mathbf{F}$, and frame fluents $\mathbf{F}_f$. We will show that there is a one-to-one correspondence between models of D and consistent extensions of $\delta(D)$ such that a value proposition V is true in a model of D if and only if the formula $[V]$ belongs to the corresponding extension of $\delta(D)$.

We begin with a fundamental lemma, used to show that our default theory $\delta(D)$ correctly characterizes the possible initial situations.

Let Δ_0 be the default theory

$$R \cup \left\{ \frac{: L}{L} : L \textit{ is a fluent literal} \right\}.$$

Notice that Δ_0 is default theory over $\mathcal{L}(\mathbf{F})$.

Lemma 4.21 *A consistent set X of fluent formulas is an extension of Δ_0 if and only if there is a state S such that $X = Cn_{\mathbf{F}}(S)$.*

Proof. Recall that an interpretation S is a state if and only if $Cn_{\mathbf{F}}(S)$ is closed under R. (Recall also that interpretations are maximal consistent sets of literals.)

(Left-to-right) Assume that X is a consistent extension of Δ_0. It is easy to verify that, for every fluent F, either F or $\neg F$ belongs to X. So there is an interpretation S such that $X = Cn_{\mathbf{F}}(S)$. Moreover, since $R \subseteq \Delta_0^X$, we know that X is closed under R; so S is a state.

(Right-to-left) Assume that S is a state, and take $X = Cn_{\mathbf{F}}(S)$. It is easy to verify that

$$\Delta_0^X = S \cup R$$

from which it follows that $X \subseteq Cn_{\mathbf{F}}(\Delta_0^X)$. Of course X is closed under S, and since S is a state, we know that X is also closed under R. And since X is logically closed, we can conclude that $X = Cn_{\mathbf{F}}(\Delta_0^X)$. □

The second fundamental lemma is actually Theorem 3.9 from Section 3.5. It will be used to show that in the consistent extensions of $\delta(D)$, non-initial situations respect the transition function *Res*.

Recall that for any state S and action name A, $\Delta(A, S)$ is the default theory obtained by taking the union of the following four sets of rules.

1. All rules of the form $\frac{: L}{L}$ where L is a frame fluent literal in S.
2. $E(A, S)$
3. All rules of the forms $\frac{: F}{F}$ and $\frac{: \neg F}{\neg F}$ where $F \in F(A, S)$.
4. R

Notice that $\Delta(A, S)$ is a default theory over $\mathcal{L}(\mathbf{F})$.

We showed in Theorem 3.9 that, for any state S and any action A that is not prohibited in S, The following hold.

1. A state S' belongs to $Res(A, S)$ if and only if $Cn_{\mathbf{F}}(S')$ is a consistent extension of $\Delta(A, S)$.

2. If X is a consistent extension of $\Delta(A, S)$, then there is a state S' such that $X = Cn_{\mathbf{F}}(S')$.

Next we prepare to move these results into the language of default theory $\delta(D)$. This will require three preliminary lemmas.

Let U be the set of atoms such that $\mathcal{L}(U)$ is the language of the default theory $\delta(D)$. We can view $\mathcal{L}(U)$ as including a tree of copies of the language $\mathcal{L}(\mathbf{F})$: one copy for each action string $\overline{A}$. For any set Γ of rules (that is, any combination of default rules, inference rules and formulas) over $\mathcal{L}(\mathbf{F})$ and any action string $\overline{A}$, let $\Sigma(\Gamma, \overline{A})$ denote the set of rules over $\mathcal{L}(U)$ obtained by replacing each occurrence of each fluent atom F in Γ by the atom $Holds(F, [\overline{A}])$. Observe that for each action string $\overline{A}$, the language $\mathcal{L}(\Sigma(\mathbf{F}, \overline{A}))$ is a subset of $\mathcal{L}(U)$ such that the rules over $\mathcal{L}(\Sigma(\mathbf{F}, \overline{A}))$ and the rules over $\mathcal{L}(\mathbf{F})$ are in one-to-one correspondence.

Lemma 4.22 *For every set Γ of inference rules over $\mathcal{L}(\mathbf{F})$ and every action string $\overline{A}$, a set X of formulas from $\mathcal{L}(\mathbf{F})$ is closed under Γ if and only if $\Sigma(X, \overline{A})$ is closed under $\Sigma(\Gamma, \overline{A})$.*

Proof. (Left-to-right) Assume X is closed under Γ. Let r' be a rule from $\Sigma(\Gamma, \overline{A})$ such that $pre(r') \in \Sigma(X, \overline{A})$. We must show that $cons(r') \in \Sigma(X, \overline{A})$. We know there is a rule $r \in \Gamma$ such that r' can be obtained from r by replacing each occurrence of each fluent atom F in r by the atom $Holds(F, [\overline{A}])$. Since $pre(r') \in \Sigma(X, \overline{A})$, we know that $pre(r) \in X$. Since X is closed under Γ, $cons(r) \in X$, from which it follows that $cons(r') \in \Sigma(X, \overline{A})$. Proof in the other direction is similar. □

Notice that the previous lemma is sufficient to establish also that X is logically closed if and only if $\Sigma(X, \overline{A})$ is.

Lemma 4.23 *For every set Γ of inference rules over $\mathcal{L}(\mathbf{F})$ and every action string $\overline{A}$, we have $\Sigma(Cn_{\mathbf{F}}(\Gamma), \overline{A}) = Cn_{\Sigma(\mathbf{F}, \overline{A})}(\Sigma(\Gamma, \overline{A}))$.*

Proof. Follows easily from the previous lemma. □

Lemma 4.24 *For every default theory D over $\mathcal{L}(\mathbf{F})$ and every action string $\overline{A}$, a set X of formulas from $\mathcal{L}(\mathbf{F})$ is an extension of D if and only if $\Sigma(X, \overline{A})$ is an extension of $\Sigma(D, \overline{A})$.*

Proof. By Lemma 4.22 we know that $X = Cn_{\mathbf{F}}(D^X)$ if and only if $\Sigma(X, \overline{A}) = \Sigma(Cn_{\mathbf{F}}(D^X), \overline{A})$. By Lemma 4.23 we have $\Sigma(Cn_{\mathbf{F}}(D^X), \overline{A}) = Cn_{\Sigma(\mathbf{F}, \overline{A})}(\Sigma(D^X, \overline{A}))$. Finally, it is not difficult to verify that $\Sigma(D^X, \overline{A}) = \Sigma(D, \overline{A})^{\Sigma(X, \overline{A})}$ which suffices to establish the lemma. □

In order to apply the two fundamental lemmas, Lemma 4.21 & Theorem 3.9, to the default theory $\delta(D)$, we will split $\delta(D)$ into simpler parts, using the Splitting Sequence Theorem. To this end, we introduce a partial mapping σ from an initial segment of ordinals $\{\alpha : \alpha < \mu\}$ to action strings, which satisfies the following three conditions.

1. For each action string $\overline{A}$ there is a non-limit ordinal $\alpha < \mu$ such that $\sigma(\alpha) = \overline{A}$.
2. For each non-limit ordinal $\alpha < \mu$ there is an action string $\overline{A}$ such that $\sigma(\alpha) = \overline{A}$.
3. For all non-limit ordinals α and β such that $\alpha < \beta < \mu$, $\sigma(\alpha) \neq \sigma(\beta)$ and the length of $\sigma(\alpha)$ is no greater than the length of $\sigma(\beta)$.

Notice that $\sigma(0) = \epsilon$.

Let $\langle U_\alpha \rangle_{\alpha < \mu}$ be the sequence of pairwise disjoint subsets of U with the following two properties.

1. For each limit ordinal $\alpha < \mu$, $U_\alpha = \emptyset$.
2. For each non-limit ordinal $\alpha < \mu$, U_α consists of all atoms from U with the situation argument $[\sigma(\alpha)]$.

Let $\langle A_\alpha \rangle_{\alpha<\mu}$ be the sequence of subsets of U such that for all $\alpha < \mu$

$$A_\alpha = \bigcup_{\gamma \leq \alpha} U_\gamma \,.$$

It is not difficult to verify that $\langle A_\alpha \rangle_{\alpha<\mu}$ is a splitting sequence for $\delta(D)$.

Now we can prove that default theory $\delta(D)$ is correct with respect to the initial situation S_0, which we can also write as $[\epsilon]$ and as $[\sigma(0)]$. We do this by showing that the default theory $b_{A_0}(\delta(D))$ behaves correctly, as follows.

Lemma 4.25 *A set X of formulas from $\mathcal{L}(U_0)$ is a consistent extension of $b_{A_0}(\delta(D))$ if and only if there is a state S such that*

$$X = Cn_{U_0}[\{Reachable(S_0)\} \cup \Sigma(S, \epsilon)] \,.$$

Proof. It is easy to verify that

$$b_{A_0}(\delta(D)) = \Sigma(\Delta_0, \epsilon) \cup \left\{ \frac{: Reachable(S_0)}{Reachable(S_0)} \right\} .$$

The lemma follows in a straightforward fashion from this observation, by Lemmas 4.21 and 4.24. □

We next show that default theory $\delta(D)$ uses the structure of the situation calculus to build what is essentially a tree of embeddings of the definition of *Res*.

For each $\alpha + 1 < \mu$, let

$$D_{\alpha+1} = b_{A_{\alpha+1}}(\delta(D)) \setminus b_{A_\alpha}(\delta(D)) \,.$$

So $D_{\alpha+1}$ is the default theory over $\mathcal{L}(A_{\alpha+1})$ which can be described as follows. Let $\overline{A}$ be the action string and A the action name such that $\sigma(\alpha+1) = \overline{A}; A$. For each sufficiency proposition ϕ **suffices for** ψ in D, we have the rule

$$\frac{Holds(\phi, [\overline{A}; A]) \wedge Reachable([\overline{A}; A])}{Holds(\psi, [\overline{A}; A])} \,.$$

For each effect proposition A **causes** ϕ **if** ψ in D, we have the rule

$$\frac{Holds(\psi, [\overline{A}]) \wedge Reachable([\overline{A}; A])}{Holds(\phi, [\overline{A}; A])} \,.$$

For each influence proposition A **possibly changes** F **if** ψ in D, we have the rules

$$\frac{Holds(\psi, [\overline{A}]) \wedge Reachable([\overline{A}; A]) : Holds(F, [\overline{A}; A])}{Holds(F, [\overline{A}; A])}$$

and

$$\frac{Holds(\psi, [\overline{A}]) \wedge Reachable([\overline{A}; A]) : \neg Holds(F, [\overline{A}; A])}{\neg Holds(F, [\overline{A}; A])} \,.$$

For each executability proposition **impossible** A **if** ψ in D, we have the rule

$$\frac{Holds(\phi, [\overline{A}])}{\neg Reachable([\overline{A}; A])} \,.$$

We also have a number of additional rules, as specified below.

Reachability axioms.

$$\frac{: Reachable([\overline{A}; A])}{Reachable([\overline{A}; A])} \quad \text{and} \quad \frac{\neg Reachable([\overline{A}])}{\neg Reachable([\overline{A}; A])} \,.$$

Inertia axioms. For each frame fluent literal L,

$$\frac{Holds(L, [\overline{A}]) \wedge Reachable([\overline{A}; A]) : Holds(L, [\overline{A}; A])}{Holds(L, [\overline{A}; A])} \,.$$

For any set Y of formulas from $\mathcal{L}(A_\alpha)$, let

$$E_{\alpha+1}(Y) = e_{A_\alpha}(D_{\alpha+1}, Y) \,.$$

Notice that $E_{\alpha+1}(Y)$ is a default theory over $\mathcal{L}(U_{\alpha+1})$.

Lemma 4.26 *Let α be such that $\alpha + 1 < \mu$. Let $\overline{A}$ be the action string and A the action name such that $\sigma(\alpha+1) = \overline{A}; A$. Let γ be such that $\sigma(\gamma) = \overline{A}$. For any logically closed set Y of formulas from $\mathcal{L}(A_\alpha)$, we have*

$$E_{\alpha+1}(Y) = E_{\alpha+1}(Y \cap \mathcal{L}(U_\gamma)) \,.$$

Proof. Follows easily from the fact that every constituent of every rule in $D_{\alpha+1}$ belongs to $\mathcal{L}(U_\gamma) \cup \mathcal{L}(U_{\alpha+1})$. □

Lemma 4.27 *Let α be such that $\alpha + 1 < \mu$. Let $\overline{A}$ be the action string and A the action name such that $\sigma(\alpha+1) = \overline{A}; A$. Let γ be such that $\sigma(\gamma) = \overline{A}$. Let $Y = Cn_{U_\gamma}(\{\neg Reachable([\overline{A}])\})$. Let S be a state. Let $Z = Cn_{U_\gamma}[\{Reachable([\overline{A}])\} \cup \Sigma(S, \overline{A})]$. The following hold.*

1. *The unique extension of $E_{\alpha+1}(Y)$ is $Cn_{U_{\alpha+1}}(\{\neg Reachable([\overline{A}; A])\})$.*
2. *If A is prohibited in S, then the unique extension of $E_{\alpha+1}(Z)$ is*

$$Cn_{U_{\alpha+1}}(\{\neg Reachable([\overline{A}; A])\}).$$

3. *If A is not prohibited in S, then X is an extension of $E_{\alpha+1}(Z)$ if and only if there is a state $S' \in Res(A, S)$ such that*

$$X = Cn_{U_{\alpha+1}}[\{Reachable([\overline{A}; A])\} \cup \Sigma(S', \overline{A}; A)].$$

Proof. For this lemma we will use the Splitting Set Theorem, with splitting set $B = \{Reachable([\overline{A}; A])\}$. Notice that B splits both $E_{\alpha+1}(Y)$ and $E_{\alpha+1}(Z)$.

For the first part, it's not hard to verify that the unique extension of default theory $b_B(E_{\alpha+1}(Y))$ is $Cn_B(\{\neg Reachable([\overline{A}; A])\})$. Moreover, it is easy to verify that

$$e_B(E_{\alpha+1}(Y), Cn_B(\{\neg Reachable([\overline{A}; A])\})) = \emptyset.$$

Thus

$$\langle\ Cn_B(\{\neg Reachable([\overline{A}; A])\}),\ Cn_{\Sigma(\mathbf{F}, \overline{A}; A)}(\emptyset)\ \rangle$$

is a solution to $E_{\alpha+1}(Y)$ with respect to B, and by the Splitting Set Theorem it follows that $Cn_{U_{\alpha+1}}(\{\neg Reachable([\overline{A}; A])\})$ is the unique extension of $E_{\alpha+1}(Y)$.

For part two, assume that A is prohibited in S. Thus there is fluent formula ϕ such that the rule

$$\frac{Holds(\phi, [\overline{A}])}{\neg Reachable(\overline{A}; A)}$$

belongs to $D_{\alpha+1}$ and ϕ is satisfied in S. Since ϕ is satisfied in S, $\phi \in Cn_{\mathbf{F}}(S)$. Thus we have $Holds(\phi, [\overline{A}]) \in \Sigma(Cn_{\mathbf{F}}(S), \overline{A})$, and it follows by Lemma 4.23 that $Holds(\phi, [\overline{A}]) \in Cn_{\Sigma(\mathbf{F}, \overline{A})}(\Sigma(S, \overline{A}))$. And since $Cn_{\Sigma(\mathbf{F}, \overline{A})}(\Sigma(S, \overline{A})) \subseteq Z$, we have $Holds(\phi, [\overline{A}]) \in Z$. It follows easily that the unique extension of $b_B(E_{\alpha+1}(Z))$ is $Cn_B(\{\neg Reachable([\overline{A}; A])\})$. Again, it is easy to verify that

$$e_B(E_{\alpha+1}(Z), Cn_B(\{\neg Reachable([\overline{A}; A])\})) = \emptyset.$$

Thus, by essentially the same reasoning as in the previous case, we can conclude that the unique extension of $E_{\alpha+1}(Z)$ is $Cn_{U_{\alpha+1}}(\{\neg Reachable([\overline{A}; A])\})$.

For the third part, assume that A is not prohibited in S. Reasoning much as before, we see that unique extension of $b_B(E_{\alpha+1}(Z))$ is $Cn_B(B)$. Now take

$$D' = e_B(E_{\alpha+1}(Z), Cn_B(B)).$$

The key step is to recognize that

$$D' = \Sigma(\Delta(A, S), \overline{A}; A).$$

Thus, we can apply Theorem 3.9 relating $Res(A, S)$ and $\Delta(A, S)$, as follows.

(Left-to-right) Assume that X is an extension of $E_{\alpha+1}(Z)$. By the Splitting Set Corollary, $\langle X \cap \mathcal{L}(B), X \cap \mathcal{L}(\Sigma(\mathbf{F}, \overline{A}; A))\rangle$ is a solution to $E_{\alpha+1}(Z)$ with respect to B, and it follows immediately that $X \cap \mathcal{L}(B)$ is a consistent extension of $b_B(E_{\alpha+1}(Z))$ and that $X \cap \mathcal{L}(\Sigma(\mathbf{F}, \overline{A}; A))$ is a consistent extension of D'. Let $X' = X \cap \mathcal{L}(\Sigma(\mathbf{F}, \overline{A}; A))$. Notice that $X = Cn_{U_{\alpha+1}}(B \cup X')$. Since $D' = \Sigma(\Delta(A, S), \overline{A}; A)$, we know by Theorem 3.9 and Lemma 4.24 that there is a state $S' \in Res(A, S)$ such that we have $X' = Cn_{\Sigma(\mathbf{F}, \overline{A}; A)}[\Sigma(S', \overline{A}; A)]$. It follows that $X = Cn_{U_{\alpha+1}}(B \cup \Sigma(S', \overline{A}; A))$.

(Right-to-left) Assume there is a state $S' \in Res(A, S)$ such that we have $X = Cn_{U_{\alpha+1}}(B \cup \Sigma(S', \overline{A}; A))$. We know that $Cn_B(B)$ is a consistent extension of $b_B(E_{\alpha+1}(Z))$. Let $X' = Cn_{\Sigma(\mathbf{F},\overline{A};A)}[\Sigma(S', \overline{A}; A)]$. Since $D' = \Sigma(\Delta(A, S), \overline{A}; A)$. we know by Theorem 3.9 and Lemma 4.24 that X' is a consistent extension of D'. It follows by the Splitting Set Theorem that X is an extension of $E_{\alpha+1}(Z)$. □

At this point we have applied the fundamental lemmas, Lemma 4.21 & Theorem 3.9, to the parts of $\delta(D)$ that we obtain using the splitting sequence $\langle A_\alpha \rangle_{\alpha<\mu}$. The resulting lemmas (4.25 & 4.27) capture essential properties of $\delta(D)$ in a form that will be convenient for our proof of the Correspondence Lemma.

In what follows, we will first show that for any model Ψ of D we can construct a corresponding extension of $\delta(D)$ (Lemma 4.33). It will then remain to show that each consistent extension of $\delta(D)$ corresponds to a unique model of D (Lemma 4.40). These results together will establish that there is indeed a one-to-one correspondence between models of D and consistent extensions of $\delta(D)$. Moreover, we'll show that each model and its corresponding extension agree on the truth of all value propositions, which will suffice to establish the Correspondence Lemma.

At this point we associate with each structure for D a unique set of literals from $\mathcal{L}(U)$. It will be our goal to show that, for any model of D, the associated set of literals is a consistent extension of $\delta(D)$.

For any structure Ψ for D, let $\delta(\Psi)$ be the least set of literals from $\mathcal{L}(U)$ such that for every action string $\overline{A}$:

1. if $\overline{A} \notin Dom(\Psi)$, then $\neg Reachable([\overline{A}]) \in \delta(\Psi)$; and
2. if $\overline{A} \in Dom(\Psi)$, then $Reachable([\overline{A}]) \in \delta(\Psi)$ and $\Sigma(\Psi(\overline{A}), \overline{A}) \subseteq \delta(\Psi)$.

In Lemma 4.29 below, we will show that $\delta(\Psi)$ has the following crucial property: a value proposition V is true in Ψ if and only if $[V] \in Cn_U(\delta(\Psi))$.

Lemma 4.28 *Let Ψ be a structure for D. For all action strings $\overline{A}$, we have*

$$Cn_U(\delta(\Psi)) \cap \mathcal{L}(\Sigma(\mathbf{F}, \overline{A})) = Cn_{\Sigma(\mathbf{F},\overline{A})}(\delta(\Psi) \cap \mathcal{L}(\Sigma(\mathbf{F}, \overline{A}))) .$$

Proof. Straightforward. □

Lemma 4.29 *A value proposition V is true in a structure Ψ for D if and only if $[V] \in Cn_U(\delta(\Psi))$.*

Proof. Notice that it is sufficient to prove that the lemma holds for all atomic value propositions. So consider any value proposition ϕ **after** $\overline{A}$.

(Left-to-right) Assume that ϕ **after** $\overline{A}$ is true in Ψ. Thus, $\overline{A} \in Dom(\Psi)$ and ϕ is satisfied in $\Psi(\overline{A})$. Since ϕ is satisfied in $\Psi(\overline{A})$, we know that $\phi \in Cn_{\mathbf{F}}(\Psi(\overline{A}))$. Thus, $Holds(\phi, [\overline{A}]) \in \Sigma[Cn_{\mathbf{F}}(\Psi(\overline{A})), \overline{A}]$. We can conclude by Lemma 4.23 that $Holds(\phi, [\overline{A}]) \in Cn_U(\Sigma(\Psi(\overline{A}), \overline{A}))$. By the definition of $\delta(\Psi)$, $Reachable([\overline{A}]) \in \delta(\Psi)$ and $\Sigma(\Psi(\overline{A}), \overline{A}) \subseteq \delta(\Psi)$. So we've shown that $Reachable([\overline{A}]) \wedge Holds(\phi, [\overline{A}]) \in Cn_U(\delta(\Psi))$. That is, $[\phi$ **after** $\overline{A}] \in Cn_U(\delta(\Psi))$.

(Right-to-left) Assume that $[\phi$ **after** $\overline{A}] \in Cn_U(\delta(\Psi))$. That is just to say that $Reachable([\overline{A}]) \wedge Holds(\phi, [\overline{A}]) \in Cn_U(\delta(\Psi))$. Thus $Holds(\phi, [\overline{A}]) \in Cn_U(\delta(\Psi))$ and $Reachable([\overline{A}]) \in Cn_U(\delta(\Psi))$. By the definition of $\delta(\Psi)$, it follows that $\overline{A} \in Dom(\Psi)$. Again by the definition of $\delta(\Psi)$, we can conclude that $\Sigma(\Psi(\overline{A}), \overline{A}) = \delta(\Psi) \cap \mathcal{L}(\Sigma(\mathbf{F}, \overline{A}))$. Thus, $Cn_{\Sigma(\mathbf{F},\overline{A})}(\Sigma(\Psi(\overline{A}), \overline{A})) = Cn_{\Sigma(\mathbf{F},\overline{A})}(\delta(\Psi) \cap \mathcal{L}(\Sigma(\mathbf{F}, \overline{A})))$. By Lemmas 4.23 and 4.28 it follows that

$$\Sigma[Cn_{\mathbf{F}}(\Psi(\overline{A})), \overline{A}] = Cn_U(\delta(\Psi)) \cap \mathcal{L}(\Sigma(\mathbf{F}, \overline{A})) .$$

And since we know that $Holds(\phi, [\overline{A}]) \in Cn_U(\delta(\Psi)) \cap \mathcal{L}(\Sigma(\mathbf{F}, \overline{A}))$, it follows that $Holds(\phi, [\overline{A}]) \in \Sigma[Cn_{\mathbf{F}}(\Psi(\overline{A})), \overline{A}]$. So $\phi \in Cn_{\mathbf{F}}(\Psi(\overline{A}))$. That is, $\Psi(\overline{A})$ satisfies ϕ and thus ϕ **after** $\overline{A}$ is true in Ψ. □

Now we begin the main part of the proof of the left-to-right direction of the Correspondence Lemma.

Let Ψ be a model of D. We will show that $Cn_U(\delta(\Psi))$ is an extension of $\delta(D)$. We begin by putting $Cn_U(\delta(\Psi))$ in a form more suitable for application of the Splitting Sequence Theorem.

Let $\langle X_\alpha \rangle_{\alpha<\mu}$ be defined as follows.

1. $X_0 = Cn_{U_0}(\delta(\Psi) \cap \mathcal{L}(U_0))$.
2. For all α such that $\alpha + 1 < \mu$, $X_{\alpha+1} = Cn_{U_{\alpha+1}}(\delta(\Psi) \cap \mathcal{L}(U_{\alpha+1}))$.
3. For all limit ordinals $\alpha < \mu$, $X_\alpha = Cn_\emptyset(\emptyset)$.

Lemma 4.30 *We have*

$$Cn_U(\delta(\Psi)) = Cn_U\left(\bigcup_{\alpha<\mu} X_\alpha\right).$$

Proof. It is straightforward to verify that

$$\delta(\Psi) = \bigcup_{\alpha<\mu} (\delta(\Psi) \cap \mathcal{L}(U_\alpha)).$$

It is clear from the definitions that for all $\alpha < \mu$, $X_\alpha = Cn_{U_\alpha}(\delta(\Psi) \cap \mathcal{L}(U_\alpha))$. The lemma follows easily from these observations. □

We will show that $\langle X_\alpha \rangle_{\alpha<\mu}$ is a solution to $\delta(D)$ with respect to $\langle A_\alpha \rangle_{\alpha<\mu}$.

Lemma 4.31 *Let α be a non-limit ordinal such that $\alpha < \mu$. Let $\overline{A}$ be the action string such that $\overline{A} = \sigma(\alpha)$. The following hold.*

1. *If $\overline{A} \in Dom(\Psi)$ then $X_\alpha = Cn_{U_\alpha}[\{Reachable([\overline{A}])\} \cup \Sigma(\Psi(\overline{A}), \overline{A})]$.*
2. *If $\overline{A} \notin Dom(\Psi)$ then $X_\alpha = Cn_{U_\alpha}(\{\neg Reachable([\overline{A}])\})$.*

Proof. By the definition of $\langle X_\alpha \rangle_{\alpha<\mu}$ we know that $X_\alpha = Cn_{U_\alpha}(\delta(\Psi) \cap \mathcal{L}(U_\alpha))$. For part one, assume that $\overline{A} \in Dom(\Psi)$. From the definition of $\delta(\Psi)$ we can conclude that $\delta(\Psi) \cap \mathcal{L}(U_\alpha) = \{Reachable([\overline{A}])\} \cup \Sigma(\Psi(\overline{A}), \overline{A})$. For part two, assume that $\overline{A} \notin Dom(\Psi)$. From the definition of $\delta(\Psi)$ we can conclude that $\delta(\Psi) \cap \mathcal{L}(U_\alpha) = \{\neg Reachable([\overline{A}])\}$. □

Lemma 4.32 *For each α such that $\alpha + 1 < \mu$, $X_{\alpha+1}$ is a consistent extension of the default theory*

$$E_{\alpha+1}\left(\bigcup_{\beta\leq\alpha} X_\beta\right)$$

over $\mathcal{L}(U_{\alpha+1})$.

Proof. Let $\overline{A}$ be the action string and A the action name such that $\sigma(\alpha+1) = \overline{A}; A$. Let γ be such that $\sigma(\gamma) = \overline{A}$. By the definition of $E_{\alpha+1}$ we know that

$$E_{\alpha+1}\left(\bigcup_{\beta\leq\alpha} X_\beta\right) = E_{\alpha+1}\left(Cn_{A_\alpha}\left(\bigcup_{\beta\leq\alpha} X_\beta\right)\right).$$

It is easy to verify that

$$Cn_{A_\alpha}\left(\bigcup_{\beta\leq\alpha} X_\beta\right) \cap \mathcal{L}(U_\gamma) = X_\gamma.$$

Thus, by Lemma 4.26 we can conclude that

$$E_{\alpha+1}\left(\bigcup_{\beta\leq\alpha} X_\beta\right) = E_{\alpha+1}(X_\gamma).$$

So we will show that $X_{\alpha+1}$ is an extension of $E_{\alpha+1}(X_\gamma)$. Consider three cases.

Case 1: $\overline{A}; A \in Dom(\Psi)$. Since the domain of Ψ is prefix-closed, we have $\overline{A} \in Dom(\Psi)$. Let $S = \Psi(\overline{A})$ and $S' = \Psi(\overline{A}; A)$. By the previous lemma we have the following.

$$\begin{aligned} X_\gamma &= Cn_{U_\gamma}[\{Reachable([\overline{A}])\} \cup \Sigma(S, \overline{A})] \\ X_{\alpha+1} &= Cn_{U_{\alpha+1}}[\{Reachable([\overline{A}; A])\} \cup \Sigma(S', \overline{A}; A)] \end{aligned}$$

Since $\overline{A}; A \in Dom(\Psi)$, we know that A is not prohibited in S. Furthermore, since Ψ is a model of D, we have $S' \in Res(A, S)$. Thus, by part three of Lemma 4.27, $X_{\alpha+1}$ is a consistent extension of $E_{\alpha+1}(X_\gamma)$.

Case 2: $\overline{A}; A \notin Dom(\Psi)$ and $\overline{A} \in Dom(\Psi)$. Let $S = \Psi(\overline{A})$. In this case, X_γ is the same as in the previous case. By the previous lemma

$$X_{\alpha+1} = Cn_{U_{\alpha+1}}(\{\neg Reachable([\overline{A}; A])\}) .$$

Since D is a qualification-free domain, and $\overline{A}; A \notin Dom(\Psi)$ while $\overline{A} \in Dom(\Psi)$, we can conclude that A is prohibited in S. Thus, by part two of Lemma 4.27, $X_{\alpha+1}$ is a consistent extension of $E_{\alpha+1}(X_\gamma)$.

Case 3: $\overline{A}; A \notin Dom(\Psi)$ and $\overline{A} \notin Dom(\Psi)$. In this case, $X_{\alpha+1}$ is the same as in the previous case. By the previous lemma

$$X_\gamma = Cn_{U_\gamma}(\{\neg Reachable([\overline{A}])\}) .$$

By part one of Lemma 4.27, $X_{\alpha+1}$ is a consistent extension of $E_{\alpha+1}(X_\gamma)$. □

Lemma 4.33 *Let D be a qualification-free domain theory without value propositions. If Ψ is a model of D, then $Cn_U(\delta(\Psi))$ is a consistent extension of $\delta(D)$.*

Proof. First, the fact that X_0 is a consistent extension of $b_{A_0}(\delta(D))$ follows easily from Lemma 4.25. Second, the previous lemma shows that for each α such that $\alpha + 1 < \mu$, $X_{\alpha+1}$ is a consistent extension of

$$E_{\alpha+1}\left(\bigcup_{\gamma \leq \alpha} X_\gamma\right) .$$

Finally, by the definition of $\langle X_\alpha \rangle_{\alpha<\mu}$, we know that for all limit ordinals $\alpha < \mu$, $X_\alpha = Cn_\emptyset(\emptyset)$. These observations are sufficient to establish that $\langle X_\alpha \rangle_{\alpha<\mu}$ is a solution to $\delta(D)$ with respect to $\langle A_\alpha \rangle_{\alpha<\mu}$. By Lemma 4.30

$$Cn_U(\delta(\Psi)) = Cn_U\left(\bigcup_{\alpha<\mu} X_\alpha\right)$$

so we can conclude by the Splitting Sequence Theorem that $Cn_U(\delta(\Psi))$ is a consistent extension of $\delta(D)$. □

We have essentially established the left-to-right direction of the Correspondence Lemma. Now we turn our attention to the other direction.

Let X be a consistent extension of $\delta(D)$. We will show that there is a (unique) model Ψ of D such that $X = Cn_U(\delta(\Psi))$ (Lemma 4.40). To this end, we specify a construction that, as we will show, yields a unique model of D for each consistent extension of $\delta(D)$.

Let Ψ_X be the partial function from actions strings to sets of fluent literals that satisfies the following two conditions.

1. $Dom(\Psi_X) = \{\overline{A} : Reachable([\overline{A}]) \in X\}$.
2. For all non-limit ordinals $\alpha < \mu$, if $\sigma(\alpha) \in Dom(\Psi_X)$, then $\Psi_X(\sigma(\alpha))$ is the greatest set of fluent literals such that $\Sigma(\Psi_X(\sigma(\alpha)), \sigma(\alpha)) \subseteq X \cap \mathcal{L}(U_\alpha)$.

Thus, for every action string $\overline{A} \in Dom(\Psi_X)$, $\Psi_X(\overline{A})$ is the greatest set S of fluent literals such that $\Sigma(S, \overline{A})$ is a subset of X. One thing we will show is that such sets S are in fact states (Lemma 4.36). More generally, we will establish the fact that Ψ_X is a structure for D such that $X = Cn_U(\delta(\Psi_X))$ (Lemma 4.37).

Lemma 4.34 *The domain of Ψ_X is nonempty and prefix-closed.*

Proof. By Lemma 4.25 (and the Splitting Sequence Corollary), we can conclude that $Reachable(S_0) \in X$. Thus the domain of Ψ_X is nonempty.

For every action string $\overline{A}$, $\delta(D)$ includes the rule

$$\frac{: Reachable([\overline{A}])}{Reachable([\overline{A}])}$$

from which it follows that for every action string $\overline{A}$, either $Reachable([\overline{A}]) \in X$ or $\neg Reachable([\overline{A}]) \in X$. Furthermore, for every action string $\overline{A}$ and action name A, $\delta(D)$ includes the rule

$$\frac{\neg Reachable([\overline{A}])}{\neg Reachable([\overline{A}; A])}$$

which guarantees that if $\neg Reachable([\overline{A}]) \in X$ then $\neg Reachable([\overline{A}; A]) \in X$. Thus we can conclude that if $\overline{A} \notin Dom(\Psi_X)$ then $\overline{A}; A \notin Dom(\Psi_X)$, which is just to say that the domain of Ψ_X is prefix-closed. □

Once again we will be looking to apply Lemmas 4.25 and 4.27. In order to do this, we need an appropriate sequence, which is defined next.

Let $\langle X_\alpha \rangle_{\alpha<\mu}$ be the sequence such that for every $\alpha < \mu$, $X_\alpha = X \cap \mathcal{L}(U_\alpha)$. We know by the Splitting Sequence Corollary that $\langle X_\alpha \rangle_{\alpha<\mu}$ is a solution to $\delta(D)$ with respect to $\langle A_\alpha \rangle_{\alpha<\mu}$. Moreover, it is not hard to verify that

$$X = Cn_U\left(\bigcup_{\alpha<\mu} X_\alpha\right).$$

Lemma 4.35 *Let α be such that $\alpha + 1 < \mu$. Let $\overline{A}$ be the action string and A the action name such that $\sigma(\alpha+1) = \overline{A}; A$. The following hold.*

1. *If $Reachable([\overline{A}; A]) \in X_{\alpha+1}$, then there is a state S such that*

$$X_{\alpha+1} = Cn_{U_{\alpha+1}}[\{Reachable([\overline{A}; A])\} \cup \Sigma(S, \overline{A}; A)].$$

2. *If $Reachable([\overline{A}; A]) \notin X_{\alpha+1}$, then*

$$X_{\alpha+1} = Cn_{U_{\alpha+1}}(\{\neg Reachable([\overline{A}; A])\}).$$

Proof. Proof is by induction on α. Let γ be such that $\sigma(\gamma) = \overline{A}$. By the definition of $E_{\alpha+1}$ we know that

$$E_{\alpha+1}\left(\bigcup_{\beta\leq\alpha} X_\beta\right) = E_{\alpha+1}\left(Cn_{A_\alpha}\left(\bigcup_{\beta\leq\alpha} X_\beta\right)\right).$$

It is not difficult to verify that

$$Cn_{A_\alpha}\left(\bigcup_{\beta\leq\alpha} X_\beta\right) \cap \mathcal{L}(U_\gamma) = X_\gamma.$$

Thus, by Lemma 4.26 we can conclude that

$$E_{\alpha+1}\left(\bigcup_{\beta\leq\alpha} X_\beta\right) = E_{\alpha+1}(X_\gamma).$$

So $X_{\alpha+1}$ is an extension of $E_{\alpha+1}(X_\gamma)$. Now consider three cases.

Case 1: $\gamma = 0$. Thus $\overline{A} = \epsilon$. By Lemma 4.25 there is a state S such that

$$X_\gamma = Cn_{U_\gamma}[\{Reachable([\overline{A}])\} \cup \Sigma(S, \overline{A})].$$

To show part one for this case, assume that $Reachable([\overline{A}; A]) \in X_{\alpha+1}$. By part two of Lemma 4.27 we can conclude that A is not prohibited in S and the desired conclusion then follows from part three of Lemma 4.27. Part two for this case can be proved similarly.

Case 2: $\gamma \neq 0$ and $Reachable([\overline{A}]) \in X_\gamma$. In this case we use the inductive hypothesis, which guarantees that there is a state S such that

$$X_\gamma = Cn_{U_\gamma}[\{Reachable([\overline{A}])\} \cup \Sigma(S, \overline{A})].$$

From this point the proof proceeds as in the previous case.

Case 3: $\gamma \neq 0$ and $Reachable([\overline{A}]) \notin X_\gamma$. In this case we again use the inductive hypothesis, which guarantees that $X_\gamma = Cn_{U_\gamma}(\{\neg Reachable([\overline{A}])\})$. We reach the desired conclusion by applying part one of Lemma 4.27. □

Lemma 4.36 *Ψ_X is a partial function from action strings to states.*

Proof. We show that for all non-limit ordinals $\alpha < \mu$, if $\sigma(\alpha) \in Dom(\Psi_X)$, then $\Psi_X(\sigma(\alpha))$ is a state. First, if $\alpha = 0$, we can conclude by Lemma 4.25 (and the Splitting Sequence Corollary), along with the definition of Ψ_X, that $\sigma(\alpha) \in Dom(\Psi_X)$ and that $\Psi_X(\sigma(\alpha))$ is a state. Let α be such that $\alpha+1 < \mu$. If $\sigma(\alpha+1) \in Dom(\Psi_X)$, then $Reachable([\sigma(\alpha+1)]) \in X_{\alpha+1}$, and we can conclude by the previous lemma that $\Psi_X(\sigma(\alpha+1))$ is a state. □

Lemma 4.37 *Ψ_X is a structure for D such that $X = Cn_U(\delta(\Psi_X))$.*

Proof. By Lemma 4.34 and the previous lemma, Ψ_X is partial function from action strings to states whose domain is nonempty and prefix-closed, which shows that Ψ_X is a structure for D. Thus, $\delta(\Psi_X)$ is defined. We must show that $X = Cn_U(\delta(\Psi_X))$. To begin, it is easy to verify that

$$Cn_U(\delta(\Psi_X)) = Cn_U\left(\bigcup_{\alpha<\mu} Cn_{U_\alpha}(\delta(\Psi_X) \cap \mathcal{L}(U_\alpha))\right).$$

Recall that

$$X = Cn_U\left(\bigcup_{\alpha<\mu} X_\alpha\right).$$

Given these observations, we see that it will be sufficient to show that for every $\alpha < \mu$, $X_\alpha = Cn_{U_\alpha}(\delta(\Psi_X) \cap \mathcal{L}(U_\alpha))$.

If α is a limit ordinal, this is trivially true; so assume that α is a non-limit ordinal and let $\overline{A}$ be the action string such that $\overline{A} = \sigma(\alpha)$. Now consider two cases.

Case 1: $\overline{A} \notin Dom(\Psi_X)$. In this case, by the definition of δ, we have $\delta(\Psi_X) \cap \mathcal{L}(U_\alpha) = \{\neg Reachable([\overline{A}])\}$. Similarly, by the definition of Ψ_X, we know that $Reachable([\overline{A}]) \notin X_\alpha$. It follows by Lemma 4.25 that $\alpha \neq 0$. Thus, by part two of Lemma 4.35 we can conclude that $X_\alpha = Cn_{U_\alpha}(\delta(\Psi_X) \cap \mathcal{L}(U_\alpha))$.

Case 2: $\overline{A} \in Dom(\Psi_X)$. By the definition of Ψ_X, we have $Reachable([\overline{A}]) \in X_\alpha$. Now, if $\overline{A} = \epsilon$ we know by Lemma 4.25 that there is a state S such that

$$X_\alpha = Cn_{U_\alpha}(\{Reachable([\overline{A}])\} \cup \Sigma(S, \overline{A})).$$

On the other hand, if $\overline{A} \neq \epsilon$, the same thing follows from part one of Lemma 4.35. By the definition of Ψ_X we know that $\Sigma(\Psi_X(\overline{A}), \overline{A}) \subseteq X_\alpha$. Since X_α is consistent, we can conclude that $\Psi_X(\overline{A}) = S$. It follows by the definition of δ that

$$\delta(\Psi_X) \cap \mathcal{L}(U_\alpha) = \{Reachable([\overline{A}])\} \cup \Sigma(S, \overline{A}).$$

Thus $X_\alpha = Cn_{U\alpha}(\delta(\Psi_X) \cap \mathcal{L}(U_\alpha))$. □

Now that we know Ψ_X is a structure for D, we'll need just two more lemmas in order to establish that Ψ_X is in fact a model for D (Lemma 4.40).

Lemma 4.38 *For all $\overline{A} \in Dom(\Psi_X)$ and all action names A, if $Res(A, \Psi_X(\overline{A}))$ is nonempty, then $\overline{A}; A \in Dom(\Psi_X)$.*

Proof. Let $S = \Psi_X(\overline{A})$. By Lemma 4.36, S is a state. Let α and γ be such that $\sigma(\alpha + 1) = \overline{A}; A$ and $\sigma(\gamma) = \overline{A}$. By the construction of Ψ_X from X, we know that $\Sigma(S, \overline{A}) \subseteq X_\gamma$, and also that $Reachable([\overline{A}]) \in X_\gamma$. If $\gamma = 0$ it follows by Lemma 4.25 that

$$X_\gamma = Cn_{U_\gamma}[\{Reachable([\overline{A}])\} \cup \Sigma(S, \overline{A})]$$

since X_γ is consistent. On the other hand, if $\gamma \neq 0$, the same thing follows from part one of Lemma 4.35. Since $Res(A, S)$ is nonempty, we know that A is not prohibited in S. It follows by part three of Lemma 4.27 that $Reachable([\overline{A}; A]) \in X_{\alpha+1}$. Thus, by the construction of Ψ_X from X, $\overline{A}; A \in Dom(\Psi_X)$. □

Lemma 4.39 *For all $\overline{A}; A \in Dom(\Psi_X)$, $\Psi_X(\overline{A}; A) \in Res(A, \Psi_X(\overline{A}))$.*

Proof. By Lemma 4.34, $\overline{A} \in Dom(\Psi_X)$, since $\overline{A}; A$ is. Let $S = \Psi_X(\overline{A})$ and $S' = \Psi_X(\overline{A}; A)$. Let α and γ be such that $\sigma(\alpha + 1) = \overline{A}; A$ and $\sigma(\gamma) = \overline{A}$. By Lemma 4.36, S and S' are states. By essentially the same reasoning used in the proof of the previous lemma, we can show each of the following.

$$\begin{aligned} X_\gamma &= Cn_{U_\gamma}[\{Reachable([\overline{A}])\} \cup \Sigma(S, \overline{A})] \\ X_{\alpha+1} &= Cn_{U_{\alpha+1}}[\{Reachable([\overline{A}; A])\} \cup \Sigma(S', \overline{A}; A)] \end{aligned}$$

It follows from part two of Lemma 4.27 that A is not prohibited in S. We can conclude by part three of Lemma 4.27 that $S' \in Res(A, S)$. That is, $\Psi_X(\overline{A}; A) \in Res(A, \Psi_X(\overline{A}))$. □

Lemma 4.40 *Let D be a qualification-free domain theory without value propositions. If X is a consistent extension of $\delta(D)$, then Ψ_X is a (unique) model of D such that $X = Cn_U(\delta(\Psi_X))$.*

Proof. By Lemma 4.37, Ψ_X is a structure for D such that $X = Cn_U(\delta(\Psi_X))$. Moreover, it is clear that Ψ_X is the only such structure. Lemma 4.36 shows that $\Psi_X(\epsilon)$ is a state. Given this, Lemmas 4.38 and 4.39 establish the fact that Ψ_X is a model of D. □

Lemma 4.41 (Correspondence Lemma) *Let D be a qualification-free $\mathcal{AC}$ domain description without value propositions. There is a one-to-one correspondence between models of D and consistent extensions of $\delta(D)$ such that a value proposition V is true in a model of D if and only if the formula $[V]$ belongs to the corresponding extension of $\delta(D)$.*

Proof. We have defined a total function from models Ψ of D to consistent extensions $Cn_U(\delta(\Psi))$ of $\delta(D)$ (Lemma 4.33). Notice that this function is injective. To see that it is also surjective, notice that we have also defined a total function from consistent extensions X of $\delta(D)$ to models Ψ_X of D such that $X = Cn_U(\delta(\Psi_X))$ (Lemma 4.40). Thus δ can be used to define a one-to-one correspondence between models of D and consistent extensions of $\delta(D)$. Finally, we've shown that a value proposition V is true in a model Ψ of D if and only if the formula $[V]$ belongs to the corresponding extension $Cn_U(\delta(\Psi))$ of $\delta(D)$ (Lemma 4.29). □

We require one more lemma for the proof of the Correspondence Theorem.

Lemma 4.42 *Let D be a qualification-free domain description. Let D' be the domain description obtained by deleting all value propositions from D. Let E be a consistent set of formulas from $\mathcal{L}(U)$. $E = Cn_U(\delta(D)^E)$ if and only if $E = Cn_U(\delta(D')^E)$ and for every value proposition $V \in D$, $[V] \in E$.*

Proof. (Left-to-right) Assume that $E = Cn_U(\delta(D)^E)$. It is clear that $E = Cn_U(\delta(D')^E)$, since every rule in $\delta(D) \setminus \delta(D')$ has the form $\frac{\phi}{False}$. By the Correspondence Lemma, there is a model Ψ of D' such that for every value proposition V, V is true in Ψ if and only if $[V] \in E$. Consider any value proposition $V \in D$. Since $\frac{\neg[V]}{False} \in \delta(D)$, we know that $\neg[V] \notin E$. It follows that $\neg V$ is not true in Ψ, and thus that V is true in Ψ. So we can conclude that $[V] \in E$.

Proof in the other direction is similar, but slightly simpler. □

Proof of Correspondence Theorem. Let D be a qualification-free domain description. Let D' be the domain description obtained by deleting all value propositions from D. By the Correspondence Lemma, we know that there is a one-to-one correspondence between models of D' and consistent extensions of $\delta(D')$ such that a value proposition V is true in a model of D' if and only if the formula $[V]$ belongs to the corresponding extension of $\delta(D')$.

Let C be the set of pairs $\langle \Psi, E \rangle$ such that Ψ is a model of D' and E is the corresponding extension of $\delta(D')$. Since every model of D is a model of D' and similarly every consistent extension of $\delta(D)$ is a consistent extension of $\delta(D')$, we can complete our proof by showing that, for each pair $\langle \Psi, E \rangle$ that belongs to C, Ψ is a model of D if and only if E is a consistent extension of $\delta(D)$. So consider any $\langle \Psi, E \rangle \in C$.

(Left-to-right) Assume that Ψ is a model of D. We can conclude by the Correspondence Lemma that for every value proposition $V \in D$, $[V] \in E$. Since E is a consistent extension of $\delta(D')$, it follows by the previous lemma that E is a consistent extension of $\delta(D)$.

(Right-to-left) Assume that E is a consistent extension of $\delta(D)$. It follows by the previous lemma that E is a consistent extension of $\delta(D')$ such that for every value proposition $V \in D$, $[V] \in E$. We can conclude by the Correspondence Lemma that Ψ is a model of D' that satisfies every value proposition in D. That is, Ψ is a

model of D. □

Proof of Reachability Corollary. Let A be the set of all *Reachable* atoms in U. Since D includes no executability propositions, A is a splitting set for $\delta(D)$. Furthermore, it is clear that $Cn_A(A)$ is the unique extension of $b_A(\delta(D))$. Notice that

$$\delta'(D) = e_A(\delta(D), Cn_A(A)).$$

It follows by the Splitting Set Theorem that there is a one-to-one correspondence between the consistent extensions of $\delta(D)$ and the consistent extensions of $\delta'(D)$ such that for every value proposition V, $[V]$ belongs to a consistent extension of $\delta(D)$ if and only if $[\![V]\!]$ belongs to the corresponding extension of $\delta'(D)$. Given this fact, the corollary follows immediately from the Correspondence Theorem. □

4.5 Proof of LP Correspondence Theorem, LP Reachability Corollary, and Vivid Domains Theorem

Proof of the LP Correspondence Theorem is based on the following lemma, which is shown to follow from the Correspondence Theorem for default logic.

Lemma 4.43 (LP Correspondence Lemma)
Let D be an LP-simple, qualification-free $\mathcal{AC}$ domain description without value propositions. There is a one-to-one correspondence between models of D and consistent answer sets of $\pi(D)$ such that, for every model Ψ of D and corresponding answer set X, an LP-simple value proposition $V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$ is true in Ψ if and only if at least one of the sets $\{ [\![V_1]\!], \ldots, [\![V_m]\!] \} \cap X$ and $\{ [\![V_{m+1}]\!], \ldots, [\![V_n]\!] \} \setminus X$ is nonempty.

Proof. To begin, because D includes no value propositions, it is straightforward to determine that the default theories $dt(\pi(D))$ and $\delta(D)$ have precisely the

same extensions. By the Correspondence Theorem, we know there is a one-to-one correspondence between models of D and consistent extensions of $\delta(D)$ such that, for every model Ψ of D and corresponding extension E, a value proposition V is true in Ψ if and only if $[V] \in E$. Let C be the set of pairs $\langle \Psi, E \rangle$ such that Ψ is a model of D and E is the corresponding consistent extension of $\delta(D)$. Since $\delta(D)$ and $dt(\pi(D))$ have the same extensions, we know that the set $\{ E \cap Lit(U) : E \textit{ is a consistent extension of } \delta(D) \}$ is precisely the set of consistent answer sets for $\pi(D)$. Take $A = \{ \langle \Psi, E \cap Lit(U) \rangle : \langle \Psi, E \rangle \in C \}$.

Consider any $\langle \Psi, E \rangle \in C$, along with the corresponding pair $\langle \Psi, X \rangle \in A$. It is clear from the construction of $\pi(D)$ that for any LP-simple atomic value proposition L **after** $\overline{A}$, $[\![L$ **after** $\overline{A}]\!] \in X$ if and only if both $[\![L$ **after** $\overline{A}]\!] \in X$ and $\mathit{Reachable}([\overline{A}]) \in X$. We can conclude that $[\![L$ **after** $\overline{A}]\!] \in X$ if and only if $[L$ **after** $\overline{A}] \in E$. It follows by the Correspondence Theorem that for any LP-simple atomic value proposition V, V is true in Ψ if and only if $[\![V]\!] \in X$.

Assume that $\langle \Psi, X \rangle$ and $\langle \Psi', X' \rangle$ are distinct members of A. It follows that Ψ and Ψ' are distinct, and so must differ on the truth of some LP-simple atomic value proposition V. We can conclude that exactly one of the two sets X, X' includes $[\![V]\!]$. Thus $X \neq X'$. This shows that A captures a one-to-one correspondence between models of D and consistent answer sets of $\pi(D)$.

Now consider any LP-simple value proposition $V = V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$ and any $\langle \Psi, X \rangle \in A$. We know that V is true in Ψ if and only if at least one of $V_1, \ldots, V_m$ is true in Ψ or at least one of $V_{m+1}, \ldots, V_n$ is not true in Ψ. Since each of $V_1, \ldots, V_m$ is an LP-simple atomic value proposition, we can conclude that at least one of $V_1, \ldots, V_m$ is true in Ψ if and only if $\{ [\![V_1]\!], \ldots, [\![V_m]\!] \} \cap X$ is nonempty. Similarly, since each of $V_{m+1}, \ldots, V_n$ is an LP-simple atomic value proposition, we can conclude that at least one of $V_{m+1}, \ldots, V_n$ is not true in Ψ iff $\{ [\![V_{m+1}]\!], \ldots, [\![V_n]\!] \} \setminus X$ is nonempty. Summing up, we have shown that $V_1 \vee \cdots \vee$

$V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$ is true in Ψ iff at least one of the sets $\{ [\![V_1]\!], \ldots, [\![V_m]\!] \} \cap X$ and $\{ [\![V_{m+1}]\!], \ldots, [\![V_n]\!] \} \setminus X$ is nonempty. □

Proof of LP Correspondence Theorem. Let D be an LP-simple, qualification-free domain description. Let D' be the domain description obtained from D by deleting all value propositions. By the LP Correspondence Lemma, we know that there is a one-to-one correspondence between models of D' and consistent answer sets of $\pi(D')$ such that, for every model Ψ of D' and corresponding answer set X, an LP-simple value proposition $V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$ is true in Ψ if and only if either $\{ [\![V_1]\!], \ldots, [\![V_m]\!] \} \cap X \neq \emptyset$ or $\{ [\![V_{m+1}]\!], \ldots, [\![V_n]\!] \} \setminus X \neq \emptyset$.

Notice that $\pi(D) \setminus \pi(D')$ consists of all rules of the form

$$\mathit{False} \quad \leftarrow \quad [\![V_{m+1}]\!], \ldots, [\![V_n]\!], \mathit{not}\, [\![V_1]\!], \ldots, \mathit{not}\, [\![V_m]\!]$$

where $V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$ is an LP-simple value proposition in D. It is a consequence of this observation that, for any set X, X is an answer set for $\pi(D)$ if and only if X is an answer set for $\pi(D')$ such that, for every value proposition $V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$ in D, either $\{ [\![V_1]\!], \ldots, [\![V_m]\!] \} \cap X \neq \emptyset$ or $\{ [\![V_{m+1}]\!], \ldots, [\![V_n]\!] \} \setminus X \neq \emptyset$. We will rely on this result below.

Let A be the set of pairs $\langle \Psi, X \rangle$ such that Ψ is a model of D' and X is the corresponding answer set for $\pi(D')$. Since every model of D is a model of D' and similarly every consistent answer set for $\pi(D)$ is a consistent answer set for $\pi(D')$, we can complete our proof by showing that, for each pair $\langle \Psi, X \rangle$ that belongs to A, Ψ is a model of D if and only if X is a consistent answer set for $\pi(D)$. So consider any $\langle \Psi, X \rangle \in A$.

(Left-to-right) Assume that Ψ is a model of D. Thus, every value proposition in D is true in Ψ. It follows by the LP Correspondence Lemma that for every value proposition $V_1 \vee \cdots \vee V_m \vee \neg V_{m+1} \vee \cdots \vee \neg V_n$ in D, either $\{ [\![V_1]\!], \ldots, [\![V_m]\!] \} \cap X \neq \emptyset$ or $\{ [\![V_{m+1}]\!], \ldots, [\![V_n]\!] \} \setminus X \neq \emptyset$. We can conclude that X is an answer set for $\pi(D)$.

Proof in the other direction is similar. □

Next we sketch a proof of the LP Reachability Corollary. A complete proof would use the Splitting Set Theorem for logic programs [LT94].

Proof of LP Reachability Corollary (Sketch).
Let $B = \{\mathit{Reachable}([\overline{A}]) : \overline{A}$ *is an action string*$\}$. Since D is qualification-free, we can show that for every consistent answer set X for $\pi(D)$, $X \cap \mathit{Lit}(B) = B$. Given this, show that for any subset X of $\mathit{Lit}(U \setminus B)$, $X \cup B$ is a consistent answer set for $\pi(D)$ if and only if X is a consistent answer set for $\pi'(D)$. Finally, since every action string $\overline{A}$ belongs to $\mathit{Dom}(\Psi)$, we know that for every LP-simple atomic value proposition V, V is true in Ψ if and only if $\neg V$ is not true in Ψ. We can conclude that $\{ [\![V_1]\!], \ldots, [\![V_m]\!], \overline{[\![V_{m+1}]\!]}, \ldots, \overline{[\![V_n]\!]} \} \cap X \neq \emptyset$ if and only if either $\{ [\![V_1]\!], \ldots, [\![V_m]\!] \} \cap X \neq \emptyset$ or $\{ [\![V_{m+1}]\!], \ldots, [\![V_n]\!] \} \setminus X \neq \emptyset$. □

Finally we begin the proof of the Vivid Domains Theorem.

For any domain description D, let

$$\mathit{Rules}(D) = \left\{ \frac{\phi}{\psi} : \phi \textbf{ suffices for } \psi \in D \right\}.$$

We will need the following lemma.

Lemma 4.44 (Vivid Domains Lemma) *Let D be a vivid $\mathcal{AC}$ domain description. For any set X of fluent literals,*

$$\mathit{Cn}(X \cup \mathit{Rules}(D)) = \mathit{Cn}(X \cup \mathit{Rules}(\mathit{LP\text{-}Simple}(D))).$$

Proof. It is easy to verify that for any set X of fluent formulas we have

$$\mathit{Cn}(X \cup \mathit{Rules}(\mathit{LP\text{-}Simple}(D))) \subseteq \mathit{Cn}(X \cup \mathit{Rules}(D)).$$

For the other direction, let X be a set of fluent literals, and take

$$Y = \mathit{Cn}(X \cup \mathit{Rules}(\mathit{LP\text{-}Simple}(D))).$$

We will show that Y is closed under $X \cup \mathit{Rules}(D)$. We begin by observing that because X is a set of fluent literals and every rule in $\mathit{Rules}(\mathit{LP\text{-}Simple}(D))$ has either the form $\frac{\phi}{L}$ where L is a fluent literal or the form $\frac{\phi}{\mathit{False}}$, we can conclude that there is a set Y' of fluent literals such that $Y = \mathit{Cn}(Y')$. It's clear that Y is closed under X, so consider any rule $\frac{\phi}{\psi} \in \mathit{Rules}(D)$ such that $\phi \in Y$. We must show that ψ also belongs to Y. Since $\phi \in Y$ and $Y = \mathit{Cn}(Y')$ for some set Y' of fluent literals, we can conclude that some disjunct ϕ' of $\mathit{DNF}(\phi)$ belongs to Y. Since ϕ **suffices for** ψ is vivid, we know that ψ is either a nonempty conjunction of fluent literals or the formula *False*. If ψ is *False*, then Y is inconsistent and we're done; so assume that ψ is a nonempty conjunction of fluent literals. Consider any conjunct L of ψ. The rule ϕ' **suffices for** L belongs to $\mathit{LP\text{-}Simple}(D)$, and since $\phi' \in Y$, we have $L \in Y$. We can conclude that each conjunct of ψ belongs to Y; and since Y is logically closed, we have $\psi \in Y$. □

Notice that the preceding lemma establishes the fact that domain descriptions D and $\mathit{LP\text{-}Simple}(D)$ have the same set of states.

Proof of Vivid Domains Theorem. We have already observed that, for any structure Ψ, all of the value propositions in D are true in Ψ if and only all of the value propositions in $\mathit{LP\text{-}Simple}(D)$ are true in Ψ. By the Vivid Domains Lemma, we know that domains D and $\mathit{LP\text{-}Simple}(D)$ have the same set of states. Consider any action A and state S. We complete the first part of the proof by showing that a state S' may result from doing A in S in domain D if and only if S' may result from doing A in S in domain $\mathit{LP\text{-}Simple}(D)$.

It is easy to verify that A is prohibited in S in domain D if and only if A is prohibited in S in domain $\mathit{LP\text{-}Simple}(D)$. If A is prohibited in S in the two domains, we're done. So assume otherwise. It is also easy to see that the two domains agree on the set $F(A, S)$. Furthermore, although the two domains may not agree precisely on the set $E(A, S)$, it is clear that they do agree on $\mathit{Cn}(E(A, S))$. It

follows that E' is an explicit effect of A in S in domain $\mathit{LP\text{-}Simple}(D)$ if and only if there is an explicit effect E of A in S in domain D such that $E' = \mathit{Cn}(E) \cap \mathit{Lit}(\mathbf{F})$. Moreover, for any such E and E', it is clear that, for any set Γ of inference rules, $\mathit{Cn}(\Gamma \cup E) = \mathit{Cn}(\Gamma \cup E')$.

(Left-to-right) Assume that S' may result from doing A in S in domain D. So there is an explicit effect E of A in S in domain D such that

$$\mathit{Cn}(S') = \mathit{Cn}[\,(\,S \cap S' \cap \mathit{Lit}(\mathbf{F}_f)\,) \cup E \cup \mathit{Rules}(D)\,]\,.$$

We have already observed that there must be an explicit effect E' of A in S in domain $\mathit{LP\text{-}Simple}(D)$ such that $E' = \mathit{Cn}(E) \cap \mathit{Lit}(\mathbf{F})$ and, for any set Γ of inference rules, $\mathit{Cn}(\Gamma \cup E) = \mathit{Cn}(\Gamma \cup E')$. Thus

$$\mathit{Cn}(S') = \mathit{Cn}[\,(\,S \cap S' \cap \mathit{Lit}(\mathbf{F}_f)\,) \cup E' \cup \mathit{Rules}(D)\,]\,.$$

Furthermore, because $(\,S \cap S' \cap \mathit{Lit}(\mathbf{F}_f)\,) \cup E'$ is a set of fluent literals, we can conclude by the Vivid Domains Lemma that

$$\mathit{Cn}(S') = \mathit{Cn}[\,(\,S \cap S' \cap \mathit{Lit}(\mathbf{F}_f)\,) \cup E' \cup \mathit{Rules}(\mathit{LP\text{-}Simple}(D))\,]\,.$$

That is, S' may result from doing A in S in domain $\mathit{LP\text{-}Simple}(D)$.

Proof in the other direction is similar. Thus we have shown that the two domains agree on *Res*, which is sufficient to establish the fact that they have the same models, given the earlier observation that they have equivalent sets of value propositions (Section 3.6). Moreover, since they also agree, for each action A and state S, on the question of whether or not A is prohibited in S, we can conclude that either both domain descriptions are qualification-free or neither is. □

Chapter 5

A Logic of Universal Causation

This chapter discusses a modal nonmonotonic logic of "universal causation," called UCL, designed for formalizing commonsense knowledge about actions. It was introduced in [Tur98]. UCL extends the recently introduced causal theories formalism of McCain and Turner [MT97], which shares its underlying motivations.

The mathematical ideas underlying UCL, and the approach to formalizing actions in it, can also be understood as an outgrowth of the work described in the first part of the dissertation.

5.1 Introduction

The fundamental distinction in UCL—between facts that are caused and facts that are merely true—is expressed by means of the modal operator C, read as "caused." For example, one can write $\phi \supset \mathsf{C}\psi$ to say that ψ is caused whenever ϕ is true. These simple linguistic resources make it possible for a UCL theory to express the conditions under which facts are caused. It is in this sense that UCL is a logic of causation.

As usual for nonmonotonic logics, the main semantic definition in UCL—of

a "causally explained" interpretation—is given by a fixpoint condition. Intuitively, an interpretation is causally explained by a UCL theory T if it represents the facts true in a world that is "causally possible" according to T. The focus in UCL on causally possible worlds is motivated by the following pair of observations.

- Knowledge of the causally possible worlds is sufficient for many commonsense reasoning tasks associated with action domains, such as prediction and planning.
- In order to determine the causally possible worlds, it is sufficient to know the conditions under which facts are caused.

The first observation suggests that UCL can be useful. The second observation helps explain why UCL can be simple: it formalizes causal knowledge of a relatively simple kind, and does not attempt the notoriously difficult task of formalizing causal relations of the form "ϕ causes ψ." Happily, one can describe and reason about the conditions under which facts are caused without settling the question of what causes what.

In UCL, the notion of a causally possible world is made precise on the basis of the following pair of assumptions.

- In a causally possible world, every fact that is caused obtains.
- In a causally possible world, every fact that obtains is caused.

The first assumption is unremarkable. The second is not. As in [MT97], we call it the *principle of universal causation*. This simplifying assumption is the key to the main semantic definition of the logic, which is therefore named for it. We take these two assumptions together to define what it is for a world to be causally possible: what obtains in the world is exactly what is caused in it. Accordingly, the main semantic definition in UCL says that an interpretation I is causally explained by a UCL theory T if what is true in I is exactly what is caused in I according to T.

The principle of universal causation is easily relaxed in practice. For instance, when describing action domains, we generally have little or nothing to say about the "actual" conditions under which facts in the initial situation are caused. Instead, our UCL action theories typically stipulate that facts in the initial situation are caused. Such stipulations are straightforward in UCL, where one can say that ϕ is caused whenever it is true simply by writing

$$\phi \supset \mathsf{C}\phi\,. \tag{5.1}$$

In the same way, we typically stipulate that facts about the occurrence and non-occurrence of actions are caused.

More interesting are those facts that, roughly speaking, are true simply because they were true before and haven't been made false since. It is facts of this kind that give rise to the frame problem [MH69]. Solutions to the frame problem typically appeal to the "commonsense law of inertia," according to which the value of an inertial fluent persists unless it is caused to change. The principle of universal causation makes possible a simple, robust encoding of the commonsense law of inertia. Take f_t to stand for the proposition that a fluent f holds at a time t. One can write

$$\mathsf{C}f_t \wedge f_{t+1} \supset \mathsf{C}f_{t+1} \tag{5.2}$$

to stipulate that f is caused at time $t+1$ whenever it is caused at time t and continues to hold at time $t+1$. Thus, axioms of form (5.2) can, in effect, suspend the principle of universal causation with respect to persistent inertial fluents. Of course, universal causation still requires that if f does not persist, the new value of f must be caused. That is, the UCL theory must describe conditions sufficient for it to be caused. In this way, inertia axioms of form (5.2) interact with the principle of universal causation to solve the frame problem, guaranteeing that inertial fluents persist unless they are caused to change.

UCL differs in fundamental ways from nonmonotonic formalisms such as default logic [Rei80] and autoepistemic logic [Moo85]. First, UCL is not motivated by the problem of general default reasoning and knowledge representation. It is designed for a more specific purpose. Second, in UCL one describes the conditions under which facts are caused, rather than the conditions under which facts are believed or known. Third, the fixpoint condition in UCL characterizes complete worlds, in the form of classical interpretations, rather than incomplete, logically closed belief sets. Nonetheless, we will see that UCL is closely related to default logic, in the special case when we consider only the "complete," consistent extensions of default theories. We will also consider briefly a rather striking similarity between the main semantic definitions of UCL and autoepistemic logic.

UCL can be understood as an extension of nonmonotonic formalism of causal theories introduced in [MT97], in which there are so-called "causal laws" of the form $\phi \Rightarrow \psi$, with the intended reading "whenever ϕ is true, ψ is caused to be true." Here we show that such causal laws can be translated in UCL as

$$\phi \supset \mathsf{C}\psi \tag{5.3}$$

thus providing a more adequate semantic account of them. We go on to develop in some detail the close relationship between such causal laws and the circumscriptive approach to "causal laws" of Lin [Lin95]. Along the same lines, we show that the "static causal laws" of [MT95b] correspond to UCL formulas of the form

$$\mathsf{C}\phi \supset \mathsf{C}\psi\,. \tag{5.4}$$

The contributions of this chapter can be summarized as follows. It introduces UCL, a mathematically simple modal nonmonotonic logic designed for representing commonsense knowledge about actions. By establishing relationships with previous proposals, including the proposal studied in the first part of this dissertation, it shows how a variety of causal theories of action can be expressed in UCL. By

relating these proposals to a single logical framework, it contributes to the ongoing investigation of the relationships between various approaches. Finally, it relates UCL to some well-known nonmonotonic formalisms.

We proceed as follows. Section 5.2 defines propositional UCL, the fragment primarily investigated in this dissertation. Section 5.3 shows how the inference rules used in the first part of the dissertation are captured in UCL, and also embeds rule update (Section 3.3.2) in UCL. In doing so, we see how the commonsense law of inertia can be expressed in UCL. Section 5.4 relates UCL to default logic, and in Section 5.5 we observe that the embedding of $\mathcal{AC}$ in default logic from Chapter 3 yields a similar embedding in UCL. We also consider how to simplify the embedding somewhat, in anticipation of subsequent work. Section 5.6 introduces subclasses of UCL for which simpler semantic characterizations can be given. Flat UCL theories correspond to the causal theories formalism of [MT97], and definite UCL theories correspond to the definite causal theories of [MT97, MT98b]. Definite theories are particularly important from a computational perspective, since they have a concise translation into classical logic, called "literal completion." Section 5.7 describes a general method of formalizing action domains based on the approach from [MT97]. Here we leave behind the situation calculus and move to natural number time. Section 5.8 shows that a subset of UCL can be nicely reduced to circumscriptive theories, and Section 5.9 explores the relationship between UCL and the circumscriptive action theories of Lin [Lin95, Lin96]. In Section 5.10, we briefly consider the relationship of UCL to autoepistemic logic. In Section 5.11, we extend UCL to allow first and second-order quantifiers. In Section 5.12, we show that (second-order) UCL extends the second-order subset of the nonpropositional causal theories of Lifschitz [Lif97], which, in turn, extend the flat propositional UCL theories of Section 5.6.

5.2 Propositional UCL

5.2.1 Syntax and Semantics

Begin with a set of propositional symbols (atoms)—the signature of our language. For convenience, we assume that the language includes a zero-place logical connective *True* such that *True* is a tautology. Let *False* stand for $\neg$*True*. A *literal* is an atom or its negation. We identify each interpretation with the set of literals true in it. UCL *formulas* are defined as usual for a modal propositional language with single unary modal operator C. A formula is *nonmodal* if C does not occur in it. A UCL *theory* is a set of UCL formulas.

The main semantic definition (of a "causally explained" interpretation) is obtained by imposing a fixpoint condition on S5 modal logic. Thus, a UCL *structure* is a pair (I, S) such that I is an interpretation, and S is a set of interpretations to which I belongs. The truth of a UCL sentence in a UCL structure is defined by the standard recursions over the propositional connectives, plus the following two conditions.

$$(I, S) \models p \quad \text{iff} \quad I \models p \quad \text{(for any atom } p\text{)}$$
$$(I, S) \models \mathsf{C}\phi \quad \text{iff} \quad \text{for all } I' \in S,\ (I', S) \models \phi$$

Given a UCL theory T, we write $(I, S) \models T$ to mean that $(I, S) \models \phi$, for every $\phi \in T$. In this case, we say that (I, S) is a *model* of T. We also say that (I, S) is an *I-model* of T, emphasizing the distinguished interpretation I.

Main Definition. Let T be a UCL theory. An interpretation I is *causally explained* by T if $(I, \{I\})$ is the unique I-model of T.

We distinguish three entailment relations. The first two—classical propositional entailment and propositional S5 entailment—are standard, monotonic relations. The third—UCL entailment—is defined as follows. For any UCL theory T and nonmodal formula ϕ, we write $T \mid\!\sim \phi$ to say that ϕ is true in every interpretation

causally explained by T.

Here is an alternative characterization of causally explained interpretations.

Proposition 5.1 *For any UCL theory T, an interpretation I is causally explained by T if and only if $(I, \{I\})$ is the unique model of $T \cup I$.*

Proof. Clearly T and $T \cup I$ have the same I-models. It remains only to observe that all models of $T \cup I$ are I-models. □

5.2.2 Examples

Let T_1 be the UCL theory with one formula

$$p \supset \mathsf{C}p \tag{5.5}$$

in the language with a single atom p. Let I_1 be the interpretation $\{p\}$. The structure $(I_1, \{I_1\})$ is the unique I_1-model of T_1, so I_1 is causally explained by T. The only other interpretation is $I_2 = \{\neg p\}$. Since $(I_2, \{I_1, I_2\}) \models T_1$, I_2 is not causally explained by T_1. Therefore, $T_1 \mid\!\sim p$.

Notice that it is essential that the language of T_1 include only the atom p. If it were extended to include a second atom, there would no longer be any causally explained interpretations.

Let T_2 be the UCL theory obtained by adding to T_1 the formula

$$\neg p \supset \mathsf{C}\neg p\,. \tag{5.6}$$

Both I_1 and I_2 are causally explained by T_2. Therefore, $T_2 \not\mid\!\sim p$, which illustrates the nonmonotonicity of UCL.

Let T_3 be the UCL theory obtained from T_2 by adding a single atom q to the language, and also adding the formula

$$\mathsf{C}(q \equiv p)\,. \tag{5.7}$$

The interpretations $\{p, q\}$ and $\{\neg p, \neg q\}$ are both causally explained by T_3. No others are.

This last example illustrates the following general phenomenon. We obtain a *definitional extension* T' of a UCL theory T by adding a new atom p to the signature and also adding an *explicit definition* of p—a formula of the form

$$\mathsf{C}\,(p \equiv \phi) \tag{5.8}$$

where ϕ is a nonmodal formula in which p does not occur. Clearly, one can replace any formula equivalent to ϕ by p anywhere in T', except in (5.8), without affecting the models of T', or, therefore, the causally explained interpretations. Moreover, it is not difficult to verify that T' is a *conservative extension* of T: that is, T and T' have the same UCL-consequences (and, in fact, the same S5-consequences) in the language of T.

5.3 Possible Next States and Inertia in UCL

In this chapter we embed rule update, defined in Section 3.3.2, in UCL. In this manner we obtain a more traditional semantic account of the use of inference rules both in rule update and in the definition of possible next states in the action language $\mathcal{AC}$.

5.3.1 Inference Rules in UCL

We first make precise the simple relationship between an inference rule

$$\frac{\phi}{\psi}$$

and the corresponding UCL formula

$$\mathsf{C}\phi \supset \mathsf{C}\psi\,.$$

We begin with three preliminary definitions and an easy lemma.

Given a set $\mathcal{R}$ of inference rules, let

$$CE(\mathcal{R}) = \left\{ \mathsf{C}\phi \supset \mathsf{C}\psi : \frac{\phi}{\psi} \in \mathcal{R} \right\}.$$

Given a set S of interpretations, let $Th(S)$ denote the set of nonmodal formulas true in all members of S. Given a set Γ of nonmodal formulas, let $Mod(\Gamma)$ denote the set of interpretations that satisfy all members of Γ.

Lemma 5.2 *Let $\mathcal{R}$ be a set of inference rules, S a set of interpretations, and I an interpretation in S. $Th(S)$ is closed under $\mathcal{R}$ if and only if $(I, S) \models CE(\mathcal{R})$.*

Proof. Assume $Th(S)$ is closed under $\mathcal{R}$. Consider any formula $\mathsf{C}\phi \supset \mathsf{C}\psi$ from $CE(\mathcal{R})$ such that $(I, S) \models \mathsf{C}\phi$. It follows that $\phi \in Th(S)$. By the definition of $CE(\mathcal{R})$, we know that $\frac{\phi}{\psi} \in \mathcal{R}$; and since $Th(S)$ is closed under $\mathcal{R}$, we can conclude that $\psi \in Th(S)$. It follows that $(I, S) \models \mathsf{C}\psi$.

Proof in the other direction is similar. □

This lemma yields the following characterization of the relationship between a set $\mathcal{R}$ of inference rules and the corresponding set $CE(\mathcal{R})$ of UCL formulas.

Proposition 5.3 *For any set $\mathcal{R}$ of inference rules and any nonmodal formula ϕ, $\phi \in Cn(\mathcal{R})$ if and only if $\mathsf{C}\phi$ is an S5-consequence of $CE(\mathcal{R})$.*

Proof. In this proof we use the fact that $\phi \in Cn(\mathcal{R})$ iff ϕ belongs to every set of formulas that is both closed under propositional logic and closed under $\mathcal{R}$.

(Left-to-right) Assume that $\phi \in Cn(\mathcal{R})$. Consider any model (I, S) of $CE(\mathcal{R})$. By the previous lemma, $Th(S)$ is closed under $\mathcal{R}$. It follows that $\phi \in Th(S)$, and thus that $(I, S) \models \mathsf{C}\phi$.

(Right-to-left) Assume that $\phi \notin Cn(\mathcal{R})$. Thus there is a nonempty set S of interpretations such that $Th(S)$ is closed under $\mathcal{R}$ and $\phi \notin Th(S)$. Consider any $I \in S$. By the previous lemma, $(I, S) \models CE(\mathcal{R})$. Since $\phi \notin Th(S)$, $(I, S) \not\models \mathsf{C}\phi$. □

The causally explained interpretations of a UCL theory $CE(\mathcal{R})$ can be characterized as follows, simplifying the more general, similar result of Proposition 5.1.

Proposition 5.4 *For any set $\mathcal{R}$ of inference rules, an interpretation I is causally explained by $CE(\mathcal{R})$ if and only if $(I, \{I\})$ is the unique model of $CE(\mathcal{R})$.*

The proof begins with an observation about inference rules. If sets X and Y of formulas are both closed under a set $\mathcal{R}$ of inference rules, then $X \cap Y$ is also closed under $\mathcal{R}$. This fact, along with Lemma 5.2, yields the following.

Lemma 5.5 *For any set $\mathcal{R}$ of inference rules, if $(I, S) \models CE(\mathcal{R})$ and $(I', S') \models CE(\mathcal{R})$, then $(I, S \cup S') \models CE(\mathcal{R})$.*

Proof. By Lemma 5.2, both $Th(S)$ and $Th(S')$ are closed under $\mathcal{R}$. It follows, as observed above, that $Th(S) \cap Th(S')$ is also closed under $\mathcal{R}$. Since I belongs to S and $Th(S) \cap Th(S') = Th(S \cup S')$, we conclude by Lemma 5.5 that $(I, S \cup S') \models CE(\mathcal{R})$. □

Proof of Proposition 5.4. The right-to-left direction is trivial. For the other direction, assume I is causally explained by $CE(\mathcal{R})$. So $(I, \{I\})$ is the unique I-model of $CE(\mathcal{R})$. Assume that $(I', S) \models CE(\mathcal{R})$. By Lemma 5.5, $(I, \{I\} \cup S) \models CE(\mathcal{R})$. Since $(I, \{I\})$ is the unique I-model of $CE(\mathcal{R})$, $S = \{I\}$ and $I' = I$. □

If a UCL theory T has a unique model $(I, \{I\})$, then, for every nonmodal formula ϕ, $\mathsf{C}\phi$ is among the S5-consequences of T if and only if $I \models \phi$. Given this observation, the previous proposition yields the following corollary.

Corollary 5.6 *For any set $\mathcal{R}$ of inference rules, an interpretation I is causally explained by $CE(\mathcal{R})$ if and only if $Cn(I) = \{\phi : \mathsf{C}\phi$ is an S5-consequence of $CE(\mathcal{R})\}$, where ϕ ranges over nonmodal formulas.*

The fixpoint condition in Corollary 5.6 is very similar to that used in the definition of rule update, which we consider next.

5.3.2 Two Embeddings of Rule Update in UCL

First we recall the definition of rule update from Section 3.3.2.

Let $\mathcal{R}$ be a set of inference rules, and I an interpretation. An interpretation I' is a rule update of I by $\mathcal{R}$ if and only if

$$Cn(I') = Cn((I \cap I') \cup \mathcal{R}).$$

Here is a first, easy embedding of rule update in UCL.

Proposition 5.7 *Let $\mathcal{R}$ be a set of inference rules, and I and I' interpretations. Take*

$$T(\mathcal{R}, I, I') = CE(\mathcal{R}) \cup \{ \mathsf{C}L : L \in I \cap I' \}.$$

I' is a rule update of I by $\mathcal{R}$ if and only if I' is causally explained by $T(\mathcal{R}, I, I')$.

Proof. I' is a rule update of I by $\mathcal{R}$ iff $Cn(I') = Cn((I \cap I') \cup \mathcal{R})$, which, by Proposition 5.3, is equivalent to

$$Cn(I') = \{\phi : \mathsf{C}\phi \text{ is an S5-consequence of } CE((I \cap I') \cup \mathcal{R})\}$$

which, by Corollary 5.6, is true iff I' is causally explained by $CE((I \cap I') \cup \mathcal{R})$. It remains only to notice that $T(\mathcal{R}, I, I')$ is S5-equivalent to $CE((I \cap I') \cup \mathcal{R})$. □

We can improve the embedding of rule update in UCL by writing formulas that capture the commonsense law of inertia, which is built into the definition of rule update. We establish one way to do this in the following theorem. This embedding is similar to the embedding into default logic of the definition of possible next states from $\mathcal{AC}$ that was presented in Chapter 3.5.2.

Proposition 5.8 *Let $\mathcal{R}$ be a set of inference rules, and I an interpretation. Take*

$$T'(\mathcal{R}, I) = CE(\mathcal{R}) \cup \{ L \supset \mathsf{C}L : L \in I \}.$$

An interpretation I' is a rule update of I by $\mathcal{R}$ iff I' is causally explained by $T'(\mathcal{R}, I)$.

Proof. In light of Proposition 5.7, it is enough to show that I' is causally explained by $T'(\mathcal{R}, I)$ iff I' is causally explained by $T(\mathcal{R}, I, I')$. This follows easily from the observation that, for every superset S of I',

$$(I', S) \models T(\mathcal{R}, I, I') \text{ iff } (I', S) \models T(\mathcal{R}, I)$$

which, in turn, follows from the easily verified observation that

$$(I', S) \models \{\mathsf{C}L : L \in I \cap I'\} \text{ iff } (I', S) \models \{ L \supset \mathsf{C}L : L \in I \}.$$

□

For example, recalling the rule update example from Section 3.3.2, we see that $T'(\mathcal{R}_1, I_1)$ is the following UCL theory.

$$\mathsf{C}a \supset \mathsf{C}(\neg b \vee \neg c)$$
$$a \supset \mathsf{C}a$$
$$b \supset \mathsf{C}b$$
$$c \supset \mathsf{C}c$$

It is easy to check that $T'(\mathcal{R}_1, I_1)$ has exactly two causally explained interpretations—$\{a, \neg b, c\}$ and $\{a, b, \neg c\}$—which are the two rule updates of I_1 by $\mathcal{R}_1$.

In anticipation of our interest in UCL theories for commonsense reasoning about action, it will be helpful to consider a third, somewhat more complex embedding of rule update in UCL, which features a more explicit representation of the commonsense law of inertia.

5.3.3 A Third Embedding: Commonsense Inertia in UCL

In order to obtain a more explicit embedding of rule update in UCL, we first extend the language by adding a fresh atom A_0 for every atom A. For any literal L in the original language, let L_0 be the formula obtained by replacing the atom A that

occurs in L with the atom A_0. Given an interpretation I of the original language, let $I_0 = \{L_0 : L \in I\}$.

Given a set $\mathcal{R}$ of inference rules, the UCL theory $T''(\mathcal{R})$ is the union of the following three sets of UCL formulas.

$$CE(\mathcal{R}) \tag{5.9}$$

$$\{\, \mathsf{C}L_0 \wedge L \supset \mathsf{C}L \,:\, L \text{ is a literal in the original language} \,\} \tag{5.10}$$

$$\{\, L_0 \supset \mathsf{C}L_0 \,:\, L \text{ is a literal in the original language} \,\} \tag{5.11}$$

Proposition 5.9 *Let $\mathcal{R}$ be a set of inference rules, and I an interpretation. An interpretation I' is a rule update of I by $\mathcal{R}$ if and only if $I_0 \cup I'$ is causally explained by $T''(\mathcal{R})$.*

Proof Sketch. By Proposition 5.8, it is enough to show that I' is causally explained by $T'(\mathcal{R}, I)$ iff $I_0 \cup I'$ is causally explained by $T''(\mathcal{R})$. The key observation is that I' is causally explained by $T'(\mathcal{R}, I)$ iff $I_0 \cup I'$ is causally explained by $\{\mathsf{C}L_0 : L \in I\} \cup T'(\mathcal{R}, I)$. Thus, we need to show that

$$(I_0 \cup I', S) \models \{\mathsf{C}L_0 : L \in I\} \cup CE(R) \cup \{L \supset \mathsf{C}L : L \in I\}$$

iff

$$(I_0 \cup I', S) \models T''(\mathcal{R}),$$

which follows from the fact that $(I_0 \cup I', S)$ satisfies the formulas in (5.10) and (5.11) iff $(I_0 \cup I', S) \models \{\mathsf{C}L_0 : L \in I\} \cup \{L \supset \mathsf{C}L : L \in I\}$. □

In this alternative embedding, unlike the previous one, we do not include an implicit encoding of any particular initial interpretation I. Instead we say of every literal L that it is caused initially if it is true initially (5.11). That is roughly to say that we require no additional causal explanation for literals in the initial situation

beyond the fact that they are true. (Or, more accurately, we simply say that they are caused whenever they are true.)[1]

Given this, we can understand the formulas in (5.10) as an explicit representation of the causal understanding of the commonsense law of inertia that motivates the definition of rule update. Each rule in (5.10) says of a literal L that if it is caused initially, then it is also caused after the update whenever it is true after the update. And since every fact in a causally possible world must be caused according to our theory, we know that whenever a fluent changes its value in a causally possible world, the new value must have a causal explanation other than inertia. That is, the formulas $CE(\mathcal{R})$, in concert with the formulas (5.10), must describe conditions sufficient for it to be caused. We have often described the commonsense law of inertia in terms of this second observation—saying "things change only when made to."

The UCL formula for inertia in this second embedding of rule update is very similar to the default rules for inertia used in the translation of $\mathcal{AC}$ into default logic. It is interesting to note that Proposition 5.9 continues to hold if we replace the inertia formulas (5.10) with the stronger UCL formulas

$$\{\, L_0 \wedge L \supset \mathsf{C}L \,:\, L \text{ is a literal in the original language} \,\}. \tag{5.12}$$

We will find this sort of fact quite useful. Formula (5.12) is what we will eventually call a "definite" formula, while formula (5.10) is not. We will see that theories in which all formulas are definite have nice mathematical properties, leading to convenient methods for automated reasoning.

[1] Recall that essentially the same idea was used in the translations of $\mathcal{AC}$ in the first part of the dissertation.

5.4 UCL and Default Logic

In this section, we establish the close mathematical relationship between UCL and default logic [Rei80]. More precisely, to be more general, we consider an elaboration of default logic, called disjunctive default logic [GLPT91], which includes Reiter's default logic as a special case. The semantics of a disjunctive default theory is given in terms of its extensions, which are logically closed sets of (nonmodal) formulas that satisfy a certain fixpoint condition. Although an extension may be inconsistent, or incomplete (that is, there may be an atom p such that neither p nor $\neg p$ belong to it), we will be interested in the special case of extensions that are both consistent and complete, since it is these extensions that correspond to interpretations.

We will specify a translation from disjunctive default logic to UCL such that the complete, consistent extensions correspond to the causally explained interpretations. The translation is invertible, so there is a strong sense in which UCL is equivalent to disjunctive default logic, restricted to the special case of complete, consistent extensions.

5.4.1 Review of Disjunctive Default Logic

Here we recall definitions from [GLPT91].

A *disjunctive default rule* is an expression of the form

$$\frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma_1 | \cdots | \gamma_n} \tag{5.13}$$

where all of $\alpha, \beta_1, \ldots, \beta_m, \gamma_1, \ldots, \gamma_n$ are (nonmodal) formulas ($m \geq 0, n \geq 1$).

A *disjunctive default theory* is a set of disjunctive default rules. Let D be a disjunctive default theory and E a set of formulas. Define D^E as follows.

$$D^E = \left\{ \frac{\alpha}{\gamma_1 | \cdots | \gamma_n} : \frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma_1 | \cdots | \gamma_n} \in D \text{ and for all } i\,(1 \leq i \leq m),\ \neg\beta_i \notin E \right\}$$

A set E' of formulas is *closed under* D^E if, for every member of D^E, if $\alpha \in E'$ then at least one of $\gamma_1, \ldots, \gamma_n \in E'$. We say E is an *extension* for D if E is a minimal set

closed under propositional logic and closed under D^E. We say E is *complete* if, for every atom p, either $p \in E$ or $\neg p \in E$. Notice that, for the purpose of computing complete extensions, the prerequisites $\beta_1, \ldots, \beta_m$ of a disjunctive default rule can be safely replaced with their conjunction.

Reiter's default logic corresponds to the special case when $n = 1$.

5.4.2 UCL and Disjunctive Default Logic

Given a disjunctive default theory D, let $ucl(D)$ be the UCL theory obtained from D by replacing each disjunctive default rule (5.13) with the UCL formula

$$\mathsf{C}\alpha \wedge \beta_1 \wedge \cdots \wedge \beta_m \supset \mathsf{C}\gamma_1 \vee \cdots \vee \mathsf{C}\gamma_n\,. \tag{5.14}$$

It is a fact of propositional S5 modal logic that every theory is equivalent to one in which every formula has the form (5.14), with $m = 1$.[2] Thus, every UCL theory is equivalent to one that can be obtained by this translation from disjunctive default logic.

Theorem 5.10 *For any disjunctive default theory D and interpretation I, $Th(\{I\})$ is an extension for D if and only if I is causally explained by $ucl(D)$.*

Lemma 5.11 *For any disjunctive default theory D and any UCL structure (I, S), $(I, S) \models ucl(D)$ if and only if $Th(S)$ is closed under $D^{Th(\{I\})}$.*

Proof. Assume $(I, S) \models ucl(D)$. Consider any rule $\frac{\alpha}{\gamma_1 | \cdots | \gamma_n}$ in $D^{Th(\{I\})}$ such that $\alpha \in Th(S)$. We must show that at least one of $\gamma_1, \ldots, \gamma_n$ is in $Th(S)$. We know there is a rule $\frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma_1 | \cdots | \gamma_n}$ in D such that I satisfies all of $\beta_1, \ldots, \beta_m$. It follows that $\mathsf{C}\alpha \wedge \beta_1 \wedge \cdots \wedge \beta_m \supset \mathsf{C}\gamma_1 \vee \cdots \vee \mathsf{C}\gamma_n$ is in $ucl(D)$, and that $(I, S) \models \beta_1 \wedge \cdots \wedge \beta_m$. Since $\alpha \in Th(S)$, $(I, S) \models \mathsf{C}\alpha$. And since $(I, S) \models ucl(D)$, we can conclude that

[2] This follows, for instance, from the MCNF Theorem in [HC68].

$(I, S) \models \mathsf{C}\gamma_1 \vee \cdots \vee \mathsf{C}\gamma_n$. Thus, for at least one of γ_i $(1 \leq i \leq n)$, $(I, S) \models \mathsf{C}\gamma_i$, and consequently $\gamma_i \in Th(S)$. Proof in the other direction is similar. □

Proof of Theorem 5.10. ($\Longrightarrow$) Assume that $Th(\{I\})$ is an extension for D. We know by Lemma 5.11 that $(I, \{I\}) \models ucl(D)$. Let S be a superset of $\{I\}$ such that $(I, S) \models ucl(D)$. By Lemma 5.11, $Th(S)$ is closed under $D^{Th(\{I\})}$. Since $Th(\{I\})$ is a minimal among sets closed under $D^{Th(\{I\})}$, and $Th(S) \subseteq Th(\{I\})$, we have $Th(S) = Th(\{I\})$. It follows that $S = \{I\}$. So $(I, \{I\})$ is the unique I-model of $ucl(D)$. That is, I is causally explained by $ucl(D)$.

($\Longleftarrow$) Assume I is causally explained by $ucl(D)$. So $(I, \{I\}) \models ucl(D)$. By Lemma 5.11, $Th(\{I\})$ is closed under $D^{Th(\{I\})}$. Let E be a subset of $Th(\{I\})$ closed under propositional logic and under $D^{Th(\{I\})}$. By Lemma 5.11, $(I, Mod(E)) \models ucl(D)$. Since $(I, \{I\})$ is the unique I-model of $ucl(D)$, we have $Mod(E) = \{I\}$. It follows that $E = Th(\{I\})$. We can conclude that $Th(\{I\})$ is a minimal set closed under propositional logic and under $D^{Th(\{I\})}$. That is, $Th(\{I\})$ is an extension for D. □

In the statement of Theorem 5.10, we restrict attention to extensions that can be expressed in the form $Th(I)$, where I is an interpretation. That is, we consider only complete, consistent extensions. The restriction to complete extensions can of course be expressed in the default theory itself, simply by adding the default rule

$$\frac{: p, \neg p}{False}$$

for each atom p in the language.

5.5 Embedding $\mathcal{AC}$ in UCL

Recall that the Reachability Corollary in the first part of the dissertation shows that qualification-free $\mathcal{AC}$ domain descriptions that do not include executability propositions can be embedded in default logic (via the translation δ'). This embedding

yields a one-to-one correspondence between $\mathcal{AC}$ models and consistent default extensions, and, moreover, one can easily verify that all such default extensions are complete. In light of these observations, we obtain the following embedding theorem as an immediate consequence of Theorem 5.10 and the Reachability Corollary (Corollary 3.12).

Theorem 5.12 *Let D be a qualification-free $\mathcal{AC}$ domain description with no executability propositions. There is a one-to-one correspondence between models of D and the interpretations causally explained by $ucl(\delta'(D))$ such that, for every model Ψ of D and its corresponding causally explained interpretation I, a value proposition V is true in Ψ if and only if $I \models [\![V]\!]$.*

In this manner we obtain the UCL formalization of Example 1 (from Section 3.2) that appears in Figure 5.1 simply by translating the default theory from Figure 3.1. The first two UCL formulas in this translation are unnecessarily awkward. They are easily simplified though, in light of the following two facts. First, the formula $\mathsf{C}\phi \supset \mathsf{C}False$ is S5-equivalent to $\neg\mathsf{C}\phi$. Second, the following proposition holds.

Proposition 5.13 *For any UCL theory T and nonmodal formula ϕ, adding $\neg\mathsf{C}\phi$ to T simply eliminates the causally explained interpretations that satisfy ϕ*

Proof. Consider two cases.

Case 1: $I \not\models \phi$. Thus, for any superset S of $\{I\}$, $(I, S) \models \neg\mathsf{C}\phi$. So $(I, S) \models T$ iff $(I, S) \models T \cup \{\neg\mathsf{C}\phi\}$. Hence, adding $\neg\mathsf{C}\phi$ to T does not affect the causally explained interpretations that don't satisfy ϕ.

Case 2: $I \models \phi$. Thus, $(I, \{I\}) \not\models \neg\mathsf{C}\phi$, and so $(I, \{I\}) \not\models T \cup \{\neg\mathsf{C}\phi\}$, which shows that adding $\neg\mathsf{C}\phi$ to T eliminates any causally explained interpretations that satisfy ϕ. □

$$\mathsf{C}Holds(Trotting, S_0) \supset \mathsf{C}False$$
$$\mathsf{C}(\neg(Holds(Loaded(Gun_1), S_0) \vee Holds(Loaded(Gun_2), S_0))) \supset \mathsf{C}False$$
$$\mathsf{C}\neg Holds(Alive, s) \supset \mathsf{C}\neg Holds(Trotting, s)$$
$$\mathsf{C}Holds(Loaded(x), s) \supset \mathsf{C}\neg Holds(Alive, Result(Shoot(x), s))$$
$$Holds(f, S_0) \supset Holds(f, S_0) \qquad \neg Holds(f, S_0) \supset \neg Holds(f, S_0)$$
$$\mathsf{C}Holds(f, s) \wedge Holds(f, Result(a, s)) \supset \mathsf{C}Holds(f, Result(a, s))$$
$$\mathsf{C}\neg Holds(f, s) \wedge \neg Holds(f, Result(a, s)) \supset \mathsf{C}\neg Holds(f, Result(a, s))$$

Figure 5.1: UCL translation of default theory for Example 1.

$$Holds(Trotting, S_0)$$
$$Holds(Loaded(Gun_1), S_0) \vee Holds(Loaded(Gun_2), S_0)))$$
$$\mathsf{C}\neg Holds(Alive, s) \supset \mathsf{C}\neg Holds(Trotting, s)$$
$$\mathsf{C}Holds(Loaded(x), s) \supset \mathsf{C}\neg Holds(Alive, Result(Shoot(x), s))$$
$$Holds(f, S_0) \supset Holds(f, S_0) \qquad \neg Holds(f, S_0) \supset \neg Holds(f, S_0)$$
$$\mathsf{C}Holds(f, s) \wedge Holds(f, Result(a, s)) \supset \mathsf{C}Holds(f, Result(a, s))$$
$$\mathsf{C}\neg Holds(f, s) \wedge \neg Holds(f, Result(a, s)) \supset \mathsf{C}\neg Holds(f, Result(a, s))$$

Figure 5.2: Simpler UCL theory for Example 1.

Of course adding the formula $\neg\phi$ to a UCL theory has precisely the same effect on the causally explained interpretations: it simply eliminates those that satisfy ϕ. Hence, we have shown that one can replace $\mathsf{C}\phi \supset \mathsf{C}False$ with $\neg\phi$ without affecting causally explained interpretations.

These observations, and some propositional logic, show that the UCL theory in Figure 5.1 has exactly the same causally explained interpretations as the simpler theory shown in Figure 5.2. This approach also yields the UCL theories shown in Figures 5.3 and 5.4 for Examples 2 and 3 from Section 3.2. (In the third formula in Figure 5.3, we have also used the fact that $\mathsf{C}\phi \wedge \mathsf{C}\psi$ is S5-equivalent to $\mathsf{C}(\phi \wedge \psi)$.)

$$\neg Holds(Open, S_0)$$
$$\mathsf{C}\neg Holds(Fastened(x), Result(Unfasten(x), s))$$
$$\mathsf{C}\neg Holds(Fastened(Clasp_1), s) \wedge \mathsf{C}\neg Holds(Fastened(Clasp_2), s) \supset \mathsf{C}Holds(Open, s)$$
$$Holds(f, S_0) \supset \mathsf{C}Holds(f, S_0) \qquad \neg Holds(f, S_0) \supset \mathsf{C}\neg Holds(f, S_0)$$
$$\mathsf{C}Holds(f, s) \wedge Holds(f, Result(a, s)) \supset \mathsf{C}Holds(f, Result(a, s))$$
$$\mathsf{C}\neg Holds(f, s) \wedge \neg Holds(f, Result(a, s)) \supset \mathsf{C}\neg Holds(f, Result(a, s))$$

Figure 5.3: UCL theory for Example 2.

$$Holds(Winner, Result(BetHeads, Result(Toss, S_0)))$$
$$Holds(Heads, Result(Toss, s)) \supset \mathsf{C}Holds(Heads, Result(Toss, s))$$
$$\neg Holds(Heads, Result(Toss, s)) \supset \mathsf{C}\neg Holds(Heads, Result(Toss, s))$$
$$\mathsf{C}Holds(Heads, s) \supset \mathsf{C}Holds(Winner, Result(BetHeads, s))$$
$$\mathsf{C}\neg Holds(Heads, s) \supset \mathsf{C}\neg Holds(Winner, Result(BetHeads, s))$$
$$Holds(f, S_0) \supset \mathsf{C}Holds(f, S_0) \qquad \neg Holds(f, S_0) \supset \mathsf{C}\neg Holds(f, S_0)$$
$$\mathsf{C}Holds(f, s) \wedge Holds(f, Result(a, s)) \supset \mathsf{C}Holds(f, Result(a, s))$$
$$\mathsf{C}\neg Holds(f, s) \wedge \neg Holds(f, Result(a, s)) \supset \mathsf{C}\neg Holds(f, Result(a, s))$$

Figure 5.4: UCL theory for Example 3.

The Correspondence Theorem of Chapter 3, along with Theorem 5.10, can lead to a still more general embedding of $\mathcal{AC}$ in UCL. There is a minor complication though. The default theories used to establish the Correspondence Theorem are not guaranteed to yield complete extensions. While it is possible to modify them to do so, we will not pursue that possibility here.

Instead we move toward simpler kinds of UCL theories, and simpler representations of action domains in UCL. As we will see, one way to make UCL theories mathematically simpler is to eliminate occurrences of the modal operator C, particularly negative occurrences.

Here is a related proposition.

Proposition 5.14 *Let T be a UCL theory in which every formula has the standard form*

$$\mathsf{C}\alpha \wedge \beta \supset \mathsf{C}\gamma_1 \vee \cdots \vee \mathsf{C}\gamma_n \tag{5.15}$$

where all of $\alpha, \beta, \gamma_1, \ldots, \gamma_n$ are nonmodal formulas. Any interpretation causally explained by T is also causally explained by any UCL theory T' that can be obtained from T by replacing some or all formulas (5.15) of T with the corresponding formula

$$\alpha \wedge \beta \supset \mathsf{C}\gamma_1 \vee \cdots \vee \mathsf{C}\gamma_n\,. \tag{5.16}$$

Proof. Assume that I is causally explained by T. Thus, $(I, \{I\}) \models T$. If (5.15) is true in $(I, \{I\})$, then so is (5.16). Hence, $(I, \{I\}) \models T'$. Let S be a superset of $\{I\}$ such that $(I, S) \models T'$. If (5.16) is true in (I, S), then so is (5.15). Hence, $(I, S) \models T$, and since I is causally explained by T, $S = \{I\}$. Therefore, I is causally explained by T'. □

To see that, in general, the converse of Proposition 5.14 does not hold, consider $T = \{\mathsf{C}p \supset \mathsf{C}p\}$ and $T' = \{p \supset \mathsf{C}p\}$, with p the only atom in the language. The interpretation $\{p\}$ is causally explained by T', but not by T.

$Holds(Trotting, S_0)$
$Holds(Loaded(Gun_1), S_0) \vee Holds(Loaded(Gun_2), S_0)))$
$\neg Holds(Alive, s) \supset \mathsf{C}\neg Holds(Trotting, s)$
$Holds(Loaded(x), s) \supset \mathsf{C}\neg Holds(Alive, Result(Shoot(x), s))$
$Holds(f, S_0) \supset Holds(f, S_0)$ $\quad$ $\neg Holds(f, S_0) \supset \neg Holds(f, S_0)$
$Holds(f, s) \wedge Holds(f, Result(a, s)) \supset \mathsf{C}Holds(f, Result(a, s))$
$\neg Holds(f, s) \wedge \neg Holds(f, Result(a, s)) \supset \mathsf{C}\neg Holds(f, Result(a, s))$

Figure 5.5: Another UCL theory for Example 1.

$\neg Holds(Open, S_0)$
$\mathsf{C}\neg Holds(Fastened(x), Result(Unfasten(x), s))$
$\neg Holds(Fastened(Clasp_1), s) \wedge \neg Holds(Fastened(Clasp_2), s) \supset \mathsf{C}Holds(Open, s)$
$Holds(f, S_0) \supset \mathsf{C}Holds(f, S_0)$ $\quad$ $\neg Holds(f, S_0) \supset \mathsf{C}\neg Holds(f, S_0)$
$Holds(f, s) \wedge Holds(f, Result(a, s)) \supset \mathsf{C}Holds(f, Result(a, s))$
$\neg Holds(f, s) \wedge \neg Holds(f, Result(a, s)) \supset \mathsf{C}\neg Holds(f, Result(a, s))$

Figure 5.6: Another UCL theory for Example 2.

Nonetheless, it is possible to show that this transformation preserves all causally explained interpretations for the UCL theories in Figures 5.2–5.4, yielding the still simpler versions displayed in Figures 5.5–5.7, in which all negative occurrences of C are eliminated. Next we state a general theorem along these lines. Its proof can be constructed by using Theorem 5.10 to map the problem into default logic, and then reasoning on the basis of the Splitting Sequence Theorem for default logic (stated and proved in Section 4.1). For this we'll want a few definitions.

Let T be a UCL theory whose formulas have the (simpler) standard form

$$\mathsf{C}\alpha \wedge \beta \supset \mathsf{C}\gamma \tag{5.17}$$

$$Holds(Winner, Result(BetHeads, Result(Toss, S_0)))$$
$$Holds(Heads, Result(Toss, s)) \supset \mathsf{C}Holds(Heads, Result(Toss, s))$$
$$\neg Holds(Heads, Result(Toss, s)) \supset \mathsf{C}\neg Holds(Heads, Result(Toss, s))$$
$$Holds(Heads, s) \supset \mathsf{C}Holds(Winner, Result(BetHeads, s))$$
$$\neg Holds(Heads, s) \supset \mathsf{C}\neg Holds(Winner, Result(BetHeads, s))$$
$$Holds(f, S_0) \supset \mathsf{C}Holds(f, S_0) \qquad \neg Holds(f, S_0) \supset \mathsf{C}\neg Holds(f, S_0)$$
$$Holds(f, s) \wedge Holds(f, Result(a, s)) \supset \mathsf{C}Holds(f, Result(a, s))$$
$$\neg Holds(f, s) \wedge \neg Holds(f, Result(a, s)) \supset \mathsf{C}\neg Holds(f, Result(a, s))$$

Figure 5.7: Another UCL theory for Example 3.

where α, β and γ are nonmodal formulas. Let λ be a total function from the atoms of the language of T to the ordinals. For each atom A, we call $\lambda(A)$ the *level* of A. We say that λ *splits* T if, for every formula (5.17)

- all atoms that occur in γ have the same level, and
- no atom that occurs in α or β has a level greater than an atom that occurs in γ.

We say that a formula (5.17) is *stratified by* λ if

- every atom that occurs in α has a level less than the level of every atom in γ.

Theorem 5.15 *Let T be a UCL theory all of whose formulas have form (5.17). If λ splits T, then T has the same causally explained interpretations as any UCL theory T' that can be obtained by replacing any or all formulas (5.17) that are stratified by λ with the corresponding formula*

$$\alpha \wedge \beta \supset \mathsf{C}\gamma\,.$$

Proof Sketch. Use Theorem 5.10 to map UCL theories T and T' to default theories D and D'. The interpretations causally explained by T correspond to the

complete, consistent extensions of D, and the interpretations causally explained by T' correspond to the complete, consistent extensions of D'. If λ splits T, λ can be used to construct a splitting sequence for the default theory D, as follows. Let μ be an ordinal such that, for every atom A, $\lambda(A) < \mu$. Then for every $\alpha < \mu$, $U_\alpha = \{A : \lambda(A) < \alpha\}$. The resulting sequence also splits D'. Complete the proof by using the Splitting Sequence Theorem to show that D and D' have the same complete, consistent extensions. The key observation is that if λ stratifies a formula (5.17), then replacing the default rule $\frac{\alpha : \beta}{\gamma}$ with the corresponding rule $\frac{: \alpha \wedge \beta}{\gamma}$ does not change the complete, consistent extensions. □

It would be straightforward to extend the Splitting Theorems for default logic to cover also disjunctive default logic, in which case this theorem could be likewise extended to cover UCL theories whose formulas have standard form (5.15).

5.6 Flat and Definite UCL Theories

Here we define the class of UCL theories called "flat." Flat UCL theories correspond exactly to the so-called language of causal theories introduced by McCain and Turner in [MT97]. For this subclass of UCL there is a mathematically simpler characterization of causally explained interpretations. We also consider in this section the still more restricted class of "definite" UCL theories, for which there is a concise translation into classical propositional logic.

5.6.1 Flat UCL Theories

A UCL formula is *flat* if it has the form

$$\phi \supset \mathsf{C}\psi \tag{5.18}$$

where both ϕ and ψ are nonmodal formulas. A UCL theory is *flat* if all of its formulas are.

Notice that a nonmodal formula ϕ is S5-equivalent to the formula $\neg\phi \supset \mathsf{C}\mathit{False}$, which is flat. (So we can think of nonmodal formulas as essentially flat.)

Given a flat UCL theory T and an interpretation I, we define

$$T^I = \{\, \psi \,:\, \text{for some } \phi,\ \phi \supset \mathsf{C}\psi \in T \text{ and } I \models \phi \,\}. \tag{5.19}$$

Theorem 5.16 *For any flat UCL theory T, an interpretation I is causally explained by T if and only if I is the unique model of T^I.*

Lemma 5.17 *For any flat UCL theory T and UCL structure (I,S), $(I,S) \models T$ if and only if, for all $I' \in S$, $I' \models T^I$.*

Proof. The lemma follows easily from the following observation. For any flat UCL formula $\phi \supset \mathsf{C}\psi$, the following two conditions are equivalent.

- $(I,S) \models \phi \supset \mathsf{C}\psi$
- If $I \models \phi$, then, for all $I' \in S$, $I' \models \psi$. □

Proof of Theorem 5.16. ($\Longrightarrow$) Assume I is the unique model of T^I. By Lemma 5.17, $(I,\{I\}) \models T$. Let S be a superset of $\{I\}$ such that $(I,S) \models T$. By Lemma 5.17, for all $I' \in S$, $I' \models T^I$. It follows that $S = \{I\}$, so $(I,\{I\})$ is the unique I-model of T.

($\Longleftarrow$) Assume that $(I,\{I\})$ is the unique I-model of T. By Lemma 5.17, $I \models T^I$. Assume that $I' \models T^I$. By Lemma 5.17, $(I,\{I,I'\}) \models \mathit{ucl}(T)$. It follows that $I = I'$, so I is the unique model of T^I. □

5.6.2 Definite UCL Theories

A flat UCL formula $\phi \supset \mathsf{C}\psi$ is *definite* if ψ is either a literal or the formula *False*. A flat UCL theory T is *definite* if

- each of its formulas is definite, and
- for every literal L, finitely many formulas in T have consequent $\mathsf{C}L$.

Notice that, due to the first condition, an interpretation I is causally explained by a definite UCL theory T if and only if $I = T^I$.

We are particularly interested in definite UCL theories because they have a concise translation into classical propositional logic, which we call "literal completion."

Let T be a definite UCL theory. By the *literal completion* of T, denoted by $\mathit{lcomp}(T)$, we mean the classical propositional theory obtained by an elaboration of the Clark completion method [Cla78], as follows. For each literal L in the language of T, include in $\mathit{lcomp}(T)$ the formula

$$L \equiv (\phi_1 \vee \cdots \vee \phi_n) \tag{5.20}$$

where $\phi_1, \ldots, \phi_n$ are the antecedents of the formulas in T with consequent $\mathsf{C}L$. (Of course, if no causal law in T has consequent $\mathsf{C}L$, then (5.20) becomes $L \equiv \mathit{False}$.) We will call formula (5.20) the *completion* of L. Also, for each formula of the form $\phi \supset \mathsf{C}\mathit{False}$ in T, include in $\mathit{lcomp}(T)$ the formula $\neg\phi$. We will sometimes refer to flat UCL formulas with consequent $\mathsf{C}\mathit{False}$ as *constraints*.

For example, let T be the UCL theory (in the language with exactly the atoms p and q) consisting of the formulas

$$p \supset \mathsf{C}p\,,\ \neg q \supset \mathsf{C}p\,,\ q \supset \mathsf{C}q\,,\ \neg q \supset \mathsf{C}\neg q\,,\ q \supset \mathsf{C}\mathit{False}\,.$$

UCL theory T is definite, and $\mathit{lcomp}(T)$ is

$$\{\, p \equiv p \vee \neg q\,,\ \neg p \equiv \mathit{False}\,,\ q \equiv q\,,\ \neg q \equiv \neg q\,,\ \neg q \,\}\,.$$

Theorem 5.18 (Literal Completion Theorem) *An intepretation I is causally explained by a definite UCL theory T if and only if $I \models \mathit{lcomp}(T)$.*

Proof. Assume that I is causally explained by T. By Theorem 5.16, $I = T^I$. So, for every literal $L \in I$,

- there is a formula ϕ such that $\phi \supset \mathsf{C}L$ belongs to T and $I \models \phi$, and
- there is no formula ϕ such that $\phi \supset \mathsf{C}\overline{L}$ belongs to T and $I \models \phi$.

It follows that for every literal $L \in I$,

- I satisfies the completion of L, and
- I satisfies the completion of $\overline{L}$.

That is, I satisfies the completion of every literal in the language of T. Similarly, since $False \notin T^I$, we can conclude that I satisfies every formula in $lcomp(T)$ obtained from a constraint. So I is a model of $lcomp(T)$.

Proof in the other direction is similar. □

The following corollary to Theorem 5.18 suggests an approach to query evaluation for definite UCL theories.

Corollary 5.19 *Let T be a definite UCL theory, Γ a set of nonmodal formulas, and ϕ a formula. $\Gamma \cup T \mid\sim \phi$ if and only if $lcomp(D) \cup \Gamma \cup \{\neg\phi\}$ is unsatisfiable.*

5.7 (More) Causal Theories of Action in UCL

In this section, we introduce a family of languages for describing action domains and illustrate their use. This discussion follows the presentation in [MT97], where the same approach was introduced in a mathematically more limited setting (specifically, the language of "causal theories"). In this approach, time has the structure of the natural numbers, and action occurrences become propositions in the language. Thus, concurrent actions can be treated more conveniently. Moreover, as we have already argued (Chapter 2), this ontology avoids certain technical difficulties that can arise in the situation calculus due to the phenomenon of "unreachable" situations. When time has a linear structure and action occurrences are represented by propositions,

the question of whether an action can be performed in a particular situation need not be explicitly addressed in the theory. Instead, if an action can be performed in a situation, there will be, roughly speaking, some causally possible world in which it actually is.[3]

5.7.1 $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ Languages

It is convenient to specify the underlying propositional signature by means of three pairwise-disjoint sets: a nonempty set $\mathbf{F}$ of *fluent names*, a set $\mathbf{A}$ of *action names*, and a nonempty set $\mathbf{T}$ of *time names*, corresponding to the natural numbers or an initial segment of them. The atoms of the language $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ are divided into two classes, defined as follows. The *fluent atoms* are expressions of the form f_t such that $f \in \mathbf{F}$ and $t \in \mathbf{T}$. Intuitively, f_t is true if and only if the fluent f holds at time t. The *action atoms* are expressions of the form a_t such that $a \in \mathbf{A}$ and $t, t+1 \in \mathbf{T}$. Intuitively, a_t is true if and only if the action a occurs at time t. An *action literal* is an action atom or its negation. A *fluent literal* is a fluent atom or its negation. A *fluent formula* is a propositional combination of fluent atoms. We say that a formula refers to a time t if an atom whose subscript is t occurs in it.

An $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description is a UCL theory in an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ language.

5.7.2 $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ Domain Descriptions

We illustrate the approach by formalizing a slight elaboration of Lin's Suitcase domain, considered previously as Example 2 in Section 3.2, and discussed elsewhere in the dissertation as well. One thing to notice about the current formalization is that it allows for concurrent actions, unlike the situation calculus versions we have considered up to now.

[3] A similar idea is made precise in the third part of the dissertation, where the executablity of a plan is defined for the kind of UCL action theories introduced in this section.

Recall that there is a suitcase with two latches, each of which may be in either of two positions, up or down. The suitcase is spring-loaded so that whenever both latches are in the up position the suitcase is caused to be open. We model the opening of the suitcase as a static effect (as Lin does); that is, we do not model a state of the domain in which both latches are up but the suitcase is not (yet) open.

We take time names corresponding to the natural numbers, and we choose fluent names and action names as follows.

Fluents
- $Up(L_1)$: the first latch is up
- $Up(L_2)$: the second latch is up
- $IsOpen$: the suitcase is open

Actions
- $Toggle(L_1)$: toggle the first latch
- $Toggle(L_2)$: toggle the second latch
- $Close$: close the suitcase

Given our choice of language, the Suitcase domain can be partially formalized by the following schemas, where l is a metavariable ranging over $\{L_1, L_2\}$.

$$Toggle(l)_t \wedge Up(l)_t \supset \mathsf{C}\neg Up(l)_{t+1} \tag{5.21}$$

$$Toggle(l)_t \wedge \neg Up(l)_t \supset \mathsf{C}\, Up(l)_{t+1} \tag{5.22}$$

$$Close_t \supset \mathsf{C}\neg IsOpen_{t+1} \tag{5.23}$$

$$Up(L_1)_t \wedge Up(L_2)_t \supset \mathsf{C}\, IsOpen_t \tag{5.24}$$

According to schemas (5.21) and (5.22), whenever a latch is toggled at a time t it is caused to be in the opposite state at time $t+1$. Schema (5.23) says that whenever the suitcase is closed at a time t it is caused to be not open at $t+1$. Schema (5.24) says that whenever both latches are up at a time t the suitcase is caused to be open also at t. Schemas (5.21)–(5.23) express "dynamic causal laws." Schema (5.24) expresses a "static causal law."

Notice that (5.24) is not the weaker formula

$$\mathsf{C}\, Up(L_1)_t \wedge \mathsf{C}\, Up(L_2)_t \supset \mathsf{C}\, IsOpen_t \tag{5.25}$$

which would correspond more closely to the static causal laws studied in the first part of the dissertation (and also to the corresponding situation calculus formula in Figure 5.3). In this case, the weaker version would not affect the causally explained interpretations (as can be shown using Theorem 5.15). Given this, we prefer (5.24) to (5.25), since (5.24) is definite.

Similarly, one might expect that in place of (5.21) and (5.22), we would write

$$Toggle(l)_t \wedge \mathsf{C}\, Up(l)_t \supset \mathsf{C}\neg Up(l)_{t+1} \tag{5.26}$$

$$Toggle(l)_t \wedge \mathsf{C}\neg Up(l)_t \supset \mathsf{C}\, Up(l)_{t+1}\,. \tag{5.27}$$

Again, Theorem 5.15 could be used to show that for this domain description it makes no difference—the causally explained interpretations would not be affected. In fact, in this case, we can invoke a somewhat simpler result, Proposition 5.20 below, which is a straightforward consequence of Theorem 5.15, based on the following definitions.

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description whose formulas have the standard form (5.17). That is, every formula in D has the form

$$\mathsf{C}\alpha \wedge \beta \supset \mathsf{C}\gamma\,.$$

We say that D *respects the flow of time* if every formula (5.17) in D satisfies the following three conditions.

- γ refers to at most one time.
- If γ refers to a time t, then neither α nor β refer to a time later than t.
- If a fluent atom that refers to a time t occurs in γ, then every action atom that occurs in α or β refers to a time earlier than t.

We say that a formula (5.17) in D is *stratified by time* if

- every time that α refers to is earlier than every time that γ refers to.

Proposition 5.20 *Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description whose formulas have the standard form (5.17). If D respects the flow of time, then D has the same causally explained interpretations as any $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description D' that can be obtained by replacing any or all formulas (5.17) that are stratified by time with the corresponding formula*

$$\alpha \wedge \beta \supset \mathsf{C}\gamma\,.$$

This result follows easily from Theorem 5.15. (Let λ map each atom to the time it refers to.)

The UCL theory (5.21)–(5.24) is an incomplete description of the Suitcase domain because it does not represent sufficient conditions for certain facts being caused: namely, facts preserved by inertia, facts about the initial situation, and facts about which actions occur (and when). The following schemas provide a standard way to fill these gaps.

In the following two schemas, a is a metavariable for action names.

$$a_t \supset \mathsf{C}a_t \tag{5.28}$$

$$\neg a_t \supset \mathsf{C}\neg a_t \tag{5.29}$$

Schema (5.28) says that the occurrence of an action a at a time t is caused whenever a occurs at t. Schema (5.29) says that the non-occurrence of an action a at a time t is caused whenever a does not occur at t. In effect, by these schemas we represent that facts about action occurrences are exogenous to the theory.

In the following two schemas, f is a metavariable for fluent names.

$$f_0 \supset \mathsf{C}f_0 \tag{5.30}$$

$$\neg f_0 \supset \mathsf{C}\neg f_0 \tag{5.31}$$

In effect, by these schemas we represent that facts about the initial values of fluents may be exogenous to the theory.

By a *fluent designating formula* we mean a propositional combination of fluent names. Given a fluent designating formula σ and a time name t, we write σ_t to stand for the fluent formula obtained from σ by simultaneously replacing each occurrence of each fluent name f by the fluent atom f_t.

Let $\mathbf{I}$ be a set of fluent designating formulas. We express that the fluents designated by the formulas in $\mathbf{I}$ are inertial by writing the following schema, where σ is a metavariable ranging over $\mathbf{I}$.

$$\sigma_t \wedge \sigma_{t+1} \supset \mathsf{C}\sigma_{t+1} \tag{5.32}$$

According to schema (5.32), whenever a fluent designated in $\mathbf{I}$ holds at two successive times, its truth at the second time is taken to be caused simply by virtue of its persistence.[4] For the Suitcase domain, we take $\mathbf{I}$ to be the set of all fluent names and their negations. Thus, the inertia laws for the Suitcase domain can also be represented by the schemas

$$f_t \wedge f_{t+1} \supset \mathsf{C}f_{t+1}$$

and

$$\neg f_t \wedge \neg f_{t+1} \supset \mathsf{C}\neg f_{t+1}\,,$$

where f is a metavariable for fluent names. In other cases, there may be inertial fluents that are not designated by fluent names or their negations, and, conversely, there may be fluent names or negations of fluent names that do not designate inertial fluents. We will see such a case in Section 5.7.3, when we describe how to introduce explicit definitions in $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions.

[4] Notice that $\mathsf{C}\sigma_t \wedge \sigma_{t+1} \supset \mathsf{C}\sigma_{t+1}$ is stratified by time. Hence, when used in a domain description that respects the flow of time, (5.32) can be replaced with $\mathsf{C}\sigma_t \wedge \sigma_{t+1} \supset \mathsf{C}\sigma_{t+1}$ without affecting the causally explained interpretations.

$$
\begin{array}{l}
Toggle(l)_t \wedge Up(l)_t \supset \mathsf{C}\neg Up(l)_{t+1} \\
Toggle(l)_t \wedge \neg Up(l)_t \supset \mathsf{C}\, Up(l)_{t+1} \\
Close_t \supset \mathsf{C}\neg IsOpen_{t+1} \\
Up(L_1)_t \wedge Up(L_2)_t \supset \mathsf{C}IsOpen_t \\
a_t \supset \mathsf{C}a_t \quad \neg a_t \supset \mathsf{C}\neg a_t \quad f_0 \supset \mathsf{C}f_0 \quad \neg f_0 \supset \mathsf{C}\neg f_0 \\
f_t \wedge f_{t+1} \supset \mathsf{C}f_{t+1} \qquad \neg f_t \wedge \neg f_{t+1} \supset \mathsf{C}\neg f_{t+1}
\end{array}
$$

Figure 5.8: $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ description D_3 of Lin's Suitcase domain.

Schemas (5.21)–(5.24) and (5.28)–(5.32) express the complete UCL theory for the Suitcase domain. Schemas (5.21)–(5.24) are domain specific. We often refer to the remaining schemas (5.28)–(5.32) as *standard schemas*. Intuitively, the standard schemas exempt specific classes of facts from the principle of universal causation. (Notice that the standard schemas respect the flow of time.) The complete $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ description D_3 of the Suitcase domain appears in Figure 5.8. Notice that it is definite.

Let I be the interpretation characterized below.

$$
\begin{array}{llll}
\neg Toggle(L_1)_0 & \neg Toggle(L_1)_1 & \neg Toggle(L_1)_2 & \cdots \\
Toggle(L_2)_0 & \neg Toggle(L_2)_1 & \neg Toggle(L_2)_2 & \cdots \\
\neg Close_0 & \neg Close_1 & \neg Close_2 & \cdots \\
Up(L_1)_0 & Up(L_1)_1 & Up(L_1)_2 & \cdots \\
\neg Up(L_2)_0 & \bullet\ Up(L_2)_1 & Up(L_2)_2 & \cdots \\
\neg IsOpen_0 & \bullet\ IsOpen_1 & IsOpen_2 & \cdots
\end{array}
$$

Interpretation I specifies, for all actions a and times t, whether or not a occurs at t, and, for all fluents f and times t, whether or not f holds at t. Here, exactly one action occurs—the toggling of the second latch at time 0—and, intuitively, it results in the suitcase being open at time 1. (The ellipses indicate that after time 2 no action occurs and no fluent changes its value. The bullets indicate literals that

are "explained" by domain specific schemas. All others are explained by standard schemas.) It is not difficult to see that I is causally explained by D_3.

The following formula is a UCL-consequence of D_3.

$$Up(L_1)_0 \wedge Up(L_2)_0 \wedge Close_0 \supset Toggle(L_1)_0 \vee Toggle(L_2)_0 \tag{5.33}$$

In general, when both latches are up, it is impossible to perform *only* the action of closing the suitcase; one must also concurrently toggle at least one of the latches. If this seems unintuitive, recall that we have chosen to model the suitcase being open as a static effect of the latches being up, so there is no time in any causally possible world at which both latches are up and the suitcase is closed.

5.7.3 Expressive Possibilities

The previous example demonstrates that $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions can be used to represent some standard features of action domains, such as indirect effects of actions, implied action preconditions and concurrent actions. Next we briefly describe a few of the additional expressive possibilities of the approach.

Ramification and Qualification Constraints

Ramification and qualification constraints, in the sense of Lin and Reiter [LR94], are formalized by schemas of the forms $\mathsf{C}\sigma_t$ and σ_t respectively, where σ (the "state constraint") is a fluent designating formula. (Similar results are established by Proposition 5.8 and Theorem 5.12.)

Nondeterministic Actions

The semantics of UCL rests on the principle of universal causation, according to which every fact is caused. Intuitively, in the case of a nondeterministic action, there is no cause for one of its possible effects rather than another. We have already

$$Toss_t \wedge Heads_{t+1} \supset \mathsf{C}Heads_{t+1}$$
$$Toss_t \wedge \neg Heads_{t+1} \supset \mathsf{C}\neg Heads_{t+1}$$
$$a_t \supset \mathsf{C}a_t \quad \neg a_t \supset \mathsf{C}\neg a_t \quad f_0 \supset \mathsf{C}f_0 \quad \neg f_0 \supset \mathsf{C}\neg f_0$$
$$f_t \wedge f_{t+1} \supset \mathsf{C}f_{t+1} \qquad \neg f_t \wedge \neg f_{t+1} \supset \mathsf{C}\neg f_{t+1}$$

Figure 5.9: $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ description D_4 of Coin Toss domain.

seen, however—in standard schemas (5.28) through (5.32)—that there are ways of effectively exempting facts from the principle of universal causation. We can use laws of a similar form to describe nondeterministic actions. For instance, the nondeterministic effect of tossing a coin can be described by the following pair of schemas.

$$Toss_t \wedge Heads_{t+1} \supset \mathsf{C}Heads_{t+1} \tag{5.34}$$
$$Toss_t \wedge \neg Heads_{t+1} \supset \mathsf{C}\neg Heads_{t+1} \tag{5.35}$$

Intuitively, according to schemas (5.34) and (5.35), for every time t, $Toss_t$ renders $Heads_{t+1}$ exogenous. We'll consider some related results in Section 5.9.3.

Notice that these formulas are similar to those used in the UCL formalization of Example 3 appearing in Figure 5.7.

This description can be completed by adding the UCL formulas given by the standard schemas (5.28)–(5.32), with the set of inertial fluents $\mathbf{I} = \{Heads, \neg Heads\}$. The complete domain description D_4 is represented in Figure 5.9. We will consider this action domain, and some elaborations of it, in more detail in the next chapter, in relation to the definitions of various classes of plans. In the meantime, notice that D_4 is definite.

Defined Fluents

Given an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, we add a defined fluent f ($f \notin \mathbf{F}$) by first adding f to the set of fluent names and then defining f by means of a schema

$$f_t \equiv \sigma_t \tag{5.36}$$

where σ is a fluent designating formula that doesn't mention f. Notice that the set $\mathbf{I}$ used to designate the inertial fluents is not altered in this process. Intuitively speaking, the defined fluent inherits any inertial properties it may have from its definiens. The correctness of this method of introducing defined fluents follows from the remarks on definitional extension in Section 5.2.2. Notice that it also corresponds closely to the method inherited via the translation of $\mathcal{AC}$ in Theorem 5.12.

Delayed Effects and Things that Change by Themselves

Because we refer explicitly to time points in our action descriptions, we may, if we wish, describe actions with delayed effects. Similarly, we can write formulas that refer to more than two time points. We may also model things that change by themselves. This we can do simply by writing causal laws that relate fluents at different times, without mentioning any actions. Here we consider an example along these lines involving the dynamic mechanism of falling dominos.

We wish to describe the chain reaction of dominos falling over one after the other, after the first domino is tipped over.

Let the fluent names be

$$Up(1), \ldots, Up(4),$$

and let the single action name be

$$Tip.$$

Identify time with the natural numbers $0, \ldots, 4$.

$$Tip_t \supset \mathsf{C}\neg Up(d)_{t+1} \qquad Tip_t \supset Up(d)_t$$
$$Up(d)_t \wedge \neg Up(d)_{t+1} \supset \mathsf{C}\neg Up(d+1)_{t+2}$$
$$a_t \supset \mathsf{C}a_t \quad \neg a_t \supset \mathsf{C}\neg a_t \quad f_0 \supset \mathsf{C}f_0 \quad \neg f_0 \supset \mathsf{C}\neg f_0$$
$$f_t \wedge f_{t+1} \supset \mathsf{C}f_{t+1} \qquad \neg f_t \wedge \neg f_{t+1} \supset \mathsf{C}\neg f_{t+1}$$

Figure 5.10: $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ description D_5 of Dominos domain.

Here, as usual, we assume that facts about the occurrences of actions are exogenous. We also assume that facts about the initial values of fluents are exogenous. The fluent names $Up(1), \ldots, Up(4)$, and their negations, will be designated inertial.

We describe the direct effect and action precondition of the *Tip* action by writing

$$Tip_t \supset \mathsf{C}\neg Up(1)_{t+1} \tag{5.37}$$
$$Tip_t \supset Up(1)_t\,. \tag{5.38}$$

So *Tip* is the action of tipping over the first domino. It can only be done if the first domino is standing up.

We describe the chain reaction mechanism as follows, where d is a metavariable ranging over numbers 1, 2, 3.

$$Up(d)_t \wedge \neg Up(d)_{t+1} \supset \mathsf{C}\neg Up(d+1)_{t+2} \tag{5.39}$$

Notice that this schema does not mention an action. It describes dynamic change involving three distinct time points. Roughly speaking, if domino d falls in the interval from t to $t+1$, then domino $d+1$ is caused to fall in the interval from $t+1$ to $t+2$.

Let D_5 be the domain description given by schemas (5.37)–(5.39), along with the standard schemas (5.28)–(5.32), as shown in Figure 5.10. Notice that although D_5 is not definite, it is S5-equivalent to a definite UCL theory, since

$Tip(d)_t \supset Up(d)_t$ is S5-equivalent to the formula $\neg(Tip(d)_t \supset Up(d)_t) \supset \mathsf{C}False$, which is definite.

Let I be the interpretation shown below.

Tip_0	$\neg Tip_1$	$\neg Tip_2$	$\neg Tip_3$	
$Up(1)_0$	$\bullet\ \neg Up(1)_1$	$\neg Up(1)_2$	$\neg Up(1)_3$	$\neg Up(1)_4$
$Up(2)_0$	$Up(2)_1$	$\bullet\ \neg Up(2)_2$	$\neg Up(2)_3$	$\neg Up(2)_4$
$Up(3)_0$	$Up(3)_1$	$Up(3)_2$	$\bullet\ \neg Up(3)_3$	$\neg Up(3)_4$
$Up(4)_0$	$Up(4)_1$	$Up(4)_2$	$Up(4)_3$	$\bullet\ \neg Up(4)_4$

The only action occurrence is *Tip* at time 0. One easily verifies that I is causally explained by D_5. The four literals preceded by bullets are "explained" by the domain specific schemas (5.37) and (5.39). The others are explained by standard schemas.

The Course of Nature

The fact that the commonsense law of inertia can be expressed straightforwardly in UCL makes it easy to generalize it, as follows. Rather than supposing that things tend to stay the same, we can imagine more generally that they tend to change in particular ways. That is, there is a course that nature would follow, in the absence of interventions.

As an example, we will formalize a dynamic domain from [GL98]. In this domain there is a pendulum. In the course of nature (i.e., in the absence of interventions), the pendulum bob swings back and forth from right to left. However, at any time the agent can intervene to change the course of nature by holding the bob in its current location. So long as he continues to hold it, the bob remains where it is. When he no longer holds it, the bob resumes its natural course, swinging back and forth from right to left.

In formalizing the Pendulum domain, we will use a single action name *Hold* and fluent name *Right*. We will identify time with the natural numbers $0, \ldots, 4$.

$$Hold_t \wedge Right_t \supset \mathsf{C}Right_{t+1}$$
$$Hold_t \wedge \neg Right_t \supset \mathsf{C}\neg Right_{t+1}$$
$$\neg Right_t \wedge Right_{t+1} \supset \mathsf{C}Right_{t+1}$$
$$Right_t \wedge \neg Right_{t+1} \supset \mathsf{C}\neg Right_{t+1}$$
$$a_t \supset \mathsf{C}a_t \quad \neg a_t \supset \mathsf{C}\neg a_t \quad f_0 \supset \mathsf{C}f_0 \quad \neg f_0 \supset \mathsf{C}\neg f_0$$

Figure 5.11: $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ description D_6 of Pendulum domain.

The effects of the action *Hold* are specified straightforwardly by writing

$$Hold_t \wedge Right_t \supset \mathsf{C}Right_{t+1} \tag{5.40}$$
$$Hold_t \wedge \neg Right_t \supset \mathsf{C}\neg Right_{t+1}. \tag{5.41}$$

The behavior of the pendulum in the absence of interventions is described by writing

$$\neg Right_t \wedge Right_{t+1} \supset \mathsf{C}Right_{t+1} \tag{5.42}$$
$$Right_t \wedge \neg Right_{t+1} \supset \mathsf{C}\neg Right_{t+1}. \tag{5.43}$$

Like the standard schema (5.32) for inertia, schemas (5.42) and (5.43) describe a course of nature. Here the course of nature is dynamic rather than static, but otherwise there are clear similarities between the two pairs of schemas. Both pairs allow for the possibility that the course of nature may be overridden by the effects of actions, and both achieve this without mentioning facts about the non-occurrence of actions as preconditions. In essence, schemas (5.42) and (5.43) solve the frame problem for the dynamic fluent *Right* in the same way that (5.32) solves the frame problem for inertial fluents.

Let D_6 be the definite causal theory given by schemas (5.28)–(5.31) and (5.40)–(5.43), as shown in Figure 5.11. Consider the following interpretation I.

$$\neg Hold_0 \quad Hold_1 \quad Hold_2 \quad \neg Hold_3$$
$$Right_0 \quad \neg Right_1 \quad \neg Right_2 \quad \neg Right_3 \quad Right_4$$

One easily verifies that I is causally explained by D_6.

5.8 A Subset of UCL in Circumscription

Let T be a finite UCL theory, with a finite signature, in which the operator C is applied only to literals. In this section, we show that T can be reduced to a circumscriptive theory $ct(T)$.[5]

The language $\mathcal{L}$ of $ct(T)$ is a second-order language with equality, with two sorts, *atom* and *value*. Let At stand for the set of atoms in the language of T. In $\mathcal{L}$, the set of all object constants of sort *atom* is exactly the set At. The symbols $\top$ and $\bot$ will be the two object constants of sort *value*. $\mathcal{L}$ includes exactly two predicates, in addition to equality: a unary predicate *Holds* of sort *atom* and a binary predicate *Caused* of sort *atom* $\times$ *value*. We will use a variable x of sort *atom*, and a variable v of sort *value*.

We begin the description of $ct(T)$ by letting $\mathcal{C}(T)$ stand for the sentence

$$\bigwedge_{\phi \in T} \mathcal{C}(\phi) \tag{5.44}$$

where $\mathcal{C}(\phi)$ is defined recursively, as follows.

$$\mathcal{C}(X) = Holds(X) \quad \text{if } X \in At \tag{5.45}$$
$$\mathcal{C}(\mathsf{C}X) = Caused(X, \top) \quad \text{if } X \in At \tag{5.46}$$
$$\mathcal{C}(\mathsf{C}\neg X) = Caused(X, \bot) \quad \text{if } X \in At \tag{5.47}$$
$$\mathcal{C}(True) = True \tag{5.48}$$
$$\mathcal{C}(\neg\phi) = \neg\mathcal{C}(\phi) \tag{5.49}$$
$$\mathcal{C}((\phi \odot \psi)) = (\mathcal{C}(\phi) \odot \mathcal{C}(\psi)) \tag{5.50}$$

Here $\odot$ stands for any binary propositional connective. Notice that this definition

[5] Familiarity with circumscription will be assumed. See, for example, [Lif93a].

depends on the finiteness of the UCL theory T, as well as the assumption that the modal operator C is applied only to literals. Notice also that $\mathcal{C}(T)$ is ground.

We will want a unique names axiom (denoted by UNA) to say that all object constants of sort *atom* denote distinct domain objects. Thus, UNA stands for the conjunction of all formulas $X \neq X'$ such that X and X' are distinct members of At. Notice that this definition depends on the finiteness of the signature of T.

The complete embedding $ct(T)$ consists of the following five sentences.

$$\text{CIRC}[\mathcal{C}(T) : \mathit{Caused}] \tag{5.51}$$

$$\forall x\,(\mathit{Holds}(x) \equiv \mathit{Caused}(x, \top)) \tag{5.52}$$

$$\forall x\,(\neg \mathit{Holds}(x) \equiv \mathit{Caused}(x, \bot)) \tag{5.53}$$

$$\text{UNA} \tag{5.54}$$

$$\forall v(v = \top \not\equiv v = \bot) \tag{5.55}$$

Notice that second-order quantification is used only implicitly, in (5.51). The models of (5.51) are simply the models of $\mathcal{C}(T)$ in which the extent of *Caused* is minimal (for a fixed universe and fixed interpretation of all nonlogical constants except *Caused*).

For every model M of $ct(T)$, there is a one-to-one correspondence between the domain objects of sort *atom* and the members of At. To see this, first notice that, because of the UNA, M maps each pair of distinct members of At to distinct domain objects. Now, suppose there is a domain object δ of sort *atom* such that M maps no member of At to δ. Because $\mathcal{C}(T)$ is ground, and M is a model of $\mathcal{C}(T)$ in which the extent of the predicate *Caused* is minimal, we can conclude that neither $\langle \delta, \top^M \rangle$ nor $\langle \delta, \bot^M \rangle$ belong to Caused^M. But the axioms (5.52) and (5.53) together imply that

$$\forall x(\mathit{Caused}(x, \top) \not\equiv \mathit{Caused}(x, \bot))\,. \tag{5.56}$$

Given, in addition, the axiom (5.55) expressing the unique names and domain closure assumptions for sort *values*, we can conclude that every model of $ct(T)$ is

isomorphic to some Herbrand model of $ct(T)$. Thus, in what follows, we restrict our attention to Herbrand interpretations.

For every UCL structure (I, S), let $M(I, S)$ be the Herbrand interpretation of $\mathcal{L}$ such that, for every $X \in At$, the following three conditions hold.

- $M(I,S) \models \mathit{Holds}(X)$ iff $(I,S) \models X$
- $M(I,S) \models \mathit{Caused}(X, \top)$ iff $(I,S) \models \mathsf{C}X$
- $M(I,S) \models \mathit{Caused}(X, \bot)$ iff $(I,S) \models \mathsf{C}\neg X$

The following lemma is a straightforward consequence of the definitions.

Lemma 5.21 *Let T be a finite UCL theory, with finite signature, in which C is applied only to literals. For any UCL structure (I,S), $(I,S) \models T$ if and only if $M(I,S) \models \mathcal{C}(T)$.*

Theorem 5.22 *Let T be a finite UCL theory, with finite signature, in which C is applied only to literals. An interpretation I is causally explained by T if and only if $M(I,\{I\})$ is a model of $ct(T)$. Moreover, every model of $ct(T)$ is isomorphic to an interpretation $M(I,\{I\})$, for some interpretation I of the language of T.*

Proof. We've already established that every model of $ct(T)$ is isomorphic to a Herbrand model. In light of (5.52) and (5.53), every Herbrand model can be written in the form $M(I,\{I\})$. Now we turn to the first part of the theorem.

($\Longrightarrow$) Assume I is causally explained by T. So $(I,\{I\}) \models T$. By Lemma 5.21, $M(I,\{I\}) \models \mathcal{C}(T)$. Also, $M(I,\{I\})$ clearly satisfies all of (5.52), (5.53), (5.54) and (5.55). It remains only to show that the extent of *Caused* in $M(I,\{I\})$ is minimal among models of $\mathcal{C}(T)$ with the same universe, and the same interpretation of all nonlogical constants except *Caused*. Any possible counterexample can be written in the form $M(I,S)$, for some superset S of $\{I\}$. So assume that $M(I,S) \models \mathcal{C}(T)$. By Lemma 5.21, $(I,S) \models T$. Since $(I,\{I\})$ is the unique I-model of T, $S = \{I\}$.

($\Longleftarrow$) Assume $M(I,\{I\})$ is a model of $ct(T)$. By Lemma 5.21, $(I,\{I\}) \models T$. Let S be a superset of $\{I\}$ such that $(I,S) \models T$. By Lemma 5.21, $M(I,S) \models \mathcal{C}(T)$. Since $M(I,\{I\})$ is a model of (5.51), we can conclude that $M(I,S) = M(I,\{I\})$. It follows that $S = \{I\}$, which shows that $(I,\{I\})$ is the unique I-model of T. That is, I is causally explained by T. □

5.9 UCL and Lin's Circumscriptive Action Theories

Lin [Lin95] recently introduced a causal approach to reasoning about action based on circumscription. In this section, we explore the relationship between Lin's circumscriptive action theories and the UCL action theories described in Section 5.7, restricted to the case when C is applied only to literals. We show that on a wide range of action domains, the two approaches coincide.

5.9.1 Lin's Circumscriptive Causal Action Theories

For the purpose of comparison, we present an account of Lin's proposal that is simplified in several ways. We do not consider non-propositional fluent and action symbols. We also do not employ the situation calculus. Instead we model worlds in which time has the structure of the natural numbers. As discussed previously (Chapter 2 and elsewhere), this simplifies matters somewhat, eliminating the need for a *Poss* (or *Reachable*) predicate. Finally, we include a domain closure assumption for fluents. In the case of propositional fluents, this is not very significant.

In some other ways, the circumscriptive approach that we describe is more general than Lin's. Because our language includes propositions about the occurrence and non-occurrence of actions, we can accomodate concurrent actions more easily than Lin. We also accomodate a wider variety of causal laws. For instance, we allow formulas expressing causal laws that refer to more than one time point and yet do not involve the occurrence of an action. We allow also for causal laws that involve

more than two time points, and we do not require that the time points be successive.

The language of the circumscriptive theory is constructed in the same manner as in the previous section, on the basis of the signature At of an underlying propositional language. For this purpose, we employ $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ languages, as described in Section 5.7.1, under the additional restriction that each of the sets $\mathbf{F}$, $\mathbf{A}$, and $\mathbf{T}$ is finite.

Let us introduce an abbreviation. Given a fluent formula ϕ, $Holds(\phi)$ stands for the formula obtained by replacing every occurrence of every fluent atom f_t in ϕ by $Holds(f_t)$.

We need axioms expressing domain closure and unique names assumptions for both sorts, as follows.

$$\forall x(\bigvee_{X \in At} x = X) \tag{5.57}$$

$$\bigwedge_{X,X' \in At, X \neq X'} X \neq X' \tag{5.58}$$

$$\forall v(v = \top \not\equiv v = \bot) \tag{5.59}$$

We also need the following axioms, saying that whatever is caused is true.

$$\forall x(Caused(x,\top) \supset Holds(x)) \tag{5.60}$$

$$\forall x(Caused(x,\bot) \supset \neg Holds(x)) \tag{5.61}$$

For the purposes of this chapter, a *Lin formula* is the conjunction of a finite set of ground sentences in which *Caused* appears only positively, and at most once in each sentence.

The next few observations help characterize the relationship between Lin formulas as specified here and the kinds of circumscriptive action theories described in [Lin95]. Assume that σ is a fluent designating formula, A is an action name, F is a fluent name, and V is either $\top$ or $\bot$. Lin's "direct effect" axioms correspond to schemas of the form

$$Holds(A_t) \wedge Holds(\sigma_t) \supset Caused(F_{t+1}, V)\,. \tag{5.62}$$

His "causal rule" axioms correspond to schemas of the form

$$Holds(\sigma_t) \supset Caused(F_t, V). \tag{5.63}$$

His "explicit precondition" axioms correspond to schemas of the form

$$Holds(A_t) \supset Holds(\sigma_t). \tag{5.64}$$

His "qualification state constraint" axioms correspond to schemas of the form

$$Holds(\sigma_t). \tag{5.65}$$

For example, consider again the Suitcase domain from [Lin95]. We'll use almost the same $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ language as in Section 5.7.2, but restrict time to a finite initial segment of the natural numbers. The Lin formula for this example is characterized by the following schemas of type (5.62) and (5.63), where l is again metavariable ranging over $\{L_1, L_2\}$.

$$Holds(Toggle(l)_t) \wedge Holds(Up(l)_t) \supset Caused(Up(l)_{t+1}, \bot) \tag{5.66}$$

$$Holds(Toggle(l)_t) \wedge \neg Holds(Up(l)_t) \supset Caused(Up(l)_{t+1}, \top) \tag{5.67}$$

$$Holds(Close_t) \supset Caused(IsOpen_{t+1}, \bot) \tag{5.68}$$

$$Holds(Up(L_1)_t) \wedge Holds(Up(L_2)_t) \supset Caused(IsOpen_t, \top) \tag{5.69}$$

Let D_7 be the UCL theory given by schemas (5.21)–(5.24), which express the domain specific part of the UCL description of the Suitcase domain from Section 5.7.2. The conjunction of the sentences given by schemas (5.66)–(5.69) is exactly $\mathcal{C}(D_7)$, where $\mathcal{C}$ is the translation function defined in Section 5.8.

Given a Lin formula D, the complete circumscriptive action theory $cat(D)$ consists of

$$\text{CIRC}[\, D : Caused \,] \tag{5.70}$$

along with the standard axioms (5.57)–(5.61) and the inertia axioms given by the schema

$$Holds(f_{t+1}) \equiv (Holds(f_t) \wedge \neg Caused(f_{t+1}, \bot)) \vee Caused(f_{t+1}, \top) \tag{5.71}$$

where f is a metavariable ranging over fluent names.

5.9.2 Lin's Circumscriptive Action Theories in UCL

The first thing to observe is that, for every Lin formula D, there is a UCL theory T in language $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ such that $\mathcal{C}(T) = D$. We will show that there is an extension $uclat(D)$ of T such that the interpretations causally explained by $uclat(D)$ correspond to the models of $cat(D)$. We obtain $uclat(D)$ by adding to T the formulas given by the standard schemas (5.28)–(5.32) from Section 5.7.2, taking $\mathbf{I}$ to be the set of all fluent names and their negations.[6]

Theorem 5.23 *For any Lin formula D, an interpretation I is causally explained by $uclat(D)$ if and only if there is a superset S of $\{I\}$ such that $M(I,S)$ is a model of $cat(D)$. Moreover, every model of $cat(D)$ is isomorphic to an interpretation $M(I,S)$, for some UCL structure (I,S).*

We begin the proof of Theorem 5.23 with a straightforward lemma.

Lemma 5.24 *Let T be a UCL theory with no nested occurrences of C, in which C occurs only positively. If $(I,S) \models T$, then for all subsets S' of S such that $I \in S'$, $(I,S') \models T$. If, in addition, C occurs at most once in each formula, $(I, S \cup S') \models T$ whenever $(I,S) \models T$ and $(I,S') \models T$.*

Proof of Theorem 5.23. We first prove the second part of the theorem. Axioms (5.57)–(5.59) allow us to restrict our attention to the Herbrand models of $cat(D)$.

[6] We note in passing that $uclat(D)$ is easily seen to be S5-equivalent to a definite UCL theory.

Axioms (5.60) and (5.61) show that every Herbrand model of $cat(D)$ can be expressed in the form $M(I,S)$.

Now we turn to the main part of the theorem. Let T be the UCL theory such that $\mathcal{C}(T) = D$. (Here we assume the most natural choice of T, which satisfies the conditions of Lemma 5.24.) Let $T' = uclat(D)$.

($\Longrightarrow$) Assume I is causally explained by T'. Thus, $(I, \{I\}) \models T$, and by Lemma 5.21, $M(I,\{I\}) \models D$. It follows that there is a superset S of $\{I\}$ such that $M(I,S)$ is a model of CIRC$[\,D : \mathit{Caused}\,]$. Clearly, $M(I,S)$ satisfies the standard axioms (5.57)–(5.61). It remains to show that $M(I,S)$ satisfies the inertia axioms given by (5.71). Suppose otherwise. Thus, there is a fluent atom f_t such that either

$$M(I,S) \models \neg \mathit{Holds}(f_t) \wedge \mathit{Holds}(f_{t+1}) \wedge \neg \mathit{Caused}(f_{t+1}, \top)$$

or

$$M(I,S) \models \mathit{Holds}(f_t) \wedge \neg \mathit{Holds}(f_{t+1}) \wedge \neg \mathit{Caused}(f_{t+1}, \bot).$$

We'll argue the first case. (The second is similar.) By Lemma 5.21, we know that $(I,S) \models \neg f_t \wedge f_{t+1} \wedge \neg \mathsf{C} f_{t+1}$. Let $I' = I \cup \{\neg f_{t+1}\} \setminus \{f_{t+1}\}$. Notice that $I' \neq I$, with $I \models f_{t+1}$ and $I' \models \neg f_{t+1}$. Since $M(I,S) \not\models \mathit{Caused}(f_{t+1}, \top)$ and $I \in S$, we can conclude by our choice of I' that $M(I,S) = M(I, S \cup \{I'\})$. Thus, $M(I, S \cup \{I'\}) \models D$, and by Lemma 5.21, $(I, S \cup \{I'\}) \models T$. By Lemma 5.24, $(I, \{I, I'\}) \models T$. One easily checks that $(I, \{I, I'\})$ also satisfies (5.28)–(5.32). So $(I, \{I, I'\}) \models T'$, which contradicts the assumption that I is causally explained by T'.

($\Longleftarrow$) Assume $M(I,S)$ is a model of $cat(D)$. Thus, $M(I,S) \models D$. By Lemma 5.21, $(I,S) \models T$. By Lemma 5.24, $(I,\{I\}) \models T$. One easily verifies that, since T' is obtained from T by adding (5.28)–(5.32), $(I,\{I\}) \models T'$. We wish to show that $(I,\{I\})$ is the unique I-model of T'. Suppose otherwise. So there is a strict superset S' of $\{I\}$ such that $(I,S') \models T'$, and there is a literal L such that $(I,S') \models L \wedge \neg \mathsf{C} L$. In light of (5.28)–(5.31), L is a fluent literal that refers to a non-zero time. Assume L has the form f_{t+1}. (The argument is analagous if L is $\neg f_{t+1}$.) So $(I,S') \models f_{t+1} \wedge \neg \mathsf{C} f_{t+1}$. In light of (5.32), $(I,S') \models \neg f_t$. So $(I,S') \models \neg f_t \wedge f_{t+1}$, and

thus $(I,S) \models \neg f_t \wedge f_{t+1}$ also. By Lemma 5.21, $M(I,S) \models \neg \mathit{Holds}(f_t) \wedge \mathit{Holds}(f_{t+1})$. It follows by (5.71) that $M(I,S) \models \mathit{Caused}(f_{t+1}, \top)$. On the other hand, since $(I,S') \not\models \mathsf{C} f_{t+1}$, $M(I,S') \not\models \mathit{Caused}(f_{t+1}, \top)$. Hence $M(I, S \cup S') \not\models \mathit{Caused}(f_{t+1}, \top)$, which shows that $M(I, S \cup S') \neq M(I,S)$. Since $(I,S') \models T'$, $(I,S') \models T$. And since $(I,S) \models T$ also, we know by Lemma 5.24 that $(I, S \cup S') \models T$. By Lemma 5.21, $M(I, S \cup S') \models D$. Since $M(I,S)$ is a model of CIRC$[\,D : \mathit{Caused}\,]$, we can conclude that $M(I, S \cup S') = M(I,S)$. Contradiction. □

5.9.3 Discussion

In [Lin95], Lin briefly discusses the possibility of using a more general form of "causal rule" axiom (5.63), in which *Caused* can occur negatively any number of times in a sentence, in addition to the one positive occurrence. For example, he suggests extending the circumscriptive action theory for the Suitcase domain with an additional fluent *IsClosed*, understood as the antonym of *IsOpen*, and adding (essentially) the schemas

$$\mathit{Caused}(\mathit{IsClosed}_t, \top) \equiv \mathit{Caused}(\mathit{IsOpen}_t, \bot) \tag{5.72}$$

$$\mathit{Caused}(\mathit{IsClosed}_t, \bot) \equiv \mathit{Caused}(\mathit{IsOpen}_t, \top) \tag{5.73}$$

to reflect this understanding. Notice that this resembles the notion of a "defined fluent," discussed in Section 5.7.3, according to which one would augment the UCL Suitcase domain description D_3 from Section 5.7.2 with

$$\mathsf{C}\,(\mathit{IsClosed}_t \equiv \neg \mathit{IsOpen}_t)\,. \tag{5.74}$$

The first thing to observe is that (5.74) entails $\mathit{IsClosed}_t \equiv \neg \mathit{IsOpen}_t$ (in S5), while (5.72) and (5.73) do not entail

$$\mathit{Holds}(\mathit{IsClosed}_t) \equiv \neg \mathit{Holds}(\mathit{IsOpen}_t)\,. \tag{5.75}$$

This correctly suggests that some models of the circumscriptive action theory fail to satisfy (5.75), for some time names t. It appears that, in this case, one can obtain a more satisfactory "definition" of *IsClosed* by also including (5.75) in the circumscriptive theory. In general though, it is unclear how to introduce defined fluents in circumscriptive action theories.

A related complication arises if we try, for instance, to replace the causal rule axiom (5.69) with

$$Caused(Up(L_1)_t, \top) \wedge Caused(Up(L_2)_t, \top) \supset Caused(IsOpen_t, \top). \quad (5.76)$$

(Replacing (5.69) with (5.76) greatly alters the meaning of the circumscriptive action theory. For instance, it allows models that fail to satisfy

$$Holds(Up(L_1)_t) \wedge Holds(Up(L_2)_t) \supset Holds(IsOpen_t). \quad (5.77)$$

It also make it impossible, intuitively speaking, to open the suitcase unless one toggles both latches at the same time.

In a subsequent paper [Lin96], Lin investigates how to extend his circumscriptive action theories to accomodate nondeterministic actions. For our purposes, the first thing to observe is that nondeterministic actions can often be described using the natural counterpart to the approach from Section 5.7.3. For instance, one can describe the nondeterministic effect of coin tossing by the schemas

$$Holds(Toss_t) \wedge Holds(Heads_{t+1}) \supset Caused(Heads_{t+1}, \top) \quad (5.78)$$

$$Holds(Toss_t) \wedge \neg Holds(Heads_{t+1}) \supset Caused(Heads_{t+1}, \bot) \quad (5.79)$$

which correspond to the UCL schemas (5.34) and (5.35) from Section 5.7.3. Lin does not (directly) consider this approach. Instead, he begins by considering a variety of methods involving sentences with multiple positive occurrences of *Caused*. For instance, he (essentially) considers a coin-toss axiom like

$$Holds(Toss_t) \supset Caused(Heads_{t+1}, \top) \vee Caused(Heads_{t+1}, \bot). \quad (5.80)$$

Notice that, in the presence of standard axioms (5.60) and (5.61) guaranteeing that whatever is caused obtains, one can equivalently replace (5.80) with (5.78) and (5.79). In UCL, the corresponding formula

$$Toss_t \supset \mathsf{C} Heads_{t+1} \vee \mathsf{C} \neg Heads_{t+1} \quad (5.81)$$

also works, since it is S5-equivalent to the conjunction of (5.34) and (5.35). But in general such approaches do not translate faithfully into UCL. For instance, if we were to add an action *MaybeFlipUp* to the Suitcase domain, using the UCL schema

$$MaybeFlipUp_t \supset \mathsf{C}\, Up(L_1)_{t+1} \vee \mathsf{C}\, Up(L_2)_{t+1} \quad (5.82)$$

to describe its effects, it would never cause the second latch to go up, when performed alone, if the first was already up.

Lin shows particular interest in two special cases of nondeterministic effects, which he calls "inclusive" and "exclusive." Inclusive nondeterminism corresponds, in the UCL setting, to families of effect axioms of the following form, where A is an action name, and $\sigma^0, \sigma^1, \ldots, \sigma^n$ are fluent designating formulas.

$$A_t \wedge \sigma^0_t \supset \mathsf{C}\sigma^1_{t+1} \vee \cdots \vee \mathsf{C}\sigma^n_{t+1} \quad (5.83)$$

$$A_t \wedge \sigma^0_t \supset \mathsf{C}\sigma^1_{t+1} \vee \mathsf{C}\neg\sigma^1_{t+1} \quad (5.84)$$

$$\vdots$$

$$A_t \wedge \sigma^0_t \supset \mathsf{C}\sigma^n_{t+1} \vee \mathsf{C}\neg\sigma^n_{t+1} \quad (5.85)$$

The first of these axioms, in the presence of the subsequent axioms, can be equivalently replaced (in S5) by

$$A_t \wedge \sigma^0_t \supset \sigma^1_{t+1} \vee \cdots \vee \sigma^n_{t+1}. \quad (5.86)$$

We can also equivalently replace each of the subsequent axioms with the following pair.

$$A_t \wedge \sigma^0_t \wedge \sigma^k_{t+1} \supset \mathsf{C}\sigma^k_{t+1} \quad (5.87)$$

$$A_t \wedge \sigma^0_t \wedge \neg\sigma^k_{t+1} \supset \mathsf{C}\neg\sigma^k_{t+1} \quad (5.88)$$

Notice that these transformations yield formulas in which C occurs at most once, and only positively. Analagous equivalence transformations apply to the corresponding axioms in Lin's theory, given the axioms (5.60) and (5.61) guaranteeing that any fluent literal that is caused is true. These observations show that Lin's proposal for inclusive nondeterminism can be applied in the UCL setting, on the basis of Theorem 5.23. In fact, what we see is that Lin's method for inclusive nondeterminism is essentially a variant of the approach to nondeterminism described briefly in Section 5.7.3. The same observations apply to Lin's proposal for exclusive nondeterminism, which, in the UCL setting, is equivalent to augmenting the inclusive nondeterminism axioms with the additional axiom

$$A_t \wedge \sigma_t^0 \supset \bigwedge_{1 \leq i < j \leq n} \neg(\sigma_{t+1}^i \wedge \sigma_{t+1}^j)\,. \tag{5.89}$$

Ultimately, Lin [Lin96] introduces a more satisfactory general method for formalizing nondeterminism, using auxiliary *Case* symbols to distinguish between possible nondeterministic outcomes. Without going into details, we note that Lin's "cases" method is easily adapted to the UCL setting.

Finally, while Theorem 5.23 is concerned with an embedding of Lin's circumscriptive action theories in UCL, it is interesting to consider also what happens when we proceed in the opposite direction. Recall that a UCL action theory, as described in Section 5.7, typically includes the formulas given by the standard schemas (5.28)–(5.32). Let us assume about such a UCL theory T that it is finite, with a finite signature, and that C is applied only to literals. In this case, the translation $\mathcal{C}(T)$ is defined. Let us assume in addition that the inertial fluents (**I**) are given by the set of fluent names and their negations. If we extend the definition of *cat* so that it applies even when *Caused* is allowed to occur negatively and more than once in each sentence, then it is straightforward to verify that the circumscriptive theory $cat(\mathcal{C}(T))$ is equivalent to $ct(T)$. In light of Theorem 5.22, this observation shows that when we augment Lin's circumscriptive action theories with axioms

corresponding to the standard schemas—so that all models satisfy the principle of universal causation—his approach converges with ours. The principal difference remaining, in the propositional setting, is the ability of the modal language to express directly the fact that a complex formula is caused to hold. In particular, this makes it possible to introduce defined fluents and to express traditional ramification constraints, as described in Section 5.7.3.

In establishing the relationship between Lin's circumscriptive action theories and UCL, it is crucial that we assume that the set of inertial fluents is given by the set of fluent names and their negations. More generally, in UCL action theories, as remarked earlier, the inertial fluents may differ from this. In fact, as demonstrated in Section 5.7.3, it is possible in UCL to generalize the commonsense law of inertia so as to allow for fluents that tend to change in particular ways (instead of tending to remain unchanged).

5.10 UCL and Autoepistemic Logic

It may be interesting to consider briefly the mathematical relationship of UCL to autoepistemic logic, which is surely the most widely-familiar modal nonmonotonic logic. For this purpose we employ the elegant model-theoretic characterization of autoepistemic logic from [LS93].

Let T be an autoepistemic theory. We say that a set S of interpretations is an *AE model* of T if

$$S = \{I : (I, S) \models T\}\,. \tag{5.90}$$

Recall that for autoepistemic logic (AEL) we do not require that structures (I, S) satisfy the condition $I \in S$.

The definition of an AE model can be reformulated as follows. A set S of interpretations is an AE model of an AE theory T if and only if, for all interpreta-

tions I,

$$(I, S) \models T \text{ iff } I \in S\,. \tag{5.91}$$

In this form, we can observe a strong resemblance to the fixpoint condition in UCL, which can be similarly reformulated, as follows. An interpretation I is causally explained by a UCL theory T if and only if, for every set S of interpretations such that $I \in S$,

$$(I, S) \models T \text{ iff } S = \{I\}\,. \tag{5.92}$$

Roughly speaking, the reversal of the roles of S and I in the fixpoint conditions (5.91) and (5.92) is reflected in a corresponding reversal of the role of the modal operator in the two logics. In accordance with this observation, it is not difficult to establish the following.[7]

Proposition 5.25 *Let T be a UCL theory consisting of formulas of the form*

$$\phi \vee \mathsf{C}\psi \tag{5.93}$$

where ϕ and ψ are nonmodal formulas. Take the AEL theory T' obtained by replacing each UCL formula (5.93) *with the AEL formula*

$$\mathsf{B}\phi \vee \psi\,. \tag{5.94}$$

An interpretation I is causally explained by T if and only if $\{I\}$ is an AE model of T'.

We can obtain a more general result of this kind by translating UCL formulas of the form

$$\phi \vee \mathsf{C}\psi^1 \vee \cdots \vee \mathsf{C}\psi^n \tag{5.95}$$

where $\phi, \psi^1, \ldots, \psi^n$ are nonmodal formulas, into autoepistemic formulas

$$\mathsf{B}\phi \vee (\psi^1 \wedge \mathsf{B}\psi^1) \vee \cdots \vee (\psi^n \wedge \mathsf{B}\psi^n)\,. \tag{5.96}$$

[7] We will use the symbol B for the AEL modal operator, rather than $\mathbf{L}$, which is also often used.

In this more complex translation, "caused" becomes, roughly speaking, "truly believed."

It is unclear what lessons to draw from such mathematical facts. Notice that, for AE models of the form $\{I\}$, the fixpoint condition involves only structures of the form $(I', \{I\})$. Therefore, one can, for instance, replace $\mathsf{B}\phi$ with $\neg\mathsf{B}\neg\phi$ without affecting the "complete" AE models. This suggests that the "complete" subset of autoepistemic logic is relatively inexpressive as a logic of belief, as one would intuitively expect.

5.11 UCL with Quantifiers

In this section, we extend UCL to allow first and second-order quantifiers. This makes it possible to write much more compact theories. Second-order quantification in particular is convenient for axiomatizing the structure of situations in action theories. For instance, one can use the Peano axioms, including second-order induction, to characterize completely the structure of situations in linear worlds.

The signature of a (nonpropositional) UCL language is given by a set of nonlogical constants: that is, function symbols (with arities and sorts) and predicate symbols (with arities and sorts). The definitions of a formula, sentence, theory, free occurrence of a variable and so on are as expected.

A UCL *structure* is a pair (I, S) such that S is a set of interpretations with the same universe, and $I \in S$.

In the recursive truth definition, we extend the language each time a quantifier is encountered, adding a new nonlogical constant of the appropriate type (that is, a new function or predicate constant of suitable arity and sort). To this end, we introduce the following auxiliary definition.

Let (I, S) be a UCL structure for a given language. When we add a new nonlogical constant X to the signature, we call a UCL structure (I', S') for the

resulting language an X-*extension* of (I, S) if I' is an extension of I and S' is obtained by taking, for each member of S, the unique extension that interprets X as I' does.

The truth of a UCL sentence in a UCL structure is defined by the standard recursions over the propositional connectives, plus the following four conditions.

$(I, S) \models P$ iff $I \models P$ (for any ground atom P)
$(I, S) \models \mathsf{C}\phi$ iff for all $I' \in S$, $(I', S) \models \phi$
$(I, S) \models \forall x \phi(x)$ iff for every X-extension (I', S') of (I, S), $(I', S') \models \phi(X)$
$(I, S) \models \exists x \phi(x)$ iff for some X-extension (I', S') of (I, S), $(I', S') \models \phi(X)$

Here we assume that X is a new nonlogical constant of the same type as the variable x. By $\phi(X)$ we denote the formula obtained from $\phi(x)$ by simultaneously replacing each free occurrence of x by X.

It is often convenient to designate some nonlogical constants *exempt*, which, intuitively, exempts them from the principle of universal causation. Mathematically, this practice is reflected in the definition of an I-*structure*: a UCL structure (I, S) such that all members of S interpret all exempt symbols exactly as I does. An I-*model* of a UCL theory T is an I-structure that is a model of T.

The definition of a causally explained interpretation is just as it was in the propositional case. An interpretation I is *causally explained* by a UCL theory T if $(I, \{I\})$ is the unique I-model of T.

Clearly, this definition extends the definition introduced in Section 5.2 for propositional UCL, assuming that the nonlogical constants of the language (i.e. the propositional symbols) are not declared exempt.

Before continuing, a few remarks about the truth definition, and in particular the definition of an X-extension, may be in order. Notice that, roughly speaking, the proposed definition of an X-extension makes each newly introduced logical constant exempt from the principle of universal causation, since all members of the second

component of an X-extension of (I, S) interpret X as I does. As we will see in the next section, this is mathematically consistent with the definition of nonpropositional causal theories proposed in [Lif97]. Moreover, such an approach seems to be necessary in order to catch our intended meaning in the case of first-order quantifiers. For example, consider a language with only a unary predicate symbol P in its signature, where P is not declared exempt. Let T be the UCL theory consisting of the following two sentences.

$$\forall x \, (P(x) \supset \mathsf{C}P(x)) \tag{5.97}$$

$$\forall x \, (\neg P(x) \supset \mathsf{C}\neg P(x)) \tag{5.98}$$

We wish to understand these sentences to say that, for every domain object δ, if δ has the property P then that is caused, and if δ does not have the property P then *that* is caused. Thus, our intention is that every interpretation of the language be causally explained by T. Under the proposed definition, this is indeed the case. By comparison, if we were to allow members of the second component of an X-extension of (I, S) to interpret X differently from I, we would find that, in every model of T, either $\forall x P(x)$ or $\forall x \neg P(x)$ holds. Given that the current approach seems correct for first-order quantifiers, it is (mathematically) natural to define truth for second-order quantifiers in essentially the same manner. Perhaps we will eventually learn to do better. In the meantime, it is convenient to have second-order quantifiers available.

As an example, one more version of Lin's Suitcase domain is displayed in Figure 5.12. The signature of the language and the types of variables should be clear from context. All nonlogical constants except *Holds* are exempt. We abbreviate $Succ(s)$ as s'. The axioms for inertia and for the exogeneity of fluents in the initial situation have a different form than previous examples would suggest. (They are equivalent to what one would expect, but shorter.)

$$\forall s(0 \neq s') \wedge \forall s,t(s'=t' \supset s=t) \wedge \forall p\,(p(0) \wedge \forall s(p(s) \supset p(s')) \supset \forall s(p(s)))$$
$$\forall l(l=L_1 \not\equiv l=L_2) \wedge Up(L_1) \neq Up(L_2) \wedge Toggle(L_1) \neq Toggle(L_2)$$
$$\forall f(\exists l(f=Up(l)) \not\equiv f=IsOpen) \wedge \forall a(\exists l(a=Toggle(l)) \not\equiv a=Close)$$
$$\forall f(\mathsf{C}Holds(f,0) \vee \mathsf{C}\neg Holds(f,0))$$
$$\forall s,f\,((Holds(f,s) \equiv Holds(f,s')) \supset \mathsf{C}Holds(f,s') \vee \mathsf{C}\neg Holds(f,s'))$$
$$\forall s,l\,(Occurs(Toggle(l),s) \wedge Holds(Up(l),s) \supset \mathsf{C}\neg Holds(Up(l),s'))$$
$$\forall s,l\,(Occurs(Toggle(l),s) \wedge \neg Holds(Up(l),s) \supset \mathsf{C}Holds(Up(l),s'))$$
$$\forall s\,(Occurs(Close,s) \supset \mathsf{C}\neg Holds(IsOpen,s'))$$
$$\forall s(Holds(Up(L_1),s) \wedge Holds(Up(L_2),s) \supset \mathsf{C}Holds(IsOpen,s))$$

Figure 5.12: Lin's Suitcase domain in second-order UCL.

5.12 Nonpropositional Causal Theories in UCL

Here we review Lifschitz's definition of nonpropositional causal theories [Lif97], altering some terminology and notation to follow more closely the presentation of propositional causal theories in [MT97], which, as we have previously observed, coincide with flat propositional UCL theories. We then sketch a proof of the fact that second-order causal theories are subsumed by second-order UCL.

5.12.1 Lifschitz's Nonpropositional Causal Theories

Begin with a language of classical logic. As in the previous section, some nonlogical constants may be designated exempt. In fact, here we must require that only a finite number of nonlogical constants are not designated exempt. A *causal law* is an expression of the form

$$\phi \Rightarrow \psi \tag{5.99}$$

where ϕ and ψ are formulas of the language. A *causal theory* is a finite set of causal laws. (Except for the finiteness requirements, this definition of a causal theory extends that of [MT97].) In Lifschitz's account, an interpretation is causally

explained by a causal theory just in case it is a model of an associated theory of classical logic.

In what follows, let $\overline{N}$ be a list of all nonexempt nonlogical constants. We say that a list of nonlogical constants or variables is *similar* to $\overline{N}$ if it has the same length as $\overline{N}$ and each of its members is of the same type as the corresponding member of $\overline{N}$. We can denote a formula (in which none, some, or all nonexempt nonlogical constants appear) by $\phi(\overline{N})$. Then for any list $\overline{M}$ that is similar to $\overline{N}$, we can write $\phi(\overline{M})$ to denote the formula obtained by simultaneously replacing each occurrence of each member of $\overline{N}$ by the corresponding member of $\overline{M}$.

Consider a nonpropositional causal theory D with causal laws

$$\phi_1(\overline{N}, x^1) \Rightarrow \psi_1(\overline{N}, x^1) \tag{5.100}$$
$$\vdots$$
$$\phi_k(\overline{N}, x^k) \Rightarrow \psi_k(\overline{N}, x^k) \tag{5.101}$$

where x^i is the list of all free variables for the i-th causal law. Let $\overline{n}$ be a list of new variables that is similar to $\overline{N}$. By $D^*(\overline{n})$ we denote the formula

$$\bigwedge_{1 \leq i \leq k} \forall x^i \left(\phi_i(\overline{N}, x^i) \supset \psi_i(\overline{n}, x^i)\right). \tag{5.102}$$

An interpretation is *causally explained* by D if it is a model of

$$\forall \overline{n} \left(D^*(\overline{n}) \equiv \overline{n} = \overline{N}\right) \tag{5.103}$$

where $\overline{n} = \overline{N}$ stands for the conjunction of the equalities between members of $\overline{n}$ and the corresponding members of $\overline{N}$.

As shown in [Lif97], this definition of a causally explained interpretation extends the definition for propositional causal theories [MT97].

5.12.2 Second-Order Causal Theories in UCL

Here we sketch a proof of the following theorem.

Theorem 5.26 *Let D be a (nonpropositional) causal theory in which all variables are either first or second order, and let T be the UCL theory obtained by replacing each causal law $\phi \Rightarrow \psi$ by the universal closure of the UCL formula $\phi \supset \mathsf{C}\psi$. An interpretation is causally explained by D if and only if it is causally explained by T.*

Proof sketch. Let us write $\forall \overline{n} T^*(\overline{n})$ to denote the sentence (5.103) whose models are the interpretations causally explained by D. Extend the language of $\forall \overline{n} T^*(\overline{n})$ as follows. For every $X \in \overline{N}$, add a new nonlogical constant X' of the same type. Let $\overline{N'}$ be the list of these new symbols, which is similar to $\overline{N}$. Given an interpretation I of the original language, an interpretation J of the new language is called an I-interpretation if J extends I. (That is, if J has the same universe as I and interprets all nonlogical constants in the original language exactly as I does.) The first observation is that $I \models \forall \overline{n} T^*(\overline{n})$ iff, for every I-interpretation J, $J \models T^*(\overline{N'})$. Let $\hat{I}$ be the unique I-interpretation such that for every $X \in \overline{N}$, $(X')^J = X^I$. The sentence $T^*(\overline{N'})$ is an equivalence whose right-hand side is the sentence $\overline{N'} = \overline{N}$. Since $\hat{I}$ is the only I-interpretation that satisfies $\overline{N'} = \overline{N}$, it follows from the previous observation that $I \models \forall \overline{n} T^*(\overline{n})$ iff $\hat{I}$ is the unique I-interpretation satisfying $D^*(\overline{N'})$.

The proof can be completed by showing that $\hat{I}$ is the unique I-interpretation satisfying $D^*(\overline{N'})$ iff $(I, \{I\})$ is the unique I-model of T. The first step in this is to show that $\hat{I} \models D^*(\overline{N'})$ iff $(I, \{I\}) \models T$, which can be done by showing that, for any i, $\hat{I} \models \forall x^i \left(\phi_i(\overline{N}, x^i) \supset \psi_i(\overline{N'}, x^i)\right)$ iff $I \models \forall x^i \left(\phi_i(\overline{N}, x^i) \supset \psi_i(\overline{N}, x^i)\right)$ iff $(I, \{I\}) \models \forall x^i \left(\phi_i(\overline{N}, x^i) \supset \mathsf{C}\psi_i(\overline{N}, x^i)\right)$. It remains only to prove that if $\hat{I} \models D^*(\overline{N'})$, then some I-interpretation other than $\hat{I}$ satisfies $D^*(\overline{N'})$ iff there is a proper superset S of I such that $(I, S) \models T$.

Now we describe observations and a lemma sufficient to complete this last step. First, because C occurs only positively in T, we know that if $(I, S) \models T$, then for any subset S' of S such that $I \in S'$, $(I, S') \models T$. Consequently, one need only consider I-structures of the form $(I, \{I, I'\})$ in order to determine whether $(I, \{I\})$

is the unique I-model of T. This is convenient because there is a natural one-to-one correspondence between I-interpretations and I-structures of the form $(I, \{I, I'\})$ such that, for every I-interpretation J and corresponding I-structure $(I, \{I, I'\})$, for all $X \in \overline{N}$, $(X')^J = X^{I'}$. In light of these observations, we can complete the proof by establishing the following lemma. If $(I, \{I\}) \models T$, then for any I-interpretation J and corresponding I-structure $(I, \{I, I'\})$, $J \models D^*(\overline{N'})$ iff $(I, \{I, I'\}) \models T$. For the proof of this lemma, it is convenient to extend the truth definition for UCL to apply also to structures of the form $(I, \{I'\})$, where I' may differ from I. Under this extended truth definition, one can show that $J \models \forall x^i \left(\phi_i(\overline{N}, x^i) \supset \psi_i(\overline{N'}, x^i)\right)$ if and only if $(I, \{I'\}) \models \forall x^i \left(\phi_i(\overline{N}, x^i) \supset \mathsf{C}\psi_i(\overline{N}, x^i)\right)$. So $J \models D^*(\overline{N'})$ iff $(I, \{I'\}) \models T$. To complete the proof of the lemma, notice that $(I, \{I, I'\}) \models T$ iff both $(I, \{I\}) \models T$ and $(I, \{I'\}) \models T$, since C appears at most once, and only positively, in each sentence of T. □

Chapter 6

Satisfiability Planning with Causal Action Theories

6.1 Introduction

In this chapter, we describe an implemented approach to satisfiability planning [KS92, KS96], which is based on the translation from the "definite" subclass of UCL theories into classical propositional logic that was described in Section 5.6. This material is adapted from [MT98b]. This approach to planning is noteworthy for two reasons. First, it is based on a formalism for describing action domains that is more expressive than the STRIPS-based formalisms traditionally used in automated planning. Secondly, our experiments suggest that the additional expressiveness of causal theories comes with no performance penalty in satisfiability planning. Specifically, in this chapter we show that the large blocks world and logistics planning problems used by Kautz and Selman [KS96] to demonstrate the effectiveness of satisfiability planning can be conveniently represented as causal theories and solved in times comparable to those that they have obtained.

Because UCL theories are more expressive than traditional planning languages, we must consider the preliminary question of when a sequence of actions is a valid plan for achieving a goal G in an initial situation S_0. A valid plan has two fundamental properties: sufficiency and executability. Roughly speaking, a sufficient plan will always achieve G if carried out starting in S_0, and an executable plan can always be carried out starting in S_0. We will make these ideas precise, in the setting of UCL action theories.

We must also consider how to find valid plans by the satisfiability method. Assume that T is a classical propositional theory describing the worlds that are "causally possible" for an action domain. In satisfiability planning, a plan is obtained by extracting the sequence of actions from a model of T that satisfies both the initial state S_0 and the goal G. We will call a plan obtained in this way a causally possible plan, because what we know in this case is simply that there is at least one causally possible world in which the plan achieves G starting in S_0. In order for satisfiability planning to be sound, we must guarantee that the causally possible plans are in fact valid. Accordingly, we define a subclass of definite UCL theories, called "simple," and show that their translations into classical logic are suitable for satisfiability planning. That is, the plans obtained from the models of their translations are not only causally possible, but also deterministic, and thus, as we will show, valid.

The main contributions of the chapter are (1) to provide a theoretical foundation for satisfiability planning on the basis of causal action theories, and (2) to present experimental evidence that the approach is relatively effective. More specifically, we define a family of fundamental properties a plan may have: causally possible, deterministic, sufficient, executable. We say a plan is valid if and only if it is sufficient and executable. We prove that every causally possible, deterministic plan is valid. We then identify a class of "simple" UCL action theories suitable for satisfiability planning. Simple theories have a concise translation into classical logic, and, as we prove, the classical models yield valid plans. Simple theories are very

expressive, thus enabling planning with respect to a wide variety of action domains. We also provide experimental evidence that this planning approach can be very effective on classical problems, by solving, comparatively quickly, the large blocks worlds and logistics planning problems from [KS96].

The chapter is organized as follows. Section 6.2 defines plan validity and related notions for $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions. Section 6.3 defines the class of simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions, and presents the main theorem showing that satisfiability planning is sound for them. Section 6.4 describes an implementation of satisfiability planning with $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions. Section 6.5 reports experimental results on the large blocks world and logistics planning problems from [KS96]. Section 6.6 consists of the proof of the main theorem.

6.2 Planning with $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ Domain Descriptions

In this section we define fundamental notions related to planning, in the setting of $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions.

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description. By an *initial state description* we mean a set S_0 of fluent literals that refer to time 0 such that (1) for every fluent name $F \in \mathbf{F}$, exactly one of $F_0, \neg F_0$ belongs to S_0, and (2) $S_0 \cup D \not\models False$. Intuitively, an initial state description specifies an initial state that occurs in some causally possible world, i.e., a causally possible initial state. By a *time-specific goal*, we simply mean a fluent formula. Notice that a time-specific goal may refer to more than one time. By an *action history* we mean a set P of action literals such that, for every action name $A \in \mathbf{A}$ and time t such that $t+1 \in \mathbf{T}$, exactly one of $A_t, \neg A_t$ belongs to P. Every interpretation includes exactly one action history.

We will define when an action history P is a valid plan for achieving a time-specific goal G in an initial state S_0. This definition rests on the more fundamental notions of sufficiency and executability, which we also define. We define two other

properties of plans, more naturally associated with satisfiability planning. One is determinism. The other is discussed next.

6.2.1 Causally Possible Plans

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is a *causally possible* plan for achieving G in S_0 if

$$S_0 \cup P \cup D \not\models \neg G .$$

This condition says that there is, intuitively speaking, some causally possible world in which G can be achieved by executing P in initial state S_0.

Corollary 5.18 (Section 5.6) yields the following proposition showing that the satisfiability method yields causally possible plans.

Proposition 6.1 *Let D be a definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is included in a model of $lcomp(D) \cup S_0 \cup \{G\}$ if and only if P is a causally possible plan for achieving G in S_0.*

This proposition guarantees that every plan obtained by the satisfiability method is causally possible. Unfortunately, this is a rather weak guarantee. For example, in the nondeterministic Coin Toss domain D_4, introduced in Section 5.7.3 (Figure 5.9), a causally possible plan for having the coin lie heads at time 1, after lying tails at time 0, is simply to toss the coin at time 0. We can make this precise, as follows. There is a single fluent name *Heads*, and a single action name *Toss*. Assume that there are two times, 0 and 1. Take $S_0 = \{\neg Heads_0\}$, $G = Heads_1$, and $P = \{Toss_0\}$. One easily checks that the interpretation $S_0 \cup P \cup \{G\}$ is causally explained by D_4. Hence, P is a causally possible plan for achieving $Heads_1$ in S_0. On the other hand, P is also a causally possible plan for achieving $\neg Heads_1$ in S_0, since $S_0 \cup P \cup \{\neg G\}$ is also causally explained by D_4.

6.2.2 Sufficient Plans

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is a *sufficient* plan for achieving G in S_0 if

$$S_0 \cup P \cup D \mid\!\sim G\,.$$

Intuitively, according to this definition, G will be achieved whenever P is done starting in S_0.

Sufficiency does not say anything about whether P can be executed in S_0, so it is not surprising that some sufficient plans are not valid. In fact, even plans that are both causally possible and sufficient can fail to be valid. Here is an example, again involving coin tossing, along with a second action of truly saying that the coin lies heads. We have a single fluent name, *Heads*, two action names, *Toss* and *TrulySayHeads*, and three times, 0, 1 and 2. Again *Heads* is inertial. The $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description D_8 for this action domain is represented by the schemas of Figure 5.9, along with one additional domain specific schema, shown below.

$$TrulySayHeads_t \supset Heads_t \tag{6.1}$$

Take $S_0 = \{\neg Heads_0\}$, $G = Heads_2$, and

$$P = \{Toss_0, \neg TrulySayHeads_0, \neg Toss_1, TrulySayHeads_1\}\,.$$

So the plan is to toss the coin and then truly say heads. There is exactly one model of $S_0 \cup P$ that is causally explained by D_8—namely, the interpretation $S_0 \cup P \cup \{Heads_1, Heads_2\}$. Therefore, P is a sufficient, causally possible plan for achieving $Heads_2$ in S_0. That is, roughly speaking, there is a causally possible world in which doing P in S_0 achieves $Heads_2$, and, moreover, in any causally possible world in which P is done in S_0, $Heads_2$ is achieved. Nonetheless, P is not a valid plan. Intuitively, the problem is that P is not executable in S_0—it could be that the coin comes up tails after the initial toss, in which case the agent cannot truly say heads at time 1.

6.2.3 Executable Plans

We next define when a plan is executable in an initial state. Unfortunately, this condition is less convenient to state and check than the previous ones.

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description. For any time name $t \in \mathbf{T}$, let $\mathbf{T}|t = \{s \in \mathbf{T} : s \leq t\}$. Given a set X of $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ literals, and a time name t, we write $X|t$ to denote the set of all literals in X that belong to the restricted language $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T}|t)$.

Let P be an action history and S_0 an initial state description. We specify when $P|t$ is executable in S_0 by the following recursive definition. $P|0$ is *executable* in S_0. (Note that $P|0 = \emptyset$.) For all times $t+1 \in \mathbf{T}$, $P|t+1$ is *executable* in S_0 if the following two conditions hold: (i) $P|t$ is executable in S_0, and (ii) for every causally explained interpretation I that satisfies $S_0 \cup P|t$, there is a model of $I|t \cup P|t+1$ that is causally explained. Finally, we say that P itself is *executable* in S_0 if, for every time $t \in \mathbf{T}$, $P|t$ is executable in S_0.

So a plan P is executable if all of its prefixes are. Recall that a prefix $P|t$ completely specifies all action occurrences before time t, and that $P|t+1$ specifies in addition the action occurrences at time t. Thus $P|t+1$ is executable, roughly speaking, if $P|t$ is and, no matter the state of the world after executing $P|t$, the actions specified by P for time t can then be performed.

For example, consider more closely why the plan P from the last example is not executable in S_0. Recall that initially the coin lies tails. The prefix $P|1$ is executable in S_0. That is, it is possible to toss and not truly say heads at time 0. But prefix $P|2$ is not executable in S_0. Intuitively, it may not be possible to truly say heads at time 1. More precisely, notice that the interpretation I obtained from $S_0 \cup P|1$ by adding $\neg Heads_1, \neg Toss_1, \neg TrulySayHeads_1, \neg Heads_2$ is causally explained by D_8, yet no model of $I|1 \cup P|2$ is causally explained. This is because no causally explained interpretation satisfies both $\neg Heads_1$ and $TrulySayHeads_1$.

6.2.4 Valid Plans

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is a *valid* plan for achieving G in S_0 if it is both sufficient and executable.

The next proposition shows that valid plans are causally possible.

Lemma 6.2 *Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description and S_0 an initial state description. If an action history P is executable in S_0, then there is model of $S_0 \cup P$ that is causally explained by D.*

Proof Sketch. The definition of the executability of P in S_0 provides a basis for constructing a causally explained interpretation I such that, for all times $t \in \mathbf{T}$, I satisfies $S_0 \cup P|t$. □

Proposition 6.3 *Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. If P is a valid plan for achieving G in S_0, then it is a causally possible plan for achieving G in S_0.*

Proof. By Lemma 6.2, since P is executable in S_0, some model I of $S_0 \cup P$ is causally explained. Since P is sufficient for G in S_0, $S_0 \cup P \cup D \mid\!\sim G$. Hence, I satisfies G, which shows that $S_0 \cup P \cup D \not\mid\!\sim \neg G$. □

6.2.5 Deterministic Plans

We will define one more class of plans, the deterministic plans. We will show that if a plan is causally possible and deterministic, it is valid. This is a key result in our approach to satisfiability planning. In Section 6.3 we will introduce the class of simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions, and show that for them all causally possible plans are deterministic, and thus valid.

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, P an action history, and S_0 an initial state description. For every time t, $P|t$ is *deterministic* in S_0 if for all fluent names F and times $s \leq t$, $S_0 \cup P|t \cup D \mid\!\sim F_s$ or $S_0 \cup P|t \cup D \mid\!\sim \neg F_s$. We say that P is *deterministic* in S_0 if for every time $t \in \mathbf{T}$, $P|t$ is deterministic in S_0.

Thus a plan P is deterministic if all of its prefixes are. Recall that a prefix $P|t$ is a complete specification of action occurrences for all times before t. Prefix $P|t$ is deterministic if, roughly speaking, performance of the actions in $P|t$ starting in S_0 would completely determine the values of all fluents up to time t.

This definition yields a strong lemma.

Lemma 6.4 *Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description and S_0 an initial state description. If an action history P is deterministic in S_0, then at most one model of $S_0 \cup P$ is causally explained by D.*

Proof. Let I and I' be causally explained models of $S_0 \cup P$. Consider any fluent atom F_t. Since P is deterministic in S_0, so is $P|t$. Since both I and I' satisfy $S_0 \cup P|t$, it follows that they agree on F_t. Hence $I = I'$. □

The converse of Lemma 6.4 does not hold. P may fail to be deterministic in S_0 even when there is at most one causally explained model of $S_0 \cup P$. We illustrate this with another coin tossing example. Take a single fluent name, *Heads*, and three action names, *Toss*, *TrulySayHeads* and *TrulySayTails*. Identify time with the natural numbers. Once more we designate *Heads* inertial. The domain description D_9 is represented by the schemas in Figure 5.9 together with the additional domain specific schema (6.1) from D_8, and two more domain specific schemas, shown below.

$$TrulySayTails_t \supset \neg Heads_t \tag{6.2}$$

$$TrulySayHeads_t \not\equiv TrulySayTails_t \tag{6.3}$$

Due to (6.3), exactly one non-toss action occurs at every time in every causally possible world. Moreover, by (6.1) and (6.2), whenever truly say heads occurs, the coin lies heads, and whenever truly say tails occurs, the coin lies tails. Thus, each causally possible world is completely determined by its initial state and the actions that are performed in it. For instance, let $S_0 = \{\neg Heads_0\}$ and consider the plan P in which the agent initially tosses and concurrently truly says tails, and forever after truly says heads and does not toss. Although exactly one model of $S_0 \cup P$ is causally explained, P is not deterministic. This is because $P|1$ is not deterministic. That is, tossing and concurrently truly saying tails at time 0 simply does not determine the state of the coin at time 1.

Proposition 6.5 *Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. If P is a causally possible plan for achieving G in S_0 and P is also deterministic in S_0, then P is a valid plan for achieving G in S_0.*

Proof. Since P is a causally possible plan for achieving G in S_0, some model I^* of $S_0 \cup P \cup \{G\}$ is causally explained. By Lemma 6.4, no other model of $S_0 \cup P$ is causally explained. Since I^* satisfies G, P is sufficient for achieving G in S_0. To show that P is executable in S_0, we prove by induction that for all times t, $P|t$ is executable in S_0. The base case is trivial. For the inductive step, we show that $P|t+1$ is executable in S_0. By the inductive hypothesis, $P|t$ is executable in S_0. Thus we can complete the proof as follows. Assume that I is a causally explained model of $S_0 \cup P|t$. Notice that both I and I^* satisfy $S_0 \cup P|t$. Since P is deterministic in S_0, so is $P|t$, and it follows that $I^*|t = I|t$. Since I^* also satisfies $P|t+1$, we're done. □

Of course the converse of Proposition 6.5 does not hold, since valid plans need not be deterministic.

6.3 Satisfiability Planning with $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ Domain Descriptions

In this section, we consider how to restrict definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions so that the causally possible plans are deterministic and thus, by Proposition 6.5, valid. To this end, we introduce the class of "simple" $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions.

6.3.1 Simple Domain Descriptions

A definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description D is *simple* if it has the following three (yet to be defined) properties: it is inertially unambiguous, adequately acyclic, and respects the flow of time.

Inertially Unambiguous

Let $\mathbf{F}^+$ denote the set of all fluent atoms that refer to nonzero times. Formulas in D of the form $\phi \wedge L \supset \mathsf{C}L$, where $L \in \mathbf{F}^+$ or $\overline{L} \in \mathbf{F}^+$, and ϕ is any $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ formula, will be called *inertia-like laws*.

Note that this definition covers not only UCL formulas obtained from the standard inertia schema (5.32) but also, for instance, formulas such as those obtained from schemas (5.42)–(5.43) in the Pendulum domain D_3, which describe a dynamic course of nature. This definition also covers formulas such as those obtained from schemas (5.34)–(5.35) in the coin-tossing domains. Although these UCL formulas express the direct nondeterministic effect of the coin-tossing action, they have a form similar to that of inertia laws.

D is called *inertially unambiguous* if it includes no pair of inertia-like laws

$$\phi \wedge F_{t+1} \supset \mathsf{C}F_{t+1} \tag{6.4}$$

$$\psi \wedge \neg F_{t+1} \supset \mathsf{C}\neg F_{t+1} \tag{6.5}$$

such that the formula $\phi \wedge \psi$ is satisfiable.

This exclusivity condition on ϕ and ψ is the only non-syntactic component of the definition of a simple domain description. Notice that the formulas represented by the schema (5.32) for inertia and schemas (5.42)–(5.43) in the Pendulum domain satisfy this condition.

Adequately Acyclic

The *proper atom dependency graph* of D is the directed graph defined as follows. Its nodes are the atoms of the language of D. Let D' be the UCL theory obtained from D by (i) deleting all formulas whose consequent is $\mathsf{C}\mathit{False}$, and (ii) replacing each inertia-like law $\phi \wedge L \supset \mathsf{C}L$ with the UCL formula $\phi \supset \mathsf{C}L$. For each formula in D', there is an edge from the atom that occurs in the consequent to each atom that occurs in the antecedent. We use the proper atom dependency graph to define an ordering on $\mathbf{F}^+$ as follows. For all $A, A' \in \mathbf{F}^+$, $A <_D A'$ if there is a nonempty path from A' to A. (So the edges in the graph point downward in the ordering.) We say that D is *adequately acyclic* if the ordering $<_D$ on $\mathbf{F}^+$ is well-founded.

Intuitively, this condition restricts cyclic causal dependencies between fluents, while allowing cycles that arise due to formulas related to inertia.

Respects the Flow of Time

Here we provide a simpler version (specialized to definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ doman descriptions) of a definition first presented in Section 5.7. We say that D *respects the flow of time* if every formula in D satisfies the following two conditions.

- If the consequent refers to a time t, then the antecedent does not refer to a time later than t.
- If the consequent is a fluent literal that refers to time t, then every action atom in the antecedent refers to a time earlier than t.

Notice that the description D_3 of the Suitcase domain (Figure 5.8), as well as the descriptions D_5 of the Dominos domain (Figure 5.10) and D_6 of the Pendulum domain (Figure 5.11) are all simple domain descriptions.[1] The coin-tossing domains D_4, D_8 and D_9 are not, because they are not inertially unambiguous.[2]

6.3.2 Simple Domain Descriptions Yield Valid Plans

Here is the main technical result related to simple domain descriptions. Its proof is postponed to Section 6.6.

Proposition 6.6 *Let D be a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. If P is a causally possible plan for achieving G in S_0, then P is a valid plan for achieving G in S_0.*

From this result, along with Propositions 6.1 and 6.3, we obtain the following characterization of satisfiability planning with simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain descriptions.

Theorem 6.7 *Let D be a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and G a time-specific goal. An action history P is included in a model of*

$$lcomp(D) \cup S_0 \cup \{G\}$$

if and only if P is a valid plan for achieving G in S_0.

For effective satisfiability planning, we must of course also require that the simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description be finite, with $\mathbf{F}$, $\mathbf{A}$, and $\mathbf{T}$ finite as well.

[1] More precisely, as noted previously, D_5 is easily seen to be S5-equivalent to a simple domain description.

[2] Again, we should note that although D_8 and D_9 are not definite, they are clearly S5-equivalent to definite theories.

```
% File: pendulum
:- declare_types
     type(fluent,[right]),
     type(action,[hold]),
     type(time,[0..4]),
     type(atom,[h(fluent,time),o(action,time)]).
:- declare_variables
     var(A,action),
     var(F,fluent),
     var([T,T1],time).
% domain specific schemas
   o(hold,T) & h(right,T) => h(right,T1)  where T1 is T+1.
   o(hold,T) & -h(right,T) => -h(right,T1) where T1 is T+1.
   -h(right,T) & h(right,T1) => h(right,T1)  where T1 is T+1.
   h(right,T) & -h(right,T1) => -h(right,T1) where T1 is T+1.
% standard schemas
   o(A,T) => o(A,T).     -o(A,T) => -o(A,T).
   h(F,0) => h(F,0).     -h(F,0) => -h(F,0).
```

Figure 6.1: Example input file for the planning system: the Pendulum domain.

6.4 Satisfiability Planning Program

Given a finite signature and a set of schemas representing a finite, definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, it is straightforward to instantiate the schemas to obtain the represented (ground) $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, form its literal completion, and convert it to clausal form. Norm McCain and I (mostly Norm) wrote a Prolog program to carry out these tasks. It includes a procedure named `load_file/1`, which reads in a file such as the one displayed in Figure 6.1 for the Pendulum domain (compare Figure 5.11), and writes out in clausal form the literal completion of the UCL theory. In the input syntax, fluent atoms f_t are represented as `h(`f`,`t`)`, and action atoms a_t are represented as `o(`a`,`t`)`. The symbols `h` and `o` are read as "holds" and "occurs," respectively. Also, we write $\phi \Rightarrow L$ to stand for the UCL formula $\phi \supset \mathsf{C}L$.

After `load_file/1` has processed the domain description, planning problems are posed by calling the procedure `plan/0`, as shown in Figure 2. The procedure `plan/0` reads in an initial state description S_0 and a time-specific goal G, converts them to clausal form, and adds them to the clause set obtained from the domain description.[3] The resulting clause set is simplified, as in [KS96][4], and submitted to the satisfiablity checker *rel_sat* [BS97]. If $lcomp(D) \cup S_0 \cup \{G\}$ is satisfiable, *rel_sat* finds a satisfying interpretation and `plan/0` displays it, answering "yes." The plan P can be read off from this display. By Theorem 6.7, if D is simple, P is guaranteed to be a valid plan for achieving the goal G starting in initial state S_0. If *rel_sat* fails to find a satisfying interpretation, `plan/0` answers "no." Since the solver *rel_sat* is systematic, we know in this case that G cannot be achieved starting in S_0. In either case, the time spent in the solver *rel_sat* is reported.

6.5 Large Planning Problems

Here we report on the performance of our approach when applied to the large blocks world and logistics planning problems from [KS96]. As far as we know, the results obtained there compare favorably with the best current general-purpose planning systems. We obtain comparable results.

6.5.1 Blocks World Problems

The large blocks world planning problems from [KS96] are characterized in Figure 6.3. To provide a rough idea of the quality of our experimental results, we note that Kautz and Selman report for the planner GraphPlan [BF95] a solution time of over 7 hours for Blocks World B (on an SGI Challenge). By comparison, we solve

[3] As illustrated in Figure 6.2, the initial state description S_0 can be replaced by a set Γ of formulas referring only to time 0 such that $\Gamma \cup D \mid\!\sim \phi$, where ϕ is the conjunction of the members of S_0, and yet $\Gamma \cup D \not\mid\!\sim \mathit{False}$.

[4] Steps: subsumption, unit propogation, subsumption.

```
| ?- load_file(pendulum).
% 9 atoms, 28 rules, 16 clauses loaded.
yes
| ?- plan.
enter facts and goal (then ctrl-d)
|: h(right,0).
|: -h(right,2) & h(right,4).
|:
0.  right
Actions: hold
1.  right
Actions:
2.  -right
Actions: hold
3.  -right
Actions:
4.  right
Elapsed Time (cpu sec): 0.01
yes
```

Figure 6.2: Planning session with Pendulum domain.

```
Blocks World A.  9 blocks.  Requires 6 moves.
 Initial state: 2/1/0   4/3   8/7/6/5
 Goal state:    4/0   7/8/3   1/2/6/5
Blocks World B. 11 blocks.  Requires 9 moves.
 Initial state: 2/1/0   10/9/4/3   8/7/6/5
 Goal state:    0/4/9   7/8/3   1/2/10/6/5
Blocks World C. 15 blocks.  Requires 14 moves.
 Initial state: 2/1/0/11/12   10/9/4/3/13/14   8/7/6/5
 Goal state:    13/0/4/9   14/12/7/8/3   11/1/2/10/6/5
Blocks World D. 19 blocks.  Requires 18 moves.
 Initial state: 0/11/12   10/9/4/3/13/14   8/7/6/5   18/17/16/15/2/1
 Goal state:    16/17/18/13/0/4/9   14/12/7/8/3    11/1/2/15/10/6/5
```

Figure 6.3: Characterization of large blocks world problems from [KS96].


```
:- declare_types
     type(block,[0..18]),
     type(location,[block,table]),
     type(fluent,[on(block,location)]),
     type(action,[pickup(block),putat(location)]),
     type(inaction,[nopickup,noputat]),
     type(time,[0..18]),
     type(atom,[o(action,time),o(inaction,time),h(fluent,time)]).
:- declare_variables
     var([B,B1],block),
     var([L,L1],location),
     var(F,fluent),
     var(A,action),
     var(X,inaction),
     var([T,T1],time).
% state constraints: the first two allow concise input of initial state and goal
     h(on(B,L),0) & h(on(B,L1),0) => false  where B \== L, B \== L1, L @< L1.
     h(on(B,L),18) & h(on(B,L1),18) => false  where B \== L, B \== L1, L @< L1.
     h(on(B,B),T) => false.
% direct effects of actions
     o(pickup(B),T) & o(putat(L),T) => h(on(B,L),T1)  where T1 is T+1, B \== L.
     h(on(B,L),T) & o(pickup(B),T) => -h(on(B,L),T1)  where T1 is T+1, B \== L.
% explicit action preconditions
     o(pickup(B),T) & h(on(B1,B),T) => false  where B =\= B1.
     o(putat(B),T) & h(on(B1,B),T) => false  where B =\= B1.
     o(pickup(B),T) & o(putat(B),T) => false.
     o(pickup(B),T) & o(putat(table),T) & h(on(B,table),T) => false.
% at most one move action at a time
     o(pickup(B),T) & o(pickup(B1),T) => false  where B @< B1.
     o(putat(L),T) & o(putat(L1),T) => false  where L @< L1.
     o(pickup(B),T) => -o(nopickup,T).
     o(putat(L),T) => -o(noputat,T).
     o(nopickup,T) & -o(noputat,T) => false.
     -o(nopickup,T) & o(noputat,T) => false.
% standard schemas
     h(F,0) => h(F,0).   -h(F,0) => -h(F,0).
     h(F,T) & h(F,T1) => h(F,T1)  where T1 is T+1.
     -h(F,T) & -h(F,T1) => -h(F,T1)  where T1 is T+1.
     o(A,T) => o(A,T).   -o(A,T) => -o(A,T).   o(X,T) => o(X,T).
```

Figure 6.4: Input file for Blocks World D.

Blocks World B in under a second (on a slower Sparcstation 5).

Our input file representing Blocks World D is displayed in Figure 6.4. We adapt the "operator splitting" approach used by Kautz and Selman. Instead of axiomatizing an action $Move(b, l', l)$, they axiomatize three "component" actions, which we can write: $Pickup(b)$, $Takefrom(l')$, $Putat(l)$. Their axioms are based on Schubert's "explanation closure" [Sch90], augmented with state constraints. In comparison, we introduce names for only two components of the move action: $Pickup(b)$, $Putat(l)$. (When moved, a block is taken from where it currently is.) We also do not introduce a fluent $Clear(b)$. Kautz and Selman include in their description a number of state constraints that we omit.[5] Preliminary experiments indicated that additional state constraints in our blocks world descriptions increase solution times on larger problems.

One can easily verify that the domain description represented in Figure 6.4 is simple. The main complication, compared to our descriptions of the Dominos and Pendulum domains involves action atoms, which are largely irrelevant in determining whether a description is simple. Here we include a family of action atoms that are "true by default" rather than exogenous. Thus, for example, the action $NoPickup$ is assumed to occur, roughly speaking, and we describe the conditions under which it is caused not to occur—whenever $PickUp(B)$ occurs, for some block B. These auxiliary "inaction" atoms are used to stipulate that a pickup action occurs if and only if a putat action does.

[5] Their state constraints still do not rule out all "physically impossible" states. This is in accordance with the usual practice in describing action domains for planning. Roughly speaking, one need only say enough to guarantee that no "illegal" state can be reached from a legal one. Intuitively, this is adequate because planning problems are posed in part by specifying a legal initial state.

6.5.2 Logisitics Planning Problems

The logistics domain is due to Veloso [Vel92]. Kautz and Selman studied three large logistics planning problems. Our input file for the largest of these problems appears in Figure 6.5.

The logistics domain is more complex than the blocks world domain. It includes several kinds of actions that can occur concurrently. Our description of the logistics domain does not use operator splitting (which is not generally applicable to concurrent actions). Preliminary experiments indicated that, in constrast to the blocks world, logistics domain descriptions should include a variety of state constraints in order to get consistently good performance. We note that the logistics domain description used in our experiments is simple, and thus suitable for satisfiablity planning.

6.5.3 Experimental Results

In our experimental results on these planning problems, we report the size of the clausal theory obtained from the literal completion of the causal action theory—in terms of numbers of atoms, clauses and literal occurrences, after simplification—and time spent in the solver, following the reporting methodology of [KS96]. Solution times are averaged over 20 runs of the solver *rel_sat* on a Sparcstation 5, using different random number seeds. Table 6.1 displays statistics for finding plans by our method.

For the sake of comparison, we performed the corresponding experiments on the problem descriptions from [KS96], again using the solver *rel_sat* on a Sparcstation 5.[6] The results appear in Table 6.2. Bayardo and Schrag [1997] showed that,

[6] Kautz and Selman considered two kinds of descriptions of the logistics domains (both in classical propositional logic): one based on intuitions underlying the planner Graphplan [BF95]; the other obtained by first describing the domain as in explanation closure, then eliminating all action atoms. In this second case, a satisfying interpretation does not include an action history. Rather it provides, as it were, a refinement of the planning problem. That is, the satisfying interpretation can be

```
:- declare_types  type(package,[0..6]), type(city,[0..3]), type(airplane,[0..1]),
  type(cityLoc,[p,a]), type(packageLoc,[inPlane(airplane),inVan(city),
    unloaded(city,cityLoc)]), type(inertialFluent,[planeLoc(airplane,city),
    vanLoc(city,cityLoc),at(package,packageLoc)]),
  type(defaultFalseFluent,[nowhere(airplane),misplaced(package)]),
  type(fluent,[inertialFluent,defaultFalseFluent]),
  type(action,[fly(airplane,city),drive(city,cityLoc),
    loadPlane(package,airplane,city), unloadPlane(package,airplane,city),
    loadVan(package,city,cityLoc), unloadVan(package,city,cityLoc)]),
  type(time,[0..13]), type(atom,[o(action,time),h(fluent,time)]).
:- declare_variables  var([T,T1],time), var(If,inertialFluent),
  var(Dff,defaultFalseFluent), var(E,action), var(P,package), var([C,C1],city),
  var([PL,PL1],packageLoc), var([L,L1],cityLoc), var([A,A1],airplane).
h(planeLoc(A,C),T) & h(planeLoc(A,C1),T) => false  where C < C1.
h(nowhere(A),T) => false.  -h(vanLoc(C,a),T) & -h(vanLoc(C,p),T) => false.
h(vanLoc(C,L),T) & h(vanLoc(C,L1),T) => false  where L @< L1.
h(at(P,PL),T) & h(at(P,PL1),T) => false  where PL @< PL1.
h(misplaced(P),T) => false.  h(planeLoc(A,C),T) => -h(nowhere(A),T).
h(at(P,PL),T) => -h(misplaced(P),T).  h(If,0) => h(If,0).
-h(If,0) => -h(If,0).  h(If,T) & h(If,T1) => h(If,T1)  where T1 is T+1.
-h(If,T) & -h(If,T1) => -h(If,T1)  where T1 is T+1.  h(Dff,T) => h(Dff,T).
o(fly(A,C),T) => h(planeLoc(A,C),T1)  where T1 is T+1.
o(fly(A,C),T) => -h(planeLoc(A,C1),T1)  where T1 is T+1, C =\= C1.
o(fly(A,C),T) & h(planeLoc(A,C),T) => false.
o(drive(C,L),T) => h(vanLoc(C,L),T1)  where T1 is T+1.
o(drive(C,L),T) => -h(vanLoc(C,L1),T1)  where T1 is T+1, L \== L1.
o(drive(C,L),T) & h(vanLoc(C,L),T) => false.
o(loadPlane(P,A,C),T) => h(at(P,inPlane(A)),T1)  where T1 is T+1.
o(loadPlane(P,A,C),T) => -h(at(P,unloaded(C,a)),T1)  where T1 is T+1.
o(loadPlane(P,A,C),T) & -h(planeLoc(A,C),T) => false.
o(loadPlane(P,A,C),T) & -h(at(P,unloaded(C,a)),T) => false.
o(loadVan(P,C,L),T) => h(at(P,inVan(C)),T1)  where T1 is T+1.
o(loadVan(P,C,L),T) => -h(at(P,unloaded(C,L)),T1)  where T1 is T+1.
o(loadVan(P,C,L),T) & -h(vanLoc(C,L),T) => false.
o(loadVan(P,C,L),T) & -h(at(P,unloaded(C,L)),T) => false.
o(unloadPlane(P,A,C),T) => h(at(P,unloaded(C,a)),T1)  where T1 is T+1.
o(unloadPlane(P,A,C),T) => -h(at(P,inPlane(A)),T1)  where T1 is T+1.
o(unloadPlane(P,A,C),T) & -h(planeLoc(A,C),T) => false.
o(unloadPlane(P,A,C),T) & -h(at(P,inPlane(A)),T) => false.
o(unloadVan(P,C,L),T) => h(at(P,unloaded(C,L)),T1)  where T1 is T+1.
o(unloadVan(P,C,L),T) => -h(at(P,inVan(C)),T1)  where T1 is T+1.
o(unloadVan(P,C,L),T) & -h(vanLoc(C,L),T) => false.
o(unloadVan(P,C,L),T) & -h(at(P,inVan(C)),T) => false.
o(fly(A,C),T) & o(loadPlane(P,A,C1),T) => false.
o(fly(A,C),T) & o(unloadPlane(P,A,C1),T) => false.
o(drive(C,L),T) & o(loadVan(P,C,L1),T) => false.
o(drive(C,L),T) & o(unloadVan(P,C,L1),T) => false.
o(E,T) => o(E,T).  -o(E,T) => -o(E,T).
```

Figure 6.5: Input file for Logistics C.

Table 6.1: Satisfiability Planning with Causal Action Theories. Sizes are for clausal theories obtained, via literal completion, from causal action theories (after simplification). Time in seconds using the satisfiability solver *rel_sat* on a Sparcstation 5.

Instance	Atoms	Clauses	Literals	Time
BW A	383	2412	5984	0.13
BW B	934	6241	15903	0.81
BW C	2678	18868	48704	35.2
BW D	5745	41726	108267	620.0
LOG A	1643	9205	20712	3.7
LOG B	1760	10746	24134	8.4
LOG C	2300	14450	32346	25.0

Table 6.2: Kautz and Selman Problem Descriptions. Here we establish the benchmarks—the results for the clausal theories used in [KS96], with solution times obtained in the same manner as in Table 6.1.

Instance	Atoms	Clauses	Literals	Time
BW A	459	4675	10809	0.20
BW B	1087	13772	31767	1.4
BW C	3016	50457	114314	66.3
BW D	6325	131973	294118	1052.0
LOG A	1782	20895	42497	2.5
LOG B	2069	29508	59896	9.8
LOG C	2809	48920	99090	32.3

Table 6.3: Proving Plans Optimal: Satisfiability Planning with Causal Action Theories. Here, in each case, the domain description includes one time step less than needed for a solution. Time reported is number of seconds required for solver *rel_sat* to determine unsatisfiability.

Instance	Atoms	Clauses	Literals	Time
BW A	281	1741	4211	0.04
BW B	788	5246	13276	0.43
BW C	2420	17033	43865	21.6
BW D	5343	38795	100544	374.2
LOG A	1354	7378	16595	2.2
LOG B	1498	8908	20026	31.3
LOG C	1946	11924	26710	54.8

for the clausal theories of Kautz and Selman that we consider, their solver *rel_sat* outperforms both of the solvers—one systematic, one stochastic—used in [KS96].[7] Notice that in all cases except Logistics A our solution times are better.

Finally, in order to show that a plan is optimal (in the number of time steps), it is necessary to show that no shorter plan exists. For this purpose it is essential that a systematic solver be used. In Table 6.3, we report on the performance of our approach for this task, again using the solver *rel_sat*. For each problem, we report the time to fail to find a plan one step shorter than the optimal plan. Notice that, for these planning problems, the time needed to fail is comparable to the time needed to succeed.

6.6 Proof of Main Proposition

We begin with the main lemma.

understood as an initial state and goal which together specify completely the values of all fluent atoms. In our reported results, we refer to the first kind of description. We note in comparison that the solver *rel_sat* takes longer for each instance of the second kind of description.

[7] On the other hand, for their description of the logistics domain in which the action names are eliminated, their stochastic solver (properly tuned) is faster than *rel_sat*.

Lemma 6.8 *Let D be a definite $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description that is inertially unambiguous and adequately acyclic. Let P be an action history and S_0 an initial state description. At most one model of $S_0 \cup P$ is causally explained by D.*

Proof. We proceed by the method of contradiction. Suppose that two distinct causally explained interpretations I and I' satisfy $S_0 \cup P$. Let X be the set of atoms on which I and I' disagree. Notice that X is a nonempty subset of $\mathbf{F}^+$, since I and I' differ, and yet agree on all atoms not in $\mathbf{F}^+$. Let X' consist of the members of X that are minimal (among members of X) with respect to the ordering $<_D$. Notice that X' is nonempty, since X is nonempty and $<_D$ restricted to X is well-founded. Finally, let F_{t+1} be a member of X' whose time subscript is minimal (among members of X'). Without loss of generality, assume that $I \models F_{t+1}$ and $I' \models \neg F_{t+1}$. Since $I = D^I$ and $I' = D^{I'}$, there must be a pair of formulas $\phi \supset \mathsf{C}F_{t+1}$ and $\psi \supset \mathsf{C}\neg F_{t+1}$ in D such that $I \models \phi$ but $I' \not\models \phi$, and $I \not\models \psi$ but $I' \models \psi$. It follows that I and I' differ on at least one atom A that occurs in ϕ. Thus, $A \in X$ and also $A <_D F_{t+1}$. Consequently, by the minimality of F_{t+1}, A is F_{t+1}. Since D is adequately acyclic, $\phi \supset \mathsf{C}F_{t+1}$ must be of the form (6.4), and so can be written $\phi' \wedge F_{t+1} \supset \mathsf{C}F_{t+1}$. Since $I \models \phi$, $I \models \phi'$. A similar argument shows that $\psi \supset \mathsf{C}\neg F_{t+1}$ has form (6.5), and can be written $\psi' \wedge \neg F_{t+1} \supset \mathsf{C}\neg F_{t+1}$, with $I' \models \psi'$. Because D is inertially unambiguous, I' cannot satisfy both ϕ' and ψ'. Hence $I' \not\models \phi'$. (We complete the proof by showing that $I' \models \phi'$.) We have already shown that the only atom in ϕ on which I and I' differ is F_{t+1}, which is to say that the only atom in $\phi' \wedge F_{t+1}$ on which I and I' differ is F_{t+1}. Since D is adequately acylic, we know F_{t+1} does not occur in ϕ'. So I and I' agree on all atoms in ϕ', and since $I \models \phi'$, $I' \models \phi'$ as well. Contradiction. □

Let D be an $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description. For any $t \in \mathbf{T}$, let $D|t$ be the UCL theory in the restricted language $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T}|t)$ consisting of all formulas from D in that language.

Observe that if D is a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, then, for every time t, $D|t$ is a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T}|t)$ domain description.

Lemma 6.9 *Let D be a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description and P an action history. For all $t \in \mathbf{T}$, if I is a model of $S_0 \cup P|t$ that is causally explained by D, then $I|t$ is the unique model of $S_0 \cup P|t$ causally explained by $D|t$.*

Proof. Clearly $I|t$ is a model of $S_0 \cup P|t$. Given that D respects the flow of time, one easily verifies that $(D|t)^{I|t} = D^I|t$. Since $I = D^I$, $I|t = D^I|t$. So $I|t = (D|t)^{I|t}$, and we've shown that $I|t$ is a model of $S_0 \cup P|t$ that is causally explained by $D|t$. We know that $D|t$ is simple since D is, so we can conclude by Lemma 6.8 that $I|t$ is unique. □

Lemma 6.10 *Let D be a simple $\mathcal{L}(\mathbf{F},\mathbf{A},\mathbf{T})$ domain description, S_0 an initial state description, and P an action history. If D has a causally explained interpretation satisfying $S_0 \cup P$, then P is deterministic in S_0.*

Proof. We need to show that, for all times $t \in \mathbf{T}$, $P|t$ is deterministic in S_0. Proof is by induction on t. The base case is trivial. By the inductive hypothesis, $P|t$ is deterministic in S_0. Assume that I and I' are models of $S_0 \cup P|t+1$ that are causally explained by D. We need to show that $I|t+1 = I'|t+1$, which follows easily from Lemma 6.9. □

Proposition 6.5 and Lemma 6.10 yield Proposition 6.6.

Chapter 7

Concluding Remarks

This dissertation belongs to a recent line of work in reasoning about action in which causal notions are represented more explicitly than they typically have been in the past. It is important that in this dissertation we do not attempt to formalize assertions of the form "ϕ causes ψ", but instead focus on causal knowledge of a simpler kind: knowledge of the conditions under which facts are caused.

In the first part of the dissertation, we use a simple, well-understood mathematical tool—inference rules—to express "static causal laws" of the form "if ϕ is caused, then ψ is also caused." In Section 3.3 we give a definition of "possible next states" based on this idea, and in Section 3.4 we use that definition as the basis for a high-level action language $\mathcal{AC}$, which incorporates it in a situation calculus setting.

In Section 3.5, we embed $\mathcal{AC}$ in the rule-based nonmonotonic formalism of default logic. The correctness proof for this embedding, presented in Chapter 4, is rather elaborate, and uses so-called Splitting Theorems for default logic, introduced for that purpose. From the embedding in default logic, we derive in Section 3.6 a similar embedding of $\mathcal{AC}$ in logic programming.

The definition of possible next states on which $\mathcal{AC}$ is based reflects a new, causal understanding of commonsense inertia. The embedding of $\mathcal{AC}$ in default logic

shows how to express this causal understanding of inertia by means of default rules of a remarkably simple form. These discoveries contributed to the development of the more satisfactory, general approach described in the second part of dissertation.

The second part of the dissertation discusses UCL, a modal nonmonotonic logic designed specifically for representing the conditions under which facts are caused. On the basis of this mathematically simple form of causal knowledge, UCL characterizes the worlds that are causally possible. The logic takes its name from the principle of universal causation, the simplifying assumption that underlies the fixpoint definition of a causally explained interpretation.

In applications of UCL to reasoning about action, discussed primarily in Sections 5.5, 5.7, and 5.9, universal causation is easily relaxed by means of standard axioms. Also, as introduced in Section 5.11, one can declare a subset of the nonlogical constants exempt from universal causation, as is done in the formalization of the Suitcase domain in second-order UCL (Figure 5.12).

Universal causation plays a key role in the simple, robust solution to the frame problem in Section 5.7. In fact, as illustrated by the pendulum example, essentially the same approach can be used to describe inertia in worlds in which, intuitively speaking, things change (in a certain way) unless they are made not to. For another example of this, imagine a timer that can be reset to zero, but never turned off. One might (partially) describe such a state of affairs in second-order UCL as follows.

$$\mathsf{C}\,Timer(0)=0 \tag{7.1}$$

$$\forall s,n(Timer(s)=n \wedge Timer(s')=n' \supset \mathsf{C}\,Timer(s')=n') \tag{7.2}$$

$$\forall s(ResetTimer(s) \supset \mathsf{C}\,Timer(s')=0) \tag{7.3}$$

(Assume here that the natural numbers are axiomatized as in Figure 5.12, that s' and n' stand for $succ(s)$ and $succ(n)$, and that $ResetTimer$ is declared exempt.)

In Sections 5.4, 5.8, and 5.10, we relate UCL to Reiter's default logic (and, more generally, disjunctive default logic), circumscription, and autoepistemic logic. In Section 5.6, we observe that UCL extends the causal theories formalism of McCain and Turner [MT97], and, in doing so, provides a more adequate semantic account of it. We also introduce the computationally useful class of definite UCL theories. In Section 5.12, we show that (second-order) UCL extends the second-order subset of the nonpropositional causal theories of Lifschitz [Lif97].

We show that UCL can express a variety of causal theories of action and change previously proposed in the literature, including the action language $\mathcal{AC}$ from the first part of the dissertation, as well as the circumscriptive action theories of Lin [Lin95, Lin96]. We also establish, by means of Theorems 5.16 and 5.23, the remarkable similarity between the action theories of Lin and the causal theories of action of [MT97]. Moreover, in light of this, Theorem 5.10 and Proposition 5.25 show how such causal action theories can also be expressed in default logic—as "prerequisite-free" default theories—and in autoepistemic logic.

The third part of the dissertation provides a theoretical foundation for satisfiability planning with UCL theories. In our approach, action domain descriptions expressed as UCL theories are translated into classical propositional logic. The classical models of the translation correspond exactly to the "causally possible" world histories according to the causal theory. Following Kautz and Selman, we then find plans by extracting them from models obtained by satisfiability checking.

In order to establish a basis upon which to judge the soundness of this approach to planning, we define a family of fundamental properties a plan may have: causally possible, deterministic, sufficient, executable. A plan is valid if and only if it is sufficient and executable. We observe that the plans obtained by the satisfiability method may, in general, fail to be sufficient or executable. They are only guaranteed to be causally possible. We show though that any causally possible plan

that is deterministic is also valid.

We identify a class of "simple" domain descriptions for which the satisfiability method is applicable. Simple domain descriptions have a concise translation into classical logic. Moreover, we show that for such domains, the causally possible plans are deterministic and thus valid.

We describe an implemented satisfiability planning system based on these ideas, and provide experimental evidence that the approach can be computationally effective, by solving hard classical planning instances from [KS96] comparatively quickly.

These developments are particularly noteworthy because of the expressive potential of simple UCL theories, as illustrated by the Dominos and Pendulum domains. Thus, future applications of satisfiability planning with causal theories may address extensions to classical planning involving such features as concurrent actions and dynamic worlds.

There remains a great deal of work to do with UCL. We may attempt to automate more expressive subsets of UCL. It would also be interesting to carry out more systematic comparisons with other approaches to planning. This could involve more exhaustive testing of classical planning examples. It could also take the form of an investigation of how well UCL handles various extensions to classical planning, such as concurrent actions. It would also be interesting to look at embeddings in UCL of some of the many other causal approaches to reasoning about action that have been proposed in recent years. Such results can clarify the relationships between the various proposals. They can also help guide future work exploring the range of action domains expressible in UCL and its various sublanguages.

Bibliography

[AB90] Krzysztof Apt and Marc Bezem. Acyclic programs. In *Logic Programming: Proc. of the Seventh Int'l Conf.*, pages 617–633, 1990.

[Ant97] Grigoris Antoniou. A comparison of two approaches to splitting default theories. In *Proc. of AAAI-97*, pages 424–429, 1997.

[Bak91] Andrew Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5–23, 1991.

[Bar94] Chitta Baral. Rule-based updates on simple knowledge bases. In *Proc. of AAAI-94*, pages 136–141, 1994.

[Bar95] Chitta Baral. Reasoning about actions: non-deterministic effects, constraints, and qualifications. In *Proc. of IJCAI-95*, pages 2017–2023, 1995.

[BF87] Nicole Bidoit and Christine Froidevaux. Minimalism subsumes default logic and circumscription. In *Proc. of LICS-87*, pages 89–97, 1987.

[BF95] A. Blum and M.L. Furst. Fast planning through planning graph analysis. In *Proc. IJCAI-95*, pages 1636–1642, 1995.

[BG93] Chitta Baral and Michael Gelfond. Representing concurrent actions in extended logic programming. In *Proc. of IJCAI-93*, pages 866–871, 1993.

[BGP95] Chitta Baral, Michael Gelfond, and Alessandro Provetti. Representing actions I: Laws, observations and hypotheses. In *Working Notes of the AAAI Spring Symposium on Extending Theories of Actions*, 1995.

[BH93] Gerhard Brewka and Joachim Hertzberg. How to do things with worlds: On formalizing actions and plans. *Journal of Logic and Computation*, 3(5):517–532, 1993.

[BM88] Robert Boyer and J Strother Moore. *A Computational Logic Handbook*. Academic Press, 1988.

[BS97] Roberto Bayardo and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proc. of AAAI-97*, pages 203–208, 1997.

[CE92] James Crawford and David Etherington. Formalizing reasoning about change: a qualitative reasoning approach. In *Proc. of AAAI-92*, pages 577–583, 1992.

[Cho94] Pawel Cholewinski. Stratified default logic. In *Computer Science Logic*, pages 456–470. Springer LNCS 933, 1994.

[Cho95] Pawel Cholewinski. Reasoning with stratified default theories. In *Proc. of 3rd Int'l Conf. on Logic Programming and Nonmonotonic Reasoning*, pages 273–286, 1995.

[Cla78] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Dav90] Ernest Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann, 1990.

[DD93] Marc Denecker and Danny DeSchreye. Representing incomplete knowledge in abductive logic programming. In *Logic Programming: Proc. of the 1993 Int'l Symposium*, pages 147–163, 1993.

[Dun93] Phan Minh Dung. Representing actions in logic programming and its applications in database updates. In *Logic Programming: Proc. of the 10th Int'l Conference*, pages 7–25, 1993.

[EK89] Kave Eshghi and Robert Kowalski. Abduction compared with negation as failure. In Giorgio Levi and Maurizio Martelli, editors, *Logic Programming: Proc. of the Sixth Int'l Conf.*, pages 234–255, 1989.

[Elk92] Charles Elkan. Reasoning about action in first-order logic. In *Proc. of the 1992 Canadian Conf. on Artificial Intelligence*, 1992.

[Eva89] Chris Evans. Negation-as-failure as an approach to the Hanks and McDermott problem. In *Proc. of the Second Int'l Symp. on Artificial Intelligence*, 1989.

[Fit85] Melvin Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[FN71] Richard Fikes and Nils Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[Gef89] Hector Geffner. *Default Reasoning: Causal and Conditional Theories*. PhD thesis, UCLA, Department of Computer Science, 1989.

[Gef90] Hector Geffner. Causal theories of nonmonotonic reasoning. In *Proc. of AAAI-90*, pages 524–530, 1990.

[Gef92] Hector Geffner. *Reasoning with defaults: causal and conditional theories.* MIT Press, Cambridge, MA, 1992.

[Gel87] Michael Gelfond. On stratified autoepistemic theories. In *Proc. AAAI-87*, pages 207–211, 1987.

[Gel88] Michael Gelfond. Autoepistemic logic and formalization of commonsense reasoning: Preliminary report. In *Proc. 2nd Int'l Workshop on Non-Monotonic Reasoning*, pages 176–186, 1988.

[GKL95] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Actions with indirect effects (extended abstract). In *Working Notes of the AAAI Spring Symposium on Extending Theories of Action*, pages 80–85, 1995.

[GKL97] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Representing actions: Indeterminacy and ramifications. *Artificial Intelligence*, 95:409–443, 1997.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Logic Programming: Proc. of the Fifth Int'l Conf. and Symp.*, pages 1070–1080, 1988.

[GL90] Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation. In David Warren and Peter Szeredi, editors, *Logic Programming: Proc. of the 7th Int'l Conference*, pages 579–597, 1990.

[GL91] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[GL93] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[GL95] Enrico Giunchiglia and Vladimir Lifschitz. Dependent fluents. In *Proc. IJCAI-95*, pages 1964–1969, 1995.

[GL98] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI-98*, 1998. To appear.

[GLPT91] Michael Gelfond, Vladimir Lifschitz, Halina Przymusińska, and Miroslaw Truszczyński. Disjunctive defaults. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the 2nd Int'l Conference*, pages 230–237, 1991.

[GLR91] Michael Gelfond, Vladimir Lifschitz, and Arkady Rabinov. What are the limitations of the situation calculus? In Robert Boyer, editor, *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pages 167–179. Kluwer Academic, Dordrecht, 1991.

[GP92] Michael Gelfond and Halina Przymusińska. On consistency and completeness of autoepistemic theories. *Fundamenta Informaticae*, XVI:59–92, 1992.

[GS88] Matthew Ginsberg and D.E. Smith. Reasoning about actions II: The qualification problem. *Artificial Intelligence*, 35:311–342, 1988.

[Gus96] P. Gustaffson, J. and Doherty. Embracing occlusion in specifying the indirect effects of actions. In *Principles of Knowledge Representation and Reasoning: Proc. of the Fifth Int'l Conference*, 1996.

[Haa87] Andrew Haas. The case for domain-specific frame axioms. In Frank M.

Brown, editor, *The Frame Problem in Artificial Intelligence, Proc. of the 1987 Workshop*, 1987.

[Hau87] Brian Haugh. Simple causal minimizations for temporal persistence and projection. In *Proc. AAAI-87*, pages 218–223, 1987.

[HC68] G.E. Hughes and M.J. Cresswell. *An introduction to modal logic.* Methuen and Co LTD, 1968.

[HM87] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

[Kar93] G. Neelakantan Kartha. Soundness and completeness theorems for three formalizations of action. In *Proc. of IJCAI-93*, pages 724–729, 1993.

[Kar94] G. Neelakantan Kartha. Two counterexamples related to Baker's approach to the frame problem. *Artificial Intelligence*, 69:379–391, 1994.

[KL94] G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects (preliminary report). In *Proc. of the Fourth Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 341–350, 1994.

[KL95] G. Neelakantan Kartha and Vladimir Lifschitz. A simple formalization of actions using circumscription. In *Proc. IJCAI-95*, pages 1970–1975, 1995.

[KM91] Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the 2nd Int'l Conference*, pages 387–394, 1991.

[Kom90] Jan Komorowski. Towards a programming methodology founded on partial deduction. In *Proc. of ECAI-90*, 1990.

[Kow74] Robert Kowalski. Predicate logic as a programming language. *Information Processing*, 75:569–574, 1974.

[KS86] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–9–5, 1986.

[KS92] Henry Kautz and Bart Selman. Planning as satisfiability. In J. Lloyd, editor, *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI 92)*, pages 359–379, Vienna, Austria, 1992.

[KS94] Robert Kowalski and Fariba Sadri. The situation calculus and event calculus compared. In *Logic Programming: Proc. of the 1994 Int'l Symposium*, pages 539–553, 1994.

[KS96] Henry Kautz and Bart Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*, pages 1194–1201, 1996.

[Kun87] Kenneth Kunen. Negation in logic programming. *Journal of Logic Programming*, 4:289–308, 1987.

[Lif85] Vladimir Lifschitz. Computing circumscription. In *Proc. of IJCAI-85*, pages 121–127, 1985.

[Lif87a] Vladimir Lifschitz. Formal theories of action. In Frank M. Brown, editor, *The Frame Problem in Artificial Intelligence, Proc. of the 1987 Workshop*, pages 35–58, 1987.

[Lif87b] Vladimir Lifschitz. On the semantics of STRIPS. In Michael Georgeff and Amy Lansky, editors, *Reasoning about Actions and Plans*, pages 1–9. Morgan Kaufmann, San Mateo, CA, 1987.

[Lif90] Vladimir Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[Lif91] Vladimir Lifschitz. Towards a metatheory of action. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proc. of the Second Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 376–386, 1991.

[Lif93a] Vladimir Lifschitz. Circumscription. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *The Handbook of Logic in AI and Logic Programming*, volume 3, pages 298–352. Oxford University Press, 1993.

[Lif93b] Vladimir Lifschitz. Restricted monotonicity. In *Proc. AAAI-93*, pages 432–437, 1993.

[Lif95] Vladimir Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 74:351–365, 1995.

[Lif97] Vladimir Lifschitz. On the logic of causal explanation. *Artificial Intelligence*, 96:451–465, 1997.

[Lin95] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of IJCAI-95*, pages 1985–1991, 1995.

[Lin96] Fangzhen Lin. Nondeterminism in causal theories of action. In *Proc. of AAAI-96*, pages 670–676, 1996.

[LR89] Vladimir Lifschitz and Arkady Rabinov. Miracles in formal theories of actions. *Artificial Intelligence*, 38(2):225–237, 1989.

[LR94] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994. Special Issue on Actions and Processes.

[LS91] Fangzhen Lin and Yoav Shoham. Provably correct theories of action (preliminary report). In *Proc. AAAI-91*, pages 349–354, 1991.

[LS93] Vladimir Lifschitz and Grigori Schwarz. Extended logic programs as autoepistemic theories. In Luis Moniz Pereira and Anil Nerode, editors, *Logic Programming and Non-monotonic Reasoning: Proceedings of the Second Int'l Workshop*, pages 101–114, 1993.

[LT94] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In Pascal Van Hentenryck, editor, *Proc. Eleventh Int'l Conf. on Logic Programming*, pages 23–37, 1994.

[LT95] Vladimir Lifschitz and Hudson Turner. From disjunctive programs to abduction. In Jürgen Dix, Luis Pereira, and Teodor Przymusinski, editors, *Non-Monotonic Extensions of Logic Programming (Lecture Notes in Artificial Intelligence 927)*, pages 23–42. Springer-Verlag, 1995.

[McC59] John McCarthy. Programs with common sense. In *Proc. of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London, 1959. Her Majesty's Stationery Office. Reproduced in [McC90].

[McC80] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1, 2):27–39,171–172, 1980. Reproduced in [McC90].

[McC86] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 26(3):89–116, 1986. Reproduced in [McC90].

[McC90] John McCarthy. *Formalizing common sense: papers by John McCarthy.* Ablex, Norwood, NJ, 1990.

[MD80] Drew McDermott and Jon Doyle. Nonmonotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[MH69] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969. Reproduced in [McC90].

[Mil95] Rob Miller. Situation calculus specifications for event calculus logic programs. In *Proc. of the 3rd Int'l Conf. on Logic Programming and Nonmonotonic Reasoning*, pages 217–230, 1995.

[Moo85] Robert Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[Mor88] Paul Morris. The anomalous extension problem in default reasoning. *Artificial Intelligence*, 35(3):383–399, 1988.

[MS88] Karen Myers and David Smith. The persistence of derived information. In *Proc. AAAI-88*, pages 496–500, 1988.

[MT93a] Wiktor Marek and Miroslaw Truszczyński. *Nonmonotonic Logic: Context-Dependent Reasoning*. Springer-Verlag, 1993.

[MT93b] Wiktor Marek and Miroslaw Truszczyński. Revision programming. Manuscript, 1993.

[MT94] Wiktor Marek and Miroslaw Truszczyński. Revision specifications by means of programs. In *Logics in AI. Proceedings of JELIA '94*, 1994.

[MT95a] Wiktor Marek and Miroslaw Truszczyński. Revision programming, database updates and integrity constraints. In *Proc. of the 5th Int'l Conf. on Database Theory*, pages 368–382, 1995.

[MT95b] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *Proc. of IJCAI-95*, pages 1978–1984, 1995.

[MT97] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. of AAAI-97*, pages 460–465, 1997.

[MT98a] Wiktor Marek and Miroslaw Truszczyński. Revision programming. *Theoretical Computer Science*, 190:241–277, 1998.

[MT98b] Norman McCain and Hudson Turner. Satisfiability planning with causal theories. In *Principles of Knowledge Representation and Reasoning: Proc. of the Sixth Int'l Conference*, 1998. To appear.

[Ped89] Edwin Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In Ronald Brachman, Hector Levesque, and Raymond Reiter, editors, *Proc. of the First Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 324–332, 1989.

[Pop94] Sally Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.

[PR93] Javier Pinto and Ray Reiter. Temporal reasoning in logic programming: A case for the situation calculus. In *Logic Programming: Proc. of the Tenth Int'l Conf.*, pages 203–217, 1993.

[Prz88] Teodor Przymusinski. On the relationship between logic programming and non-monotonic reasoning. In *Proc. AAAI-88*, pages 444–448, 1988.

[PT95] Teodor Przymusinski and Hudson Turner. Update by means of inference rules. In *Proc. of the 3rd Int'l Conf. on Logic Programming and Nonmonotonic Reasoning*, pages 156–174, 1995.

[PT97] Teodor Przymusinski and Hudson Turner. Update by means of inference rules. *Journal of Logic Programming*, 30(2):125–143, 1997.

[Rei80] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1,2):81–132, 1980.

[Rei91] Raymond Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[San94] Erik Sandewall. *Features and Fluents.* Oxford University Press, 1994.

[San96] Erik Sandewall. Assessments of ramification methods that use static domain constraints. In *Principles of Knowledge Representation and Reasoning: Proc. of the Fifth Int'l Conference*, 1996.

[Sch90] Lenhart Schubert. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H.E. Kyburg, R. Loui, and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, 1990.

[Sho87] Yoav Shoham. *Reasoning about change.* MIT Press, Boston, MA, 1987.

[Sub93] Sakthi Subramanian. *A Mechanized Framework for Specifying Problem Domains and Verifying Plans.* PhD thesis, University of Texas, Austin, Department of Computer Science, 1993.

[Thi94] Michael Thielscher. Representing actions in equational logic programming. In Pascal Van Hentenryck, editor, *Logic Programming: Proc. of the 11th Int'l Conference*, pages 207–224. MIT Press, 1994.

[Thi95a] Michael Thielscher. Computing ramifications by postprocessing. In *Proc. of IJCAI-95*, pages 1994–2000, 1995.

[Thi95b] Michael Thielscher. The logic of dynamic sytems. In *Proc. of IJCAI-95*, pages 1956–1962, 1995.

[Thi97] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

[Tur94] Hudson Turner. Signed logic programs. In Maurice Bruynooghe, editor, *Logic Programming: Proc. of the 1994 Int'l Symposium*, pages 61–75, 1994.

[Tur96a] Hudson Turner. Representing actions in default logic: A situation calculus approach. In *Working Papers of the Third Symposium on Logical Formalizations of Commonsense Reasoning*, 1996.

[Tur96b] Hudson Turner. Splitting a default theory. In *Proc. of AAAI-96*, pages 645–651, 1996.

[Tur97] Hudson Turner. Representing actions in logic programs and default theories: A situation calculus approach. *Journal of Logic Programming*, 31(1–3):245–298, 1997.

[Tur98] Hudson Turner. A logic of universal causation. *Artificial Intelligence*, 1998. To appear.

[vB88] Johan van Bentham. *A Manual of Intensional Logic.* CSLI, 1988. Second edition, revised and expanded.

[vBDS95] Christof van Bellingham, Marc Denecker, and De Schreye. Combining situation calculus and event calculus. In *Proc. of ICLP-95*, 1995.

[Vel92] Manuela Veloso. *Learning by Analogical Reasoning in General Problem Solving*. PhD thesis, CMU, 1992. CS Technical Report CMU-CS-92-174.

[VGRS90] Allen Van Gelder, Kenneth Ross, and John Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, pages 221–230, 1990.

[Win88] Marianne Winslett. Reasoning about action using a possible models approach. In *Proc. of AAAI–88*, pages 89–93, 1988.

Vita

Charles Hudson Turner was born on September 5, 1959 in Providence, Rhode Island to parents Charles and Clarice Turner. He grew up in Texas mostly, graduating from Plano Sr. High School in 1977. After attending Rice University and the University of Texas at Austin, each for a year, he worked for two years at Texas Instruments in Dallas, Texas. He then returned to school, first at the University of Texas at Dallas for one year, and then at the University of Texas at Austin, where he received a B.A. in Liberal Arts in 1984. Soon after graduation, he married Carol George, who remains his dear wife. In 1996, their first child, a son, was born.

Beginning in 1984, he worked in the University of Texas Undergraduate Library in various capacities, shelving books, manning the circulation desk and later the reference desk. During this time he earned a Masters degree in Library and Information Science (M.L.I.S.) from UT Austin (1988), and continued working in the Undergraduate Library, as a professional ("microcomputer applications") librarian, until 1991.

He began studying computer science in 1988, first by enrolling in undergraduate courses and reading on his own. In 1991 he completed a Masters degree (M.S.C.S.) in Computer Science. During the years of his doctoral work, he held an MCD fellowship and an IBM fellowship, and also worked as a graduate research assistant for his advisor, Vladimir Lifschitz, and as a teaching assistant for a variety of graduate and undergraduate computer science courses. During the last four

years, he has presented a number of technical papers at international conferences. He has published two journal papers to date, with another to appear later this year. He is scheduled to join the faculty of the Department of Computer Science at the University of Minnesota, Duluth in September.

Permanent Address: Hudson Turner
1614 W. 8th
Austin, TX 78703

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin.