A SIMPLE DECLARATIVE LANGUAGE FOR DESCRIBING NARRATIVES WITH ACTIONS

To appear in the Journal of Logic Programming Special Issue on Reasoning about Actions (Spring 1997)

ANTONIOS KAKAS AND ROB MILLER

▷ We describe a simple declarative language $\mathcal{E}$ for describing the effects of a series of action occurrences within a narrative. $\mathcal{E}$ is analogous to Gelfond and Lifschitz's Language $\mathcal{A}$ and its extensions, but is based on a different ontology. The semantics of $\mathcal{E}$ is based on a simple characterisation of persistence which facilitates a modular approach to extending the expressivity of the language. Domain descriptions in $\mathcal{A}$ can be translated to equivalent theories in $\mathcal{E}$. We show how, in the context of reasoning about actions, $\mathcal{E}$'s narrative-based ontology may be exploited in order to characterise and synthesise two complementary notions of explanation. According to the first notion, explanation may be partly modelled as the process of suitably extending an apparently inconsistent theory written in $\mathcal{E}$ so as to establish consistency, thus providing a natural method, in many cases, to account for conflicting sets of information about the domain. According to the second notion, observations made at later times can sometimes be explained in terms of what is true at earlier times. This enables domains to be given an alternative characterization in which knowledge arising from observations is appropriately separated from other aspects of the domain. We also describe how $\mathcal{E}$ domains may be implemented as Event Calculus style logic programs, which facilitate automated reasoning both backwards and forwards in time, and which behave correctly even when the knowledge entailed by the domain description is incomplete. ◁

Address correspondence to ANTONIOS KAKAS, Deptment of Computer Science, University of Cyprus, 75 Kallipoleos Street, P.O. Box 537, CY-1678 Nicosia, CYPRUS. Email: antonis@turing.cs.ucy.ac.cy *or* ROB MILLER, Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, U.K. Email: rsm@doc.ic.ac.uk, http://www-lp.doc.ic.ac.uk/UserPages/staff/rsm/rsm.html

An earlier version of this paper (August 1995) appeared as a joint Imperial College / University of Cyprus Research Report (Imperial College Research Report Number DoC 95/12)

1. INTRODUCTION

This paper largely concerns narrative reasoning, that is, reasoning about actions which actually occur at various times, and reasoning about the properties that hold or do not hold at different times as a consequence of such occurrences. The importance of narrative reasoning has been recognised elsewhere (see [2], [34] or [31]). For example, to deal with *observations* a formalism must allow the representation of a narrative, since phenomena can only be 'observed' at actual times. In the context of the Situation Calculus it makes little sense to state that Fred is observed to be alive in the 'hypothetical' or 'projected' situation

$$Result(Shoot, Result(Wait, Result(Load, S0)))$$

unless there is some extra mechanism to relate the sequence of actions $Load$ - $Wait$ - $Shoot$ to the time at which the observation took place.

For the purposes of discussion, we will informally define a *narrative-based formalism* as a formalism in which the structure or flow of time is represented independently from the notion of an action, and in which actions are 'embedded' in this independent structure using an explicit notion of an action *occurrence*. Examples of such formalisms are Allen's interval-based approach [1], Sandewall's *Features and Fluents* framework [36] (where a series of action occurrences are captured as a *schedule*) and Kowalski and Sergot's Event Calculus [24]. On the other hand, formalisms such as McCarthy and Hayes' Situation Calculus [29], Dynamic Logics (see for example [18]) and Gelfond and Lifschitz's Language $\mathcal{A}$ [16] are not narrative-based. This is not to say that they cannot be extended to deal with narrative information (see for example [34], [31] or [26]). But the notions of an independent flow of time and of an action occurrence are not central to their underlying ontology.

In [16] and [17], Gelfond and Lifschitz proposed a particular methodology for research into reasoning about action. The authors introduced the Language $\mathcal{A}$ as a "simple declarative language for describing actions", and suggested that, by describing general translation procedures from $\mathcal{A}$ domains into other formalisms, an insight could be gained into the comparative possibilities and limitations of each approach. The success of this idea is exemplified in a paper by Kartha [21], which uses translations from $\mathcal{A}$ to show the equivalence of three well-known characterisations of the Situation Calculus for a whole class of domains. The primary intention of $\mathcal{A}$ was to provide a language and semantics simple enough to be regarded as uncontroversial and intuitive, even if (initially) somewhat limited in expressivity. Of course, the 'neutrality' of any such language, used as a measuring stick for other formalisms, will inevitably be compromised to some extent by the necessity of choosing a particular ontology as a starting point. The ontology underlying the Language $\mathcal{A}$ is inherited from the Situation Calculus.

This paper shows that the methodology described above need not be limited to this particular ontology. Our central aim is to propose and develop a simple declarative language for describing narratives, called $\mathcal{E}$, based not upon the ontology of the Situation Calculus, but instead upon a narrative-based ontology similar to that of the Event Calculus. Furthermore, we aim to use $\mathcal{E}$ as a specification for developing logic programs for automated reasoning about action and change in a principled manner. We believe that the use of, and comparison between, different ontologies is vital in the study of reasoning about action. Central issues such as the *frame problem*, the *ramification problem* and the *qualification problem* all take on different flavours when set in different ontological contexts. Comparisons between approaches can help reveal which aspects of these problems are fundamental, and which are merely the product of a particular method of representation.

In particular, the notion of *persistence* (or 'inertia') is somewhat different in a narrative setting. A simple declarative characterisation of persistence is central to the semantics of $\mathcal{E}$ described below. This semantics, like that of $\mathcal{A}$, is model-theoretic, and the characterisation of persistence is achieved by listing three specific conditions which each model must satisfy. The definition of a model is intended to be modular in the sense that, in future extensions to $\mathcal{E}$, further conditions or constraints not relating to persistence may simply be added. (To illustrate this point, two simple extensions to $\mathcal{E}$ are given in an appendix.) The semantics gives rise to a notion of entailment which is independent from any particular method of derivation or computation. Thus the Language $\mathcal{E}$ helps differentiate between the ontological commitments of the Event Calculus and the computational mechanisms provided by its original logic programming setting. In this respect it serves a purpose similar to that of the formalisms described in [40], [41] and [7].

The paper is organised as follows. In Section 2, we describe the basic syntax and semantics of $\mathcal{E}$, give some examples and discuss some general properties of the formalism. In Section 3 we show a correspondence between the Languages $\mathcal{A}$ and $\mathcal{E}$ by describing a sound and complete translation from theories written in $\mathcal{A}$ into Language $\mathcal{E}$ domain descriptions. We also briefly discuss the relationship between $\mathcal{E}$ and the Language $\mathcal{L}_0$, a narrative extension of $\mathcal{A}$ recently proposed by Baral, Gelfond and Provetti [5] [6]. In the next three sections, we use $\mathcal{E}$ to characterise two complementary notions of *explanation* in temporal domains. In Section 4, we show how explanation may be partly modelled as the process of extending an apparently inconsistent theory written in $\mathcal{E}$ so as to establish consistency. In particular, we show that the syntax and semantics of $\mathcal{E}$ allows a class of 'narrative-based' explanations to be identified in a natural way. We also show how different preference relations between explanations can be combined with the simple object-level definition of entailment described previously in order to define a meta-level semantics with both an abductive and a deductive flavour. In Sections 5 and 6 we show that observations made at later times may also sometimes be explained in terms of what is true at earlier times. To do this, we identify a special class of $\mathcal{E}$ domain descriptions, and separate out observations from these theories. In Section 7 we describe how $\mathcal{E}$ domains may be implemented as Event Calculus style logic programs. We do this in a way which avoids potential problems when the information entailed by the domain is 'incomplete', which could otherwise be caused by logic programming's implicit completion of the *HoldsAt* predicate. These programs also encode a limited form of reasoning backwards in time. Finally, in Section 8 we show how, for a particular class of domains, we may use some of the notions of explanation developed earlier to add a meta-level component to these implementations, in order to facilitate a more 'complete' form of automated temporal reasoning.

2. A CLASS OF LANGUAGES FOR DESCRIBING NARRATIVES OF ACTION OCCURRENCES

First, we will describe the basic syntax of the Language $\mathcal{E}$. Strictly speaking, $\mathcal{E}$ represents a family of languages, all of which use a basic ontology and vocabulary of fluents (properties), actions and time points. The progression of time is represented by an ordering relation over the set of time points. Time may either be continuous or progress via discrete steps, and need not necessarily be linear.

Definition 2.1. [Domain Language] A *domain language* is a tuple $\langle \Pi, \preceq, \Delta, \Phi \rangle$, where $\preceq$ is a partial (possibly total) ordering[1] defined over the non-empty set Π of *time points*, Δ is a non-empty set of *action constants*, and Φ is a non-empty set of *fluent constants.*

Except where the context implies otherwise, for the remainder of the paper we assume a particular domain language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$. We will often write $T_1 \prec T_2$ to mean $T_1 \preceq T_2$ and $T_1 \neq T_2$.

Definition 2.2. [Fluent literal] A *fluent literal* of $\mathcal{E}$ is an expression either of the form F or of the form $\neg F$, where $F \in \Phi$.

Three types of statements are possible within $\mathcal{E}$. *C-propositions* ("c" for "causes") express the conditions under which particular actions can potentially initiate or terminate periods (i.e. intervals) in which a property holds. *H-propositions* ("h" for "happens") indicate that a particular action occurs at a particular time, and *t-propositions* ("t" for "time point") express that a particular property holds at a particular time.

Definition 2.3. [c-proposition] A *c-proposition* in $\mathcal{E}$ is an expression either of the form

$$A \textbf{ initiates } F \textbf{ when } C$$

or of the form

$$A \textbf{ terminates } F \textbf{ when } C$$

where $F \in \Phi$, $A \in \Delta$, and C is a set of fluent literals of $\mathcal{E}$.

Notation:
We shall often write c-propositions of the form "A **initiates** F **when** $\emptyset$" and "A **terminates** F **when** $\emptyset$" as "A **initiates** F" and "A **terminates** F" respectively.

Definition 2.4. [h-proposition] An *h-proposition* in $\mathcal{E}$ is an expression of the form

$$A \textbf{ happens-at } T$$

where $A \in \Delta$ and $T \in \Pi$.

Definition 2.5. [t-proposition] A *t-proposition* in $\mathcal{E}$ is an expression of the form

[1] We mean 'partial ordering' in the usual mathematical sense, i.e. $\preceq$ is reflexive, transitive and anti-symmetric. $\preceq$ should not be regarded as representing partial knowledge about a total order – our formalism would have to be modified to cope with this type of incomplete information. Although it might be argued that time is in fact linear, so that $\preceq$ should always be a total ordering, we consider partial orderings here for the sake of mathematical generality, and because in Section 3 a particular partially ordered set is useful in showing a correspondence between the Languages $\mathcal{E}$ and $\mathcal{A}$.

$$L \textbf{ holds-at } T$$

where L is a fluent literal of $\mathcal{E}$ and $T \in \Pi$.

Definition 2.6. [Domain description] A *domain description* in $\mathcal{E}$ is a triple $\langle \gamma, \eta, \tau \rangle$, where γ is a set of c-propositions, η is a set of h-propositions[2], and τ is a set of t-propositions in $\mathcal{E}$.

The semantics for $\mathcal{E}$ is expressed by defining the notion of an *interpretation*, and stating when an interpretation qualifies as a model for a given domain description. In the definitions below, an interpretation is defined simply as a mapping of fluent/time-point pairs to *true* or *false* (i.e. a *holds* relation). A model is an interpretation that respects four properties. The first three of these are intended to characterise a 'commonsense' notion about the persistence of properties as time progresses. In particular, they encapsulate the idea that all points at which a property starts (ceases) to hold are earmarked by an initiating (terminating) action occurrence – in other words, actions are the only mechanisms for change[3]. This is stated explicitly in condition (1) of Definition 2.10. Conditions (2) and (3) confirm that the terms *initiate* and *terminate* have their intended meanings, relative to this 'commonsense' principle.

Definition 2.7. [Interpretation] An *interpretation* of $\mathcal{E}$ is a mapping

$$H : \Phi \times \Pi \mapsto \{true, false\}$$

Definition 2.8. [Point satisfaction] Given a set of fluent literals C of $\mathcal{E}$ and a time point $T \in \Pi$, an interpretation H *satisfies* C *at* T iff for each fluent constant $F \in C$, $H(F,T) = true$, and for each fluent constant F' such that $\neg F' \in C$, $H(F',T) = false$.

Definition 2.9. [Initiation/termination point] Let H be an interpretation of $\mathcal{E}$, let $D = \langle \gamma, \eta, \tau \rangle$ be a domain description, let $F \in \Phi$ and let $T \in \Pi$. T is an *initiation-point* (respectively *termination-point*) *for* F *in* H *relative to* D iff there is an $A \in \Delta$ such that (i) there is both an h-proposition in η of the form "A **happens-at** T" and a c-proposition in γ of the form "A **initiates** F **when** C" (respectively "A **terminates** F **when** C") and (ii) H satisfies C at T.[4]

Definition 2.10. [Model] Given a domain description $D = \langle \gamma, \eta, \tau \rangle$ in $\mathcal{E}$, an interpretation H of $\mathcal{E}$ is a *model* of D iff, for every $F \in \Phi$ and $T, T', T_1, T_3 \in \Pi$ such

[2]Thus, since η can include more than one h-proposition that refers to the same time point, $\mathcal{E}$ allows for a limited form of concurrency (its semantics forces the assumption that concurrently performed actions do not interfere, for example to cancel each other's effects). In contrast $\mathcal{A}$ does not allow any concurrency, whereas its extension $\mathcal{A}_c$ [4] allows for a more general form of concurrency.

[3]This principle might be considered too strong for some domains, e.g. those involving continuous change. In this case, some distinction will be required between those properties which naturally persist (*frame fluents* in Lifschitz's terminology) and those which do not.

[4]According to this definition, not all initiation and termination points will be points of change of the fluent within a particular model. For example, if a fluent already holds before an initiation-point it will remain unchanged. Thus in Sergot's terms [37] the semantics uses "weak" initiation and termination (but see footnote, Appendix A.1).

that $T_1 \prec T_3$, the following properties hold:

1. If there is no initiation-point or termination-point T_2 for F in H relative to D such that $T_1 \preceq T_2 \prec T_3$, then $H(F,T_1) = H(F,T_3)$.

2. If T_1 is an initiation-point for F in H relative to D, and there is no termination-point T_2 for F in H relative to D such that $T_1 \prec T_2 \prec T_3$, then $H(F,T_3) = true$.

3. If T_1 is a termination-point for F in H relative to D, and there is no initiation-point T_2 for F in H relative to D such that $T_1 \prec T_2 \prec T_3$, then $H(F,T_3) = false$.

4. For all t-propositions in τ of the form "F **holds-at** T", $H(F,T) = true$, and for all t-propositions of the form "$\neg F$ **holds-at** T'", $H(F,T') = false$.

Definition 2.11. [Consistency] A domain description is *consistent* iff it has a model.

Definition 2.12. [Entailment] A domain description D *entails* the t-proposition "F **holds-at** T", written "$D \models F$ **holds-at** T", iff for every model H of D, $H(F,T) = true$. D entails the t-proposition "$\neg F$ **holds-at** T" iff for every model H of D, $H(F,T) = false$.

In keeping with our adopted methodology, two important simplifying assumptions are implicitly included in the above semantics. First, the information about the general effects of actions, expressed as c-propositions, is assumed to be complete. An analogous assumption is made about the e-propositions in a Language $\mathcal{A}$ theory. Second, the information about the occurrence of actions, expressed as h-propositions, is also assumed to be complete. (There is no directly analogous assumption in the definition of a Language $\mathcal{A}$ model, since the notion of an action occurrence is not included in its ontology.) Clearly, both these assumptions will be sources of nonmonotonicity in the language. The h-propositions not only give complete information about which actions occur, but (since $\preceq$ is assumed to be well-defined) also give complete information about the order and timing of these occurrences[5].

Condition (4) in Definition 2.10 above expresses pointwise constraints on a model which arise from the inclusion of t-propositions in the domain description. We envisage other such constraints being added in future, more expressive extensions of $\mathcal{E}$. Such extensions might also require refinements to the definitions of an interpretation or of an initiation or termination point. But we expect the characterisation of persistence encapsulated in conditions (1)-(3), which can be regarded as the 'core' of the semantics, to remain unaltered. To remain faithful to the methodology we are using, we wish here to keep the syntax and semantics of $\mathcal{E}$ as simple as possible, even at the loss of some expressivity. However, to illustrate this modular aspect of

[5]Relaxing these assumptions would be an interesting area for future research, but would inevitably lead to a more complex semantics. For example, it would be straightforward to allow for incomplete knowledge about the order and timing of action occurrences by using *temporal variables* in h-propositions, and including a fourth type of proposition in domain descriptions with which to express ordering constraints between these variables. An interpretation would then include a mapping (i.e. variable assignment) from temporal variables to time points as a second component.

the semantics we have included two simple extensions to $\mathcal{E}$ in Appendix A (relating to 'qualifications' and 'ramifications' of action occurrences).

The following two examples illustrate the effects of our model-theoretic characterisation of persistence.

Example 2.1. This example is intended to illustrate the necessity of including the first condition in Definition 2.10 of a Language $\mathcal{E}$ model above. It concerns vaccinations against a particular disease. Vaccine A only provides protection for people with blood type O, and vaccine B only works on people with blood type other than O. Fred's blood type is not known, so he is injected with vaccine A at 2 o'clock and vaccine B at 3 o'clock. For simplicity we will model time as the real number line with the usual ordering relation, so that for this example, $\mathcal{E}_v = \langle \Re, \leq, \{InjectA, InjectB\}, \{Protected, TypeO\}\rangle$. The domain description D_v consists of two c-propositions, two h-propositions and a single t-proposition:

$$InjectA\ \textbf{initiates}\ Protected\ \textbf{when}\ \{TypeO\}$$

$$InjectB\ \textbf{initiates}\ Protected\ \textbf{when}\ \{\neg TypeO\}$$

$$InjectA\ \textbf{happens-at}\ 2$$

$$InjectB\ \textbf{happens-at}\ 3$$

$$\neg Protected\ \textbf{holds-at}\ 1$$

If we now consider some time later than 3 o'clock, say 4 o'clock, we can see intuitively that Fred should be protected. Now by condition (1), in all models of D_v Fred's blood group remains constant, so that in any given model, by condition (2), Fred becomes protected either at 2 o'clock or at 3 o'clock. Consequently,

$$D_v \models Protected\ \textbf{holds-at}\ 4$$

Had condition (1) not been included in the definition of a model, it would have been possible to construct a model, for example, in which Fred's blood type "mysteriously" changed from $\neg TypeO$ to $TypeO$ at 2.30, thus rendering both vaccinations ineffective.

Example 2.2. This example shows that the Language $\mathcal{E}$ can be used to infer information about what conditions hold at the time of an action occurrence, given other information about what held at times before and afterwards. It is similar to Baker's 'murder mystery' scenario [3]. Let $\mathcal{E}_{ys} = \langle \Re^+, \leq, \{Shoot\}, \{Alive, Loaded\}\rangle$, where $\Re^+$ signifies the non-negative real numbers, and let the domain description D_{ys} consist of a single c-proposition, a single h-proposition and two t-propositions:

$$Shoot\ \textbf{terminates}\ Alive\ \textbf{when}\ \{Loaded\}$$

$$Shoot\ \textbf{happens-at}\ 2$$

$$Alive\ \textbf{holds-at}\ 1$$

$$\neg Alive\ \textbf{holds-at}\ 3$$

Since by condition (4) in any model H of D_{ys}, $H(Alive, 1) \neq H(Alive, 3)$, then by condition (1), in all models an action must occur at some time point between 1 and 3 whose initiating or terminating conditions for the property *Alive* are satisfied at that point. The only candidate is the *Shoot* occurrence at 2, whose condition for terminating *Alive* is *Loaded*. Hence

$$D_{ys} \models \mathit{Loaded}\ \textbf{holds-at}\ 2$$

Indeed, by applying condition (1) again, it is easy to see that for all n, $n \geq 0$,

$$D_{ys} \models \mathit{Loaded}\ \textbf{holds-at}\ n$$

Two properties of $\mathcal{E}$ will prove useful later. The first is that $\mathcal{E}$ is monotonic as regards addition of t-propositions to domain theories (although, as observed earlier, not as regards addition of h-propositions or c-propositions). That is to say, if H is a model of a domain description $\langle \gamma, \eta, \tau \rangle$ and $\tau' \subseteq \tau$, H is also a model of $\langle \gamma, \eta, \tau' \rangle$. This follows directly from Definition 2.10. The second property of interest concerns the deterministic nature of actions' effects within a narrative, and is somewhat analogous to Lin and Shoham's notion of *epistemological completeness* [27]. Provided the domain description under consideration is consistent and contains no finite intervals of time in which an infinite number of actions occur, then the set of fluents which hold at any point T completely determines the set of fluents which hold at any later time point. This claim is made precise in the following definition and proposition.

Definition 2.13. [Occurrence Sparsity] Let $D = \langle \gamma, \eta, \tau \rangle$ be a domain description written in a language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$. D and η are *occurrence-sparse* iff for any two points $T_1, T_2 \in \Pi$ there are only a finite number of h-propositions in η of the form "A **happens-at** T" such that $T_1 \preceq T \prec T_2$.

Proposition 2.1. Let D be an occurrence-sparse domain description written in a language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$, and let $T_1, T_2 \in \Pi$ be such that $T_1 \preceq T_2$. Let H and H' be models of D such that for all $F \in \Phi$, $H(F, T_1) = H'(F, T_1)$. Then for all $F \in \Phi$, $H(F, T_2) = H'(F, T_2)$.

Proof. See Appendix B.1

The occurrence of infinite numbers of actions in a finite period of time leads to interesting and/or unexpected results in many formalisms for reasoning about action. For a general discussion of this complex issue, the reader may refer to Davis [9]. However, for the remainder of this paper, we restrict our attention to domain descriptions which are occurrence-sparse, and thus deterministic in the sense of Proposition 2.1.

3. SIMULATING THE LANGUAGE $\mathcal{A}$ AS A CLASS OF LANGUAGE $\mathcal{E}$ DOMAINS

Because they use different ontologies, any correspondence between the Languages $\mathcal{A}$ and $\mathcal{E}$ must inevitably be expressed in rather artificial terms. In this section we show that Language $\mathcal{A}$ domains can be simulated or re-formulated as Language $\mathcal{E}$ domains with a branching structure $\langle \Pi, \preceq \rangle$ of time points, analogous to the branching 'tree' of situations often incorporated in formulations of the Situation Calculus. Since the Language $\mathcal{A}$ is not narrative-based and so does not directly include the notion of action occurrences, in our re-formulation an appropriate action occurrence has to be 'built in' at each point in this tree structure. In situation calculus terms, we must ensure that for each situation S in the tree, the action A 'occurs' between the situations S and $Result(A, S)$. To express this, we need to insert an extra time-point between S and $Result(A, S)$ – in graphical terms, we need to be able to refer to the arcs connecting the situation nodes in the tree. This is achieved in a simple way below by considering 'doubled' sequences of actions. Sequences of even length correspond to nodes of the tree structure (i.e. to situations), and sequences of odd length correspond to the inter-connecting arcs. A temporal ordering relation is then defined on both nodes and arcs.

Definition 3.1. [Δ-sequence] Given a set Δ of action constants, a Δ-*sequence* of Δ is defined inductively as follows:

- The empty sequence $\langle\langle\ \rangle\rangle$ is a Δ-sequence
- For each $A, A' \in \Delta$, the singleton sequence $\langle\langle |A| \rangle\rangle$, the sequence $\langle\langle |A|, A \rangle\rangle$ and the sequence $\langle\langle |A|, A, |A'| \rangle\rangle$ are all Δ-sequences
- For each $A_1, \ldots, A_n \in \Delta$, $\langle\langle |A_1|, A_1, \ldots, |A_n|, A_n \rangle\rangle$ is a Δ-sequence
- For each $A' \in \Delta$ and for each Δ-sequence $\langle\langle |A_1|, A_1, \ldots, |A_n|, A_n \rangle\rangle$, $\langle\langle |A_1|, A_1, \ldots, |A_n|, A_n, |A'| \rangle\rangle$ is a Δ-sequence

Definition 3.2. [Δ-ordering] Given a set Δ of action constants and the corresponding set Π_Δ of all Δ-sequences, the Δ-*ordering* $\leq_\Delta$ over Π_Δ is defined as follows

- For all $S \in \Pi_\Delta$, $\langle\langle\ \rangle\rangle \leq_\Delta S$
- For all $\langle\langle \alpha_1, \ldots, \alpha_n \rangle\rangle \in \Pi_\Delta$ and for all m such that $1 \leq m \leq n$, $\langle\langle \alpha_1, \ldots, \alpha_m \rangle\rangle \leq_\Delta \langle\langle \alpha_1, \ldots, \alpha_n \rangle\rangle$

Examples:
Suppose $\Delta = \{Wait, Load, Shoot\}$. The Δ-sequence

$$\langle\langle |Load|, Load, |Wait|, Wait, |Shoot|, Shoot \rangle\rangle$$

corresponds to the Situation Calculus term

$$Result(Shoot, Result(Wait, Result(Load, S0)))$$

and (regarding situations as arranged in a branching tree structure) the Δ-sequence

$$\langle\langle |Load|, Load, |Wait|, Wait, |Shoot| \rangle\rangle$$

corresponds to the inter-connecting arc between the situations

$$Result(Wait, Result(Load, S0))$$

and

$$Result(Shoot, Result(Wait, Result(Load, S0)))$$

Regarding the ordering $\leq_\Delta$, it is easy to see that, for example

$$\langle\langle |Load| \rangle\rangle \ \leq_\Delta \ \langle\langle |Load|, Load \rangle\rangle$$

$$\langle\langle |Load|, Load \rangle\rangle \ \leq_\Delta \ \langle\langle |Load|, Load, |Wait|, Wait, |Shoot|, Shoot \rangle\rangle$$

Notation:
We shall sometimes refer to the Δ-sequence

$$\langle\langle |A_1|, A_1, \ldots, |A_n|, A_n \rangle\rangle$$

simply as $A_1, \ldots, A_n$ and refer to the Δ-sequence

$$\langle\langle |A_1|, A_1, \ldots, |A_n|, A_n, |A'| \rangle\rangle$$

as $A_1, \ldots, A_n, |A'|$. Notice that in this notation

$$A_1, \ldots, A_n, |A'| \ \leq_\Delta \ A_1, \ldots, A_n, A'$$

The next definition allows us to express that, in general, the action A' occurs at $A_1, \ldots, A_n, |A'|$, so that the effects of A' are apparent at the following time point $A_1, \ldots, A_n, A'$.

Definition 3.3. [Complete occurrence set] Let $\mathcal{E} = \langle \Pi_\Delta, \leq_\Delta, \Delta, \Phi \rangle$, where Π_Δ is the set of Δ-sequences of Δ and $\leq_\Delta$ is the Δ-ordering over Π_Δ. The set η_Δ, called the *complete occurrence set* of Δ, is the set of all h-propositions of $\mathcal{E}$ either of the form

$$A \ \textbf{happens-at} \ \langle\langle |A| \rangle\rangle$$

or of the form

$$A_n \ \textbf{happens-at} \ \langle\langle |A_1|, A_1, \ldots, |A_{n-1}|, A_{n-1}, |A_n| \rangle\rangle$$

The following proposition shows a sense in which the Language $\mathcal{A}$ may be regarded as a special case of the Language $\mathcal{E}$.

Proposition 3.1. Let D_A be a consistent theory written in a language $\mathcal{A}$ in the sense of [16], with a set of action constants Δ and a set of fluent constants Φ. Let $\mathcal{E} = \langle \Pi_\Delta, \leq_\Delta, \Delta, \Phi \rangle$, where Π_Δ is the set of Δ-sequences of Δ and $\leq_\Delta$ is the Δ-ordering over Π_Δ. Let D_E be the domain description $\langle \gamma, \eta_\Delta, \tau \rangle$ in $\mathcal{E}$ defined as follows:

- *η_Δ is the complete occurrence set of Δ*
- *For each v-proposition in D_A of the form "L **after** $A_1; \ldots; A_m$" there is a t-proposition in τ of the form "L **holds-at** $A_1, \ldots, A_m$", and for each v-proposition in D_A of the form "**initially** L" there is a t-proposition in τ of the form "L **holds-at** $\langle\langle\ \rangle\rangle$"*
- *For each $F \in \Phi$, then for each e-proposition in D_A of the form "A **causes** F **if** $L_1, \ldots, L_n$" there is a c-proposition in γ of the form "A **initiates** F **when** $\{L_1, \ldots, L_n\}$"*
- *For each $F \in \Phi$, then for each e-proposition in D_A of the form "A **causes** $\neg F$ **if** $L_1, \ldots, L_n$" there is an c-proposition in γ of the form "A **terminates** F **when** $\{L_1, \ldots, L_n\}$"*

Then for each $F \in \Phi$ and each $A_1, \ldots, A_n \in \Delta$

- $D_E \models F$ **holds-at** $\langle\langle\ \rangle\rangle$ *if and only if*
  D_A *entails* **initially** F
- $D_E \models \neg F$ **holds-at** $\langle\langle\ \rangle\rangle$ *if and only if*
  D_A *entails* **initially** $\neg F$
- $D_E \models F$ **holds-at** $A_1, \ldots, A_n$ *if and only if*
  D_A *entails* F **after** $A_1; \ldots; A_n$
- $D_E \models \neg F$ **holds-at** $A_1, \ldots, A_n$ *if and only if*
  D_A *entails* $\neg F$ **after** $A_1; \ldots; A_n$

PROOF. Let R be the unique transition function such that there is a model (σ_0, R) of D_A (for definitions see [16]). Let $\sigma \subseteq \Phi$ be a set of fluent constants. For each $F \in \Phi$ and $A_1, \ldots, A_n, A' \in \Delta$ let the interpretation $H_{[\sigma,R]}$ be defined as follows:

- $H_{[\sigma,R]}(F, \langle\langle\ \rangle\rangle) = H_{[\sigma,R]}(F, \langle\langle |A'| \rangle\rangle) = true$ if and only if $F \in \sigma$
- $H_{[\sigma,R]}(F, \langle\langle |A_1|, A_1, \ldots, |A_n|, A_n \rangle\rangle) =$
  $H_{[\sigma,R]}(F, \langle\langle |A_1|, A_1, \ldots, |A_n|, A_n, |A'| \rangle\rangle) = true$
  if and only if $F \in R(A_n, R(A_{n-1}, \ldots, R(A_1, \sigma) \ldots))$

Clearly, for each $\sigma \subseteq \Phi$, $H_{[\sigma,R]}$ is a model of $\langle \gamma, \eta, \emptyset \rangle$, and $H_{[\sigma,R]}$ is a model of $D_E = \langle \gamma, \eta, \tau \rangle$ if and only if (σ, R) is a model of D_A in the sense of [16]. Since D_E is occurrence-sparse, by Proposition 2.1 all models of D_E are of this form, so that there is a one-to-one correspondence between models of D_A and such models of D_E, and the proposition follows directly from the definition of $H_{[\sigma,R]}$.

We conclude this section with some brief remarks about the relationship between $\mathcal{E}$ and the Language $\mathcal{L}_0$ recently introduced by Baral, Gelfond and Provetti in [5], and discussed in more detail in [6]. $\mathcal{L}_0$ is a 'narrative' extension to $\mathcal{A}$ which uses $\mathcal{A}$'s underlying Situation Calculus based ontology. Conceptually, it is close to the extension of the Situation Calculus described by Pinto and Reiter in [34]. Both are concerned with describing and reasoning about an 'actual path' through the 'tree of situations'.

A model of an $\mathcal{L}_0$ domain description is a pair (Ψ, Σ). The "Ψ" component roughly corresponds to the notion of a (partial) "transition function" in a Language $\mathcal{A}$ model. For a given domain, Ψ is characterised by a collection of *effect laws* in $\mathcal{L}_0$, which is equivalent to a set of e-propositions in $\mathcal{A}$ and may be translated into a Language $\mathcal{E}$ domain description of the form $\langle \gamma, \eta_\Delta, \tau \rangle$ as described in Proposition 3.1. Given such a translation, the "Σ" component of an $\mathcal{L}_0$ model can be regarded as an assignment of the *situation* symbol s_N (always included in $\mathcal{L}_0$'s vocabulary) to a particular Δ-sequence δ_N in Π_Δ, together with an assignment of each other situation symbol s_i to a Δ-sequence δ_i such that $\delta_i \leq_\Delta \delta_N$. (In Pinto and Reiter's terms, δ_N represents the 'actual path of situations'.) To be acceptable, the assignment δ_N must be of minimal length, subject to certain constraints which are expressed as *fluent facts*, *occurrence facts* and *precedence facts*. For example, the occurrence fact "A_1, A_2 **occurs-at** s_i" constrains δ_N to be of the form $\langle\langle \alpha_1, \ldots, \alpha_i, |A_1|, A_1, |A_2|, A_2, \ldots, \alpha_n \rangle\rangle$ (where $\langle\langle \alpha_1, \ldots, \alpha_i \rangle\rangle$ is the Δ-sequence δ_i assigned to s_i), and the precedence fact "s_i **precedes** s_j" expresses the constraints $\delta_i \leq_\Delta \delta_j$ and $\delta_i \neq \delta_j$. The minimal length requirement for δ_N corresponds to the minimisation of the *occurs* predicate in Pinto and Reiter's extended Situation Calculus.

A number of other papers which discuss the relationship between the Situation Calculus and the Event Calculus have recently been published, see for example [25], [30], [35] and [44].

4. DEALING WITH INCONSISTENT DOMAINS BY EXPLANATIONS

Like the Language $\mathcal{A}$, the Language $\mathcal{E}$ provides a very rigid way of specifying the effects of actions. Each separate effect has to be explicitly described by a c-proposition (analogous to an e-proposition in the Language $\mathcal{A}$), and the definition of entailment does not facilitate the inference of any other effects. For example, without an explicit representation of the statement "the action occurrence A terminates the property P" the statement "P is true before A, but false afterwards" gives rise to an inconsistency. This is in contrast to some other approaches to representing persistence, such as representations in circumscribed predicate calculus, which in this respect are more flexible. The Language $\mathcal{E}$ is similarly rigid in its representation of a narrative – all action occurrences must be explicitly represented by an h-proposition.

However, greater flexibility can be achieved, without altering the underlying semantics of $\mathcal{E}$, by introducing the notion of an *explanation*. Clearly, the statement that "property P is true before A, but false afterwards" is easily 'explained' by the statement "A terminates P". In this section we model the task of explanation as the task of restoring consistency, in some principled or selective way, to an inconsistent collection of facts, represented as an inconsistent domain description in $\mathcal{E}$. Under this view, explanation is a form of belief revision. The nonmonotonicity built into our language sometimes allows an inconsistent domain theory to be 'revised' simply by adding sets of propositions. We will call such sets 'explanations'. In the present context, explanations may be either in narrative or in causal terms, or in both. In other words, a given set of facts may be explained away in terms of what has happened and/or in terms of what causes what.

In Sections 5 and 6 we will extend the notion of an explanation, showing how in the context of narrative reasoning it is sometimes appropriate to regard information about what holds at earlier times as an 'explanation' of what holds at later times.

4.1. Explanations in Terms of Action Occurrences

The first class of explanations we will consider are those which can be expressed entirely in terms of action occurrences, i.e. extra h-propositions. We will use a version of Kautz's Stolen Car problem [23] as an illustration.

Example 4.1. Let $\mathcal{E}_{sc} = \langle \mathcal{N}, \leq, \{Park, Steal\}, \{Parked\} \rangle$, where $\mathcal{N}$ signifies the natural numbers, and let D_{sc} be the domain description consisting of the following two c-propositions, single h-proposition and single t-proposition:

$$Park \ \textbf{initiates} \ Parked$$

$$Steal \ \textbf{terminates} \ Parked$$

$$Park \ \textbf{happens-at} \ 2$$

$$\neg Parked \ \textbf{holds-at}\ 6$$

By itself, D_{sc} is inconsistent, since there is no terminating action occurrence for the fluent $Parked$ between 2 and 6. However, we may restore consistency by adding one or more h-propositions.

Definition 4.1. [h-explanation] Let $D = \langle \gamma, \eta, \tau \rangle$ be a domain description. An *h-explanation* for D is a (possibly empty) occurrence-sparse set η_ϵ of h-propositions, such that $\langle \gamma, \eta \cup \eta_\epsilon, \tau \rangle$ is consistent.

For example, the following are all h-explanations for D_{sc}:

$$\{\ Steal\ \textbf{happens-at}\ 3,\ Steal\ \textbf{happens-at}\ 4\ \}$$

$$\{\ Steal\ \textbf{happens-at}\ 4,\ Park\ \textbf{happens-at}\ 8\ \}$$

$$\{\ Steal\ \textbf{happens-at}\ 5\ \}$$

Indeed, it is obvious that we may construct an h-explanation for D_{sc} containing as many h-propositions as we like. Clearly, extra mechanisms are needed which enable us to prefer some explanations to others. The following definition reflects a very simple, set-theoretic notion of preference. However, the definition could be modified for specific domains, for example to reflect the fact that we wish to regard some types of occurrence as more likely than others.

Definition 4.2. [Preferable h-explanation][6]
Let η_ϵ and η'_ϵ be h-explanations for D. η_ϵ is *preferable* to η'_ϵ iff $\eta_\epsilon \subset \eta'_\epsilon$.

Having identified a class of explanations, such as the class of h-explanations in Definition 4.1, and a preference criterion such as that of Definition 4.2, it is possible to construct a corresponding meta-level, explanation-based 'semantics' similar to the semantics of Abductive Logic Programming [19], [20], [13]. We consider all possible extensions of a given domain description D with optimal explanations, and accept conclusions if and only if these hold in each such extension. This semantics can then be used to decide what can be safely 'inferred' from a seemingly inconsistent domain description.

Definition 4.3. [Optimal h-explanation] η_ϵ is an *optimal h-explanation* for D iff η_ϵ is an h-explanation for D and there is no other h-explanation η'_ϵ for D such that η'_ϵ is preferable to η_ϵ.

Definition 4.4. [h-model] Let $D = \langle \gamma, \eta, \tau \rangle$ be a domain description. H is an *h-model* of D iff there exists an optimal h-explanation η_ϵ for D such that H is a model of $\langle \gamma, \eta \cup \eta_\epsilon, \tau \rangle$

Definition 4.5. [h-consistency] A domain description is *h-consistent* iff it has at least one h-model.

[6] In this definition and throughout the paper "$\subset$" is intended to mean "is contained in and not equal to".

Definition 4.6. [h-entailment] The domain description $D = \langle \gamma, \eta, \tau \rangle$ *h-entails* the t-proposition "F **holds-at** T", written "$D \models_h F$ **holds-at** T", iff for every h-model H of D, $H(F, T) = true$. D h-entails the t-proposition "$\neg F$ **holds-at** T" iff for every h-model H of D, $H(F, T) = false$.

H-entailment is an abductive notion in the sense that optimal h-explanations are not derived from or entailed by a domain description, but are added to it (according to an external preference criterion). It also has a 'deductive flavour' in the sense that a t-proposition is entailed from a domain description simply if it is true in all h-models. And any procedure for verifying truth in all h-models will, explicitly or implicitly, have to take into account *all* optimal h-explanations.

The important point is that h-entailment is a meta-level concept, whereas the entailment relation defined in Section 2 is object-level. H-entailment is defined both in terms of object-level entailment and a particular preference criterion among explanations. Notice that Definitions 4.3 to 4.6 would still be applicable even if 'h-preferability' were to be defined differently (perhaps according to domain-specific considerations), but would yield different results.

A desirable property of any such meta-level entailment is that it should coincide with object-level entailment whenever the domain description is consistent. The following proposition shows that for h-entailment as defined Definition 4.6 above, this is indeed the case.

Proposition 4.1. Let D be a consistent domain description. Then H is a model of D if and only if H is an h-model of D.

PROOF. The proposition follows directly from the observation that since D is already consistent, it has a unique optimal h-explanation $\emptyset$.

Returning to the Stolen Car problem, D_{sc} has three optimal h-explanations:

$$\{\ Steal\ \textbf{happens-at}\ 3\ \}$$

$$\{\ Steal\ \textbf{happens-at}\ 4\ \}$$

$$\{\ Steal\ \textbf{happens-at}\ 5\ \}$$

D_{sc} is therefore h-consistent, and has a total of six h-models (two corresponding to each optimal h-explanation), since in any h-model H, $H(Parked, 0)$, $H(Parked, 1)$ and $H(Parked, 2)$ may either all be $true$ or all be $false$. It is easy to see, therefore, that

$$D_{sc} \models_h Parked\ \textbf{holds-at}\ 3$$

and for all $n \geq 6$,

$$D_{sc} \models_h \neg Parked\ \textbf{holds-at}\ n$$

Hence this example illustrates how the notion of h-entailment exploits the narrative ontology of $\mathcal{E}$ to give a natural and simple way to handle such apparently inconsistent domains.

H-entailment corresponds closely to Shanahan's formulation of explanation-based temporal reasoning in [38]. The main difference is that whereas Shanahan's concern is to abduce a single explanation for a given fact or observation, ours is to 'safely' infer new information by considering all (optimal) explanations. Shanahan's work was partly based on earlier work by Eshghi [15] showing how *planning* could be formulated within an abductive Event Calculus framework. In Language $\mathcal{E}$ terms, an initial state I and goal state G can both be represented as sets τ_I and τ_G of t-propositions, domain information can be modelled as a set γ of c-propositions, and a plan can be regarded as a single h-explanation η_e for $\langle\gamma, \emptyset, \tau_I \cup \tau_G\rangle$ such that $\langle\gamma, \eta_e, \tau_I\rangle \models \omega$ for each $\omega \in \tau_G$.

Our notion of an h-model is also somewhat analogous to the description of a Language $\mathcal{L}_0$ model given by Baral, Gelfond and Provetti in [5] and [6] (see end of Section 3). An important difference here is that, whereas the requirement that action occurrences be minimal is 'hardwired' into $\mathcal{L}_0$'s object-level semantics, our minimality requirement is to be found in the particular definition of h-preference, which could potentially be replaced by or extended with other, domain-specific preference criteria. Indeed, in the following section we give an example of a preference criterion among a class of explanations which is not entirely based around a simple notion of minimality.

4.2. Explanations in Terms of New Causal Rules

It will not always be possible to find an h-explanation for an inconsistent domain description D. In this section we briefly explore the possibilities of including both c-propositions and h-propositions in explanations. The discussion here is intended only to illustrate some of the problems relating to this – we are obviously trespassing into the related A.I. topic of *learning*. We will use the following (deliberately abstract) example to motivate the discussion:

Example 4.2. Let $\mathcal{E}_{ex} = \langle\mathcal{N}, \leq, \{A_1, A_2\}, \{F_1, F_2\}\rangle$, where $\mathcal{N}$ signifies the natural numbers, and let the domain description D_{ex} consist of three t-propositions:

$$F_1 \textbf{ holds-at } 4$$

$$F_2 \textbf{ holds-at } 5$$

$$\neg F_2 \textbf{ holds-at } 10$$

This example is neither consistent nor h-consistent. To establish consistency in this case we need to consider explanations which include both narrative and causal information.

Definition 4.7. [hc-explanation] Let $D = \langle\gamma, \eta, \tau\rangle$ be a domain description. An *hc-explanation* for D is a pair $\langle\gamma_\epsilon, \eta_\epsilon\rangle$, where γ_ϵ is a (possibly empty) set of c-propositions and η_ϵ is a (possibly empty) occurrence-sparse set of h-propositions, such that$\langle\gamma \cup \gamma_\epsilon, \eta \cup \eta_\epsilon, \tau\rangle$ is consistent.

For example, according to Definition 4.7 the following are all hc-explanations for D_{ex}:

1. $\langle\{A_1 \textbf{ terminates } F_2 \textbf{ when } \{F_1\}\}, \{A_1 \textbf{ happens-at } 8\}\rangle$
2. $\langle\{A_1 \textbf{ terminates } F_2\}, \{A_1 \textbf{ happens-at } 8\}\rangle$

3. $\langle\{A_1$ **terminates** F_2 **when** $\{\neg F_1\}$, A_2 **terminates** $F_1\}$,
$\{A_2$ **happens-at** 7, A_1 **happens-at** $8\}\rangle$

What qualifies as a 'reasonable' preference criterion among hc-explanations? In practice such a criterion will probably be domain-specific. Nevertheless, it is interesting to speculate at an abstract level, if only to discover the complexity of some of the issues involved. A simple or liberal view (in the sense that it would allow a wide class of optimal hc-explanations) would be to prefer hc-explanation ϵ over hc-explanation ϵ' only if ϵ's set of c-propositions was strictly contained in ϵ''s. However, this policy would not yield any preferences between the three hc-explanations in Example 4.2 above. Regarding D_{ex} as representing three observations at different time points, it seems reasonable that we should prefer explanations which account for the observed change in F_2 but allow F_1 to persist. This would, for example, cut out explanation (3) above. The use of $\mathcal{E}$ to describe the process of learning new c-propositions from such sets of observations is the subject of further investigation.

5. PROJECTION DOMAIN DESCRIPTIONS

In the following two sections, and again in Section 8, we focus attention on a particular sub-class of languages and domain descriptions, which we will call the class of *projection languages* and the class of *projection domain descriptions* respectively[7]. We have three reasons for doing so. In this section, we will show that it is possible to state syntactically verifiable conditions under which projection domain descriptions are consistent. In Section 6 we will show how we can use projection domain descriptions to formulate a notion of explanation complementary to that of Section 4. And in Section 8 we will show how for a particular class of domains we can build on this idea to develop meta-level Prolog implementations which facilitate a 'complete' form of automated reasoning both backwards and forwards in time.

The defining characteristic of a projection language is that the set of time points includes a null or least element, which is given a special status as regards formulation of projection domain descriptions.

Definition 5.1. [Projection Language] A *projection language* is a domain language $\langle\Pi, \preceq, \Delta, \Phi\rangle$, where Π includes an element T_0 (a null or least element) such that for all $T \in \Pi$, $T_0 \preceq T$.

In this section and the next, we assume that $\mathcal{E} = \langle\Pi, \preceq, \Delta, \Phi\rangle$ is a projection language, and that T_0 is the null element of Π. It will be convenient to identify a particular type of t-proposition which we will call an *i-proposition* ("i" for "initial").

Definition 5.2. [i-proposition] An *i-proposition* in $\mathcal{E}$ is a t-proposition of the form

$$L \textbf{ holds-at } T_0$$

where L is a fluent literal of $\mathcal{E}$. We shall sometimes write this expression as

$$\textbf{initially } L$$

[7]Note that the class of Situation-Calculus-style languages of the form $\langle\Pi_\Delta, \leq_\Delta, \Delta, \Phi\rangle$ defined in Proposition 3.1 of Section 3 are all examples of projection languages.

Definition 5.3. [Projection domain description] A *projection domain description* in $\mathcal{E}$ is a triple $\langle \gamma, \eta, \tau_i \rangle$, where γ is a set of c-propositions, η is a set of h-propositions, and τ_i is a set of i-propositions in $\mathcal{E}$.

Thus projection domain descriptions are domain descriptions that only allow t-propositions about the initial time point T_0.

Example 5.1. The following projection domain description uses the projection domain language $\mathcal{E}_{ys} = \langle \Re^+, \leq, \{Shoot\}, \{Alive, Loaded\} \rangle$ of Example 2.2, where $\Re^+$ signifies the non-negative real numbers (so that $T_0 = 0$).

$$Shoot \ \textbf{terminates} \ Alive \ \textbf{when} \ \{Loaded\}$$

$$Shoot \ \textbf{happens-at} \ 2$$

$$\textbf{initially} \ Alive$$

$$\textbf{initially} \ Loaded$$

At first sight it appears that the restriction of the set τ_i in the definition above to contain only i-propositions is a major limitation. However, in Section 6 we will describe a mode of reasoning involving both projection domain descriptions and extra sets of t-propositions which have been identified as 'observations' requiring explanation.

As stated above, one advantage of projection domains is that it is possible to characterise a whole class of such domains whose consistency can be easily verified. To state the appropriate proposition, we first need some extra definitions.

Definition 5.4. [Initial consistency] Let $D = \langle \gamma, \eta, \tau_i \rangle$ be a projection domain description. D is *initially-consistent* iff there is no fluent constant F such that both the i-proposition "**initially** F" and the i-proposition "**initially** $\neg F$" are in τ_i.

The next definition is related to the notion of "e-consistency" defined by Denecker and De Schreye [11] in the context of the Language $\mathcal{A}$.

Definition 5.5. [Conflicting actions] Let D be a domain description. The action constants A_1 and A_2 *conflict in* D iff D contains two c-propositions of the form "A_1 **initiates** F **when** C_1" and "A_2 **terminates** F **when** C_2" and there is no fluent symbol F' in $\mathcal{E}$ such that both $F' \in C_1 \cup C_2$ and $\neg F' \in C_1 \cup C_2$. When $A_1 = A_2 = A$ we say that the action constant A *self-conflicts* in D.

Definition 5.6. [Fluent independence] Let D be a projection domain description. D is *fluent-independent* iff (i) there is no time point t and pair of h-propositions in D of the form "A_1 **happens-at** t" and "A_2 **happens-at** t" such that A_1 and A_2 conflict in D, and (ii) there is no h-proposition in D of the form "A **happens-at** t" such that A self-conflicts in D.

Definition 5.7. [Non-convergence] Let D be a domain description written in a language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$. D and $\mathcal{E}$ are *non-converging* iff for every three (not necessarily distinct) time points T_1, T_2 and T_3 in Π such that $T_1 \preceq T_3$ and $T_2 \preceq T_3$, then either $T_1 \preceq T_2$ or $T_2 \preceq T_1$.

Proposition 5.1. Let D be a projection domain description which is occurrence-sparse, non-converging, initially-consistent, and fluent-independent. Then D is consistent.

PROOF. See Appendix B.2.

6. OBSERVATIONS AND EXPLANATIONS

In Section 4 we linked the notion of an explanation to the idea of transforming an inconsistent domain description into a consistent one. Another aspect of explanation that we might want to capture in the context of temporal domains is to explain what holds at a later time in terms of what holds at an earlier time. For example, we might wish to consider the statement "Fred is not alive" as explained by the statement "the gun was loaded when he was shot". This is an explanation of what holds at a later time in terms of what holds at an earlier time.

In this section we consider only a special case of this type of explanation, which arises specifically in the case of projection domain descriptions. We will regard information about what holds at the least time point T_0 of a projection domain description as a potential explanation for observations about what holds at all later times. In doing so, we are implicitly according i-propositions a special status among t-propositions – rather than recording 'observations', i-propositions are here regarded as statements about what was true 'in the beginning' or 'originally' (implicitly, before anyone was around to start recording observations or performing actions). Other t-propositions are re-introduced not simply as additional statements of the domain description, but as observations that need to be explained (in terms of what was originally true and/or in terms of events that have occurred).

Definition 6.1. [o-proposition] An *o-proposition* in $\mathcal{E}$ is a t-proposition of the form

$$L \textbf{ holds-at } T$$

where $T \neq T_0$.

Definition 6.2. [Observation set] An *observation set* is a non-empty set of o-propositions.

Definition 6.3. [i-explanation] Let $D = \langle \gamma, \eta, \tau_i \rangle$ be a projection domain description and let τ_{ob} be an observation set. An *i-explanation* for τ_{ob} in D is a set $\tau_{i\epsilon}$ of i-propositions such that $\langle \gamma, \eta, \tau_i \cup \tau_{i\epsilon} \rangle$ is consistent and such that for each $p \in \tau_{ob}$,

$$\langle \gamma, \eta, \tau_i \cup \tau_{i\epsilon} \rangle \models p$$

For simplicity, and in contrast to the discussion of Section 4, we do not include here any definition of preference between i-explanations, or any definition of optimality of an i-explanation. Our motivation for defining 'i-entailment' below is simply to maintain the distinction between observations and other aspects of the domain theory while allowing conclusions arising from observations to be properly characterised. As with h-entailment, we are concerned with capturing a 'safe' form of inference – in terms of abduction, we want to i-entailment to correspond to 'entailment in all abductive extensions'.

It is important to point out, however, that for specific domains we might well wish to introduce a definition of preference among i-explanations, and build this into the definition of i-entailment in exactly the same way as h-preference is incorporated in the definition of h-entailment. For example, given the observation that "the car didn't start after the ignition was turned", we might prefer the explanations that "the battery was dead" or that "the car had no petrol" to the explanation that "the car had no engine".

Definition 6.4. [i-model] Let $D = \langle \gamma, \eta, \tau_i \rangle$ be a projection domain description and let τ_{ob} be an observation set. H is an *i-model of D with τ_{ob}* iff there exists an i-explanation $\tau_{i\epsilon}$ for τ_{ob} in D such that H is a model of $\langle \gamma, \eta, \tau_i \cup \tau_{i\epsilon} \rangle$.

Definition 6.5. [i-consistency] Let D be a projection domain description and let τ_{ob} be an observation set. D is *i-consistent with τ_{ob}* iff there is at least one i-model of D with τ_{ob}.

Definition 6.6. [i-entailment] Let D be a projection domain description and let τ_{ob} be an observation set. D with τ_{ob} *i-entails* the t-proposition "F **holds-at** T", written "$D, \tau_{ob} \models_i F$ **holds-at** T", iff for every i-model H of D with τ_{ob}, $H(F,T) = true$. D with τ_{ob} i-entails the t-proposition "$\neg F$ **holds-at** T" iff for every i-model H of D with τ_{ob}, $H(F,T) = false$.

The following proposition shows that the particular definition of i-entailment above amounts to a re-characterisation of entailment as defined in Section 2, keeping the distinction between observation sets and projection domain descriptions. Note, however, that the proposition would not necessarily hold if we were to incorporate a (domain-specific) notion of i-preference in the definitions above.

Proposition 6.1. Let $D = \langle \gamma, \eta, \tau_i \rangle$ be an occurrence-sparse projection domain description and let τ_{ob} be an observation set. Then H is a model of $\langle \gamma, \eta, \tau_i \cup \tau_{ob} \rangle$ if and only if H is an i-model of D with τ_{ob}.

Proof. See Appendix B.3

Example 6.1. Let $\mathcal{E}_{ysp} = \langle \Re^+, \leq, \{Shoot\}, \{Alive, Loaded\} \rangle$, where $\Re^+$ signifies the non-negative real numbers, and let the projection description D_{ysp} consist of a single c-proposition and a single h-proposition:

$$Shoot\ \textbf{terminates}\ Alive\ \textbf{when}\ \{Loaded\}$$

$$Shoot\ \textbf{happens-at}\ 2$$

Let τ_{ysp} be the observation set containing the following two o-propositions:

$$Alive\ \textbf{holds-at}\ 1$$

$$\neg Alive\ \textbf{holds-at}\ 3$$

It is easy to see that there is a unique i-explanation for τ_{ysp} in D_{ysp}:

$$\{\ \textbf{initially}\ \mathit{Loaded},\ \ \textbf{initially}\ \mathit{Alive}\ \}$$

Hence for all $n \geq 0$

$$D_{ysp}, \tau_{ysp} \models_i \mathit{Loaded}\ \textbf{holds-at}\ n$$

The notions of i-explanation and h-explanation described in this section and in Section 4 are complementary and can be combined in an obvious way.

Definition 6.7. [ih-explanation] Let $D = \langle \gamma, \eta, \tau_i \rangle$ be a projection domain description and let τ_{ob} be an observation set. An *ih-explanation* for τ_{ob} in D is a pair $\langle \eta_\epsilon, \tau_{i\epsilon} \rangle$, where η_ϵ is an occurrence-sparse set of h-propositions and $\tau_{i\epsilon}$ is a set of i-propositions, such that $\langle \gamma, \eta \cup \eta_\epsilon, \tau_i \cup \tau_{i\epsilon} \rangle$ is consistent and such that for each $p \in \tau_{ob}$,

$$\langle \gamma, \eta \cup \eta_\epsilon, \tau_i \cup \tau_{i\epsilon} \rangle \models p$$

However, it is less obvious when one ih-explanation should be considered preferable to another. To conclude this section, we compare two preference relations for ih-explanations, both of which might be reasonable for particular classes of domains. We illustrate their different effects with an example. The following discussion lends further weight to the viewpoint that, although the object level semantics of Section 2 is domain-independent, decisions as to which explanations for inconsistent domains or for sets of observations should be preferred will, in general, be based partly on consideration of the specific domain.

The first preference relation we consider is as follows. It compares only the h-propositions in two given ih-explanations.

Definition 6.8. [Preferable ih-explanation]
Let $\epsilon = \langle \eta_\epsilon, \tau_{i\epsilon} \rangle$ and $\epsilon' = \langle \eta'_\epsilon, \tau'_{i\epsilon} \rangle$ be ih-explanations for τ_{ob} in D. ϵ is *ih-preferable* to ϵ' iff $\eta_\epsilon \subset \eta'_\epsilon$.

Definition 6.9. [Optimal ih-explanation] ϵ is an *optimal ih-explanation* for τ_{ob} in D iff ϵ is an ih-explanation for τ_{ob} in D and there is no other ih-explanation ϵ' for τ_{ob} in D such that ϵ' is ih-preferable to ϵ.

Definition 6.10. [ih-model] Let $D = \langle \gamma, \eta, \tau_i \rangle$ be a projection domain description and let τ_{ob} be an observation set. H is an *ih-model of D with τ_{ob}* iff there exists an optimal ih-explanation $\epsilon = \langle \eta_\epsilon, \tau_{i\epsilon} \rangle$ for τ_{ob} in D such that H is a model of $\langle \gamma, \eta \cup \eta_\epsilon, \tau_i \cup \tau_{i\epsilon} \rangle$

Definition 6.11. [ih-entailment] Let D be a projection domain description and let τ_{ob} be an observation set. D with τ_{ob} *ih-entails* the t-proposition "F **holds-at** T", written "$D, \tau_{ob} \models_{ih} F$ **holds-at** T", iff for every ih-model H of D with τ_{ob}, $H(F,T) = \mathit{true}$. D with τ_{ob} ih-entails the t-proposition "$\neg F$ **holds-at** T" iff for every ih-model H of D with τ_{ob}, $H(F,T) = \mathit{false}$.

Example 6.2. This example involves a video-recorder with an automatic timer. Suppose Fred returns home one evening and wishes to check if his video-recorder has automatically started recording his favourite TV show. Although he cannot look inside the recording machine to check if it is recording directly, he knows that if the machine is working, when the timer triggers it will both turn the 'record' light on and begin recording. Fred also knows that if the machine is not working it can be repaired. When Fred left home, the record light was not on. When he returns, the light is on, and Fred concludes that the machine is recording.

We can represent Fred's domain knowledge with a projection domain description D_{rec} consisting of three c-propositions:

$$TimerTrigger \textbf{ initiates } LightOn \textbf{ when } \{Working\}$$

$$TimerTrigger \textbf{ initiates } Recording \textbf{ when } \{Working\}$$

$$Repair \textbf{ initiates } Working$$

Fred's observations, that the recording light was of when he left home but is on when he returns, are represented by the observation set τ_{rec}:

$$\{\neg LightOn \textbf{ holds-at } T_L,\ LightOn \textbf{ holds-at } T_R\}$$

where $T_L, T_R \in \Pi$ represent the times at which Fred left and returned home respectively. It is easy to see that all optimal ih-explanations for τ_{rec} in D_{rec} are of the form

$$\langle \{TimerTrigger \textbf{ happens-at } T\},$$
$$\{\textbf{initially } Working,\ \textbf{initially } \neg LightOn\}\rangle$$

for some $T \in \Pi$ such that $T_L \preceq T \prec T_R$, so that

$$D_{rec}, \tau_{rec} \models_{ih} Recording \textbf{ holds-at } T_R$$

and hence the notion of ih-entailment correctly models Fred's reasoning in this respect.

In the example above, Fred can also use ih-entailment to reason that his video-recorder was functioning properly at time T_L (and indeed at any time point whatsoever), i.e.

$$D_{rec}, \tau_{rec} \models_{ih} Working \textbf{ holds-at } T_L$$

In doing so, Fred rejects ih-explanations in which an extra *Repair* action occurs at some time before the *TimerTrigger* action. It is a matter of debate as to whether, for this domain, it is reasonable to do this. But it is not hard to formulate an alternative preference criterion among ih-explanations which does not cause such alternatives to be rejected:

Definition 6.12. [$\overline{\text{Preferable}}$ ih-explanation]
Let $\epsilon = \langle \eta_\epsilon, \tau_{i\epsilon}\rangle$ and $\epsilon' = \langle \eta'_\epsilon, \tau'_{i\epsilon}\rangle$ be ih-explanations for τ_{ob} in D. ϵ is $\overline{ih}$*-preferable* to ϵ' iff $\eta_\epsilon \subset \eta'_\epsilon$ and $\tau_{i\epsilon} = \tau'_{i\epsilon}$.

In other words, explanations are only "$\overline{\text{ih}}$-comparable" if they contain the same i-propositions. Assuming definitions exactly analogous to Definitions 6.9, 6.10 and 6.11 for an optimal $\overline{\text{ih}}$-explanation, an $\overline{\text{ih}}$-model, and $\overline{\text{ih}}$-entailment respectively, it is easy to see that all optimal ih-explanations are also optimal $\overline{\text{ih}}$-explanations, but that (for the example above) optimal $\overline{\text{ih}}$-explanations can also be of the form

$$\langle\{\mathit{Repair}\ \mathbf{happens\text{-}at}\ T',\ \mathit{TimerTrigger}\ \mathbf{happens\text{-}at}\ T\},$$
$$\{\mathbf{initially}\ \neg \mathit{Working},\ \mathbf{initially}\ \neg \mathit{LightOn}\}\rangle$$

for some $T', T \in \Pi$ such that $T_L \preceq T \prec T_R$ and $T' \prec T$. Hence, although (as for ih-entailment)

$$D_{rec}, \tau_{rec} \models_{\overline{ih}} \mathit{Recording}\ \mathbf{holds\text{-}at}\ T_R$$

Fred cannot use $\overline{\text{ih}}$-entailment to conclude that his video-recorder was working when he left the house.

It is not hard to think up examples either where ih-entailment or where $\overline{\text{ih}}$-entailment seems more plausible. The choice seems to depend on whether (for the particular domain) one is more concerned with assuming as few action occurrences as possible (ih-entailment), or whether one is more concerned with giving equal consideration to all possible initial conditions, even in the face of observations at later times ($\overline{\text{ih}}$-entailment).

7. LOGIC PROGRAMS FOR $\mathcal{E}$ DOMAINS

In the following two sections we discuss the implementation of Language $\mathcal{E}$ domains. In this section we study how we can construct Event Calculus style logic programs from domain descriptions in general. In Section 8 we show how, for a class of projection domain descriptions, (simplified versions of) these programs can be enhanced using standard Prolog 'second-order' programming techniques.

In the original Event Calculus there was an implicit assumption that all predicate definitions were complete. In other words, it was assumed that for each predicate its negation (negation as failure) was true whenever the positive instance did not hold. Although the semantics of $\mathcal{E}$ incorporates an analogous assumption for h-propositions and c-propositions, this assumption does not extend to t-propositions (equivalent to *Holds* or *HoldsAt* literals in Event Calculus programs) – it is possible for a domain description D to be 'incomplete' in the sense that, for some fluent constant F and time T, neither "F **holds-at** T" nor "$\neg F$ **holds-at** T" is entailed by D.

However, we can partly avoid the implementation difficulties that this creates by representing negative fluent literals *inside* the *HoldsAt* predicate. In the program translations defined below, the t-proposition "$\neg F$ **holds-at** T" is represented by the positive literal $\mathit{HoldsAt}(\mathit{Neg}(F), T)$, whereas the negative literal *not* $\mathit{HoldsAt}(F, T)$ is simply interpreted as 'the t-proposition "F **holds-at** T" is not provable'. In this and other respects, the translation method here is similar to that in [30]. Analogous techniques are used in [16], [14] and [4], although not with Event Calculus style programs.

Given that our aim is to develop programs able to deal correctly with the form of incompleteness described above, it is useful to first consider incomplete or partial interpretations for a domain description and examine what can be computed from these.

Definition 7.1. [Partial interpretation] A *partial interpretation* of $\mathcal{E}$ is a partial mapping

$$I : \Phi \times \Pi \mapsto \{true, false\}$$

In the discussion which follows, we assume that Definitions 2.8 and 2.9 of 'point satisfaction' and an 'initiation' or 'termination point' are extended to cover partial interpretations as well as interpretations. In addition, we need counterparts to these notions which deal with cases where $I(F,T)$ is undefined.

Definition 7.2. [Possible point satisfaction] Given a set of fluent literals C of $\mathcal{E}$ and a time point $T \in \Pi$, a partial interpretation I *possibly satisfies* C *at* T iff for each fluent constant $F \in C$, $I(F,T) \neq false$, and for each fluent constant F' such that $\neg F' \in C$, $I(F',T) \neq true$.

Definition 7.3. [Possible initiation/termination point] Let I be a partial interpretation of $\mathcal{E}$, let D be a domain description, let $F \in \Phi$ and let $T \in \Pi$. T is a *possible initiation-point* (respectively *possible termination-point*) *for* F *in* I *relative to* D iff there is an $A \in \Delta$ such that (i) there is both an h-proposition in η of the form "A **happens-at** T" and a c-proposition in γ of the form "A **initiates** F **when** C" (respectively "A **terminates** F **when** C") and (ii) I possibly satisfies C at T.

Let us denote the set of all partial interpretations by $\mathcal{I}$. Motivated by Definition 2.10 of a model for a domain D, we can define an associated (partial) operator on $\mathcal{I}$ as follows.

Definition 7.4. [Operator $\mathcal{F}$] Given a domain description $D = \langle\gamma, \eta, \tau\rangle$ the partial operator $\mathcal{F} : \mathcal{I} \mapsto \mathcal{I}$ is defined as follows: For any partial interpretation $I \in \mathcal{I}$, and any $F \in \Phi$, $T \in \Pi$,

1. (a) For any $T_1 \in \Pi$ such that $T_1 \prec T$, if there is no possible initiation-point or possible termination-point T_2 for F in I relative to D such that $T_1 \preceq T_2 \prec T$, then $(\mathcal{F})(I)(F,T) = I(F,T_1)$.

   (b) For any $T_2 \in \Pi$ such that $T \prec T_2$, if there is no possible initiation-point or possible termination-point T_1 for F in I relative to D such that $T \preceq T_1 \prec T_2$, then $(\mathcal{F})(I)(F,T) = I(F,T_2)$.

2. If T_1 is an initiation-point for F in I relative to D, $T_1 \prec T$ and there is no possible termination-point T_2 for F in I relative to D such that $T_1 \prec T_2 \prec T$, then $(\mathcal{F})(I)(F,T) = true$.

3. If T_1 is a termination-point for F in I relative to D, $T_1 \prec T$ and there is no possible initiation-point T_2 for F in I relative to D such that $T_1 \prec T_2 \prec T$, then $(\mathcal{F})(I)(F,T) = false$.

4. If there is a t-proposition in τ of the form "F **holds-at** T", then

$$(\mathcal{F})(I)(F,T) = true,$$

and if there is a t-proposition of the form "$\neg F$ **holds-at** T",

$$(\mathcal{F})(I)(F,T) = false.$$

5. Otherwise $(\mathcal{F})(I)(F,T)$ is undefined.

Note that this operator is not always defined, as it is possible for these rules to require the assignment of both true and false to $(\mathcal{F})(I)(F,T)$ for some F and T.

It is easy to see that conditions (1) to (4) in the definition above correspond closely to conditions (1) to (4) of Definition 2.10 of a model. The following definition and three propositions show the relationship between Language $\mathcal{E}$ models and the operator $\mathcal{F}$.

Definition 7.5. [The ordering $\subseteq$] For any two partial interpretations $I_1, I_2 \in \mathcal{I}$, I_1 *is contained in* I_2, written $I_1 \subseteq I_2$, iff (i) for any F and T if $I_1(F,T) = true$ then $I_2(F,T) = true$, and (ii) if $I_1(F,T) = false$ then $I_2(F,T) = false$.

Proposition 7.1. Let D be a domain description. If D is consistent then any model H of D is a greatest (with respect to the ordering $\subseteq$) fixed point of $\mathcal{F}$.

Proof. The proof follows directly from the construction of $\mathcal{F}$ and the observation that since H is a total mapping the definition for possible initiation-point (respectively possible termination-point) coincides with that of initiation-point (respectively termination-point).

When a domain D is consistent we can apply the operator $\mathcal{F}$ iteratively to compute a partial interpretation that would be a subset of any model of D.

Proposition 7.2. Let D be a consistent domain description, let H be a model of D and let I be a partial interpretation such that $I \subseteq H$. Then $\mathcal{F}(I) \subseteq H$.

Proof. The proof follows by comparing the different cases under which $\mathcal{F}$ applies with conditions (1)–(4) in Definition 2.10 of a model, and noticing that the following two properties hold for every $T \in \Pi$ and $F \in \Phi$: (i) if T is an initiation-point (resp. termination-point) for F in I then T is an initiation-point (resp. termination-point) for F in H and (ii) if T is not a possible initiation-point (resp. possible termination-point) for F in I then T is not an initiation-point (resp. termination-point) for F in H.

Proposition 7.3. Let D be a consistent domain description and let the sequence $I_0, \ldots, I_n, \ldots$ of partial interpretations be defined as follows:

- $I_0 = \emptyset$
- $I_{n+1} = I_n \cup \mathcal{F}(I_n)$ *for each countable ordinal* $n > 0$.

Then there exists a least fixed point I^+ of this sequence, and D entails any t-proposition of the form "F **holds-at** *T" (resp. "$\neg F$* **holds-at** *T") such that $I^+(F,T) = true$ (resp. $I^+(F,T) = false$).*

Proof. By Proposition 7.2 when D is consistent this sequence is well defined. Also, by construction the sequence is monotonic (with respect to the ordering $\subseteq$

of Definition 7.5) and hence the least fixed point I^+ exists. Again by Proposition 7.2 $I^+ \subseteq H$ for any model H of D and hence the result follows.

Given Proposition 7.3, we can use Definition 7.4 to 'read off' a logic program that implements $\mathcal{F}$ and computes consequences belonging to I^+. To simplify our definitions we assume that some logic program or external definition of the order relation $\preceq$ is available.

Definition 7.6. [Ordering program] Given the language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$, the program $P(\Pi, \preceq)$ is an *ordering program* for $\mathcal{E}$ iff

- for all $T, T' \in \Pi$, $P(\Pi, \preceq)$ succeeds on the query $T \preceq T'$ if and only if $T \preceq T'$, and finitely fails otherwise.
- for all $T, T' \in \Pi$, $P(\Pi, \preceq)$ succeeds on the query $T \prec T'$ if and only if $T \prec T'$, and finitely fails otherwise.
- None of the following predicate symbols appear in $P(\Pi, \preceq)$: *HoldsAt*, *Given*, *ClippedBetween*, *HappensAt*, *PossiblyInitiates*, *Initiates*, *PossiblyTerminates*, *Terminates*, *AffectedBetween*.

We will also need the following preliminary definitions.

Definition 7.7. [lp-term and lp-complement] Given a fluent literal L of $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$, the *lp-term* of L, written $\lambda(L)$, is defined to be

- $Pos(F)$ if $L = F$ for some $F \in \Phi$
- $Neg(F)$ if $L = \neg F$ for some $F \in \Phi$

and the *lp-complement* of L, written $\overline{\lambda(L)}$, is defined to be

- $Neg(F)$ if $L = F$ for some $F \in \Phi$
- $Pos(F)$ if $L = \neg F$ for some $F \in \Phi$

Definition 7.8. [Finite domain description] The domain description $\langle \gamma, \eta, \tau \rangle$ is *finite* iff γ, η and τ are all finite, and for each c-proposition in γ either of the form "A **initiates** F **when** C" or of the form "A **terminates** F **when** C", C is also finite.

Our translation from domain descriptions to logic programs can now be given. In the definition below, the five clauses defining *HoldsAt* correspond to rules (1a)–(4) in Definition 7.4 of the operator $\mathcal{F}$ (Clauses (LP1a) and (LP1b) are 'special cases' of rules (1a) and (1b)). The use of negation-as-failure and fluent converses in the domain-specific clauses defining the predicates *PossiblyInitiates* and *PossiblyTerminates* reflects Definition 7.3 of a possible initiation and possible termination point.

Definition 7.9. [$LP[D, P(\Pi, \preceq)]$] Given a finite domain description $D = \langle \gamma, \eta, \tau \rangle$ written in the language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$, and an ordering program $P(\Pi, \preceq)$, the logic program $LP[D, P(\Pi, \preceq)]$ is defined as the program $P(\Pi, \preceq)$ augmented with the following general clauses

$$HoldsAt(l, t_3) \leftarrow \quad \text{(LP1a)}$$
$$\quad Given(l, t_1),\ t_1 \prec t_3,\ not\ AffectedBetween(t_1, l, t_3).$$

$$HoldsAt(l, t_1) \leftarrow \quad \text{(LP1b)}$$
$$\quad Given(l, t_3),\ t_1 \prec t_3,\ not\ AffectedBetween(t_1, l, t_3).$$

$$HoldsAt(Pos(f), t_3) \leftarrow \quad \text{(LP2)}$$
$$\quad HappensAt(a, t_1),\ t_1 \prec t_3,\ Initiates(a, f, t_1),$$
$$\quad not\ ClippedBetween(t_1, Pos(f), t_3).$$

$$HoldsAt(Neg(f), t_3) \leftarrow \quad \text{(LP3)}$$
$$\quad HappensAt(a, t_1),\ t_1 \prec t_3,\ Terminates(a, f, t_1),$$
$$\quad not\ ClippedBetween(t_1, Neg(f), t_3).$$

$$HoldsAt(l, t) \leftarrow Given(l, t). \quad \text{(LP4)}$$

$$ClippedBetween(t_1, Pos(f), t_3) \leftarrow \quad \text{(LP5)}$$
$$\quad HappensAt(a, t_2),\ t_1 \preceq t_2,\ t_2 \prec t_3,$$
$$\quad PossiblyTerminates(a, f, t_2).$$

$$ClippedBetween(t_1, Neg(f), t_3) \leftarrow \quad \text{(LP6)}$$
$$\quad HappensAt(a, t_2),\ t_1 \preceq t_2,\ t_2 \prec t_3,$$
$$\quad PossiblyInitiates(a, f, t_2).$$

$$AffectedBetween(t_1, l, t_3) \ \leftarrow\ ClippedBetween(t_1, l, t_3). \quad \text{(LP7)}$$

$$AffectedBetween(t_1, Neg(f), t_3) \ \leftarrow \quad \text{(LP8)}$$
$$\quad ClippedBetween(t_1, Pos(f), t_3).$$

$$AffectedBetween(t_1, Pos(f), t_3) \ \leftarrow \quad \text{(LP9)}$$
$$\quad ClippedBetween(t_1, Neg(f), t_3).$$

and the following domain-specific clauses

- For each t-proposition "L **holds-at** T" in τ, the clause

  $Given(\lambda(L), T).$

- For each h-proposition "A **happens-at** T" in η, the clause

  $HappensAt(A, T).$

- For each c-proposition "A **initiates** F **when** $\{L_1, \ldots, L_n\}$" in γ, the clause

$$\begin{array}{l} \mathit{Initiates}(A,F,t) \leftarrow \\ \qquad \mathit{HoldsAt}(\lambda(L_1),t),\ldots,\mathit{HoldsAt}(\lambda(L_n),t). \end{array}$$

and the clause

$$\begin{array}{l} \mathit{PossiblyInitiates}(A,F,t) \leftarrow \\ \qquad \mathit{not}\ \mathit{HoldsAt}(\overline{\lambda(L_1)},t),\ldots,\mathit{not}\ \mathit{HoldsAt}(\overline{\lambda(L_n)},t). \end{array}$$

- For each c-proposition "A **terminates** F **when** $\{L_1,\ldots,L_n\}$" in γ, the clause

$$\begin{array}{l} \mathit{Terminates}(A,F,t) \leftarrow \\ \qquad \mathit{HoldsAt}(\lambda(L_1),t),\ldots,\mathit{HoldsAt}(\lambda(L_n),t). \end{array}$$

and the clause

$$\begin{array}{l} \mathit{PossiblyTerminates}(A,F,t) \leftarrow \\ \qquad \mathit{not}\ \mathit{HoldsAt}(\overline{\lambda(L_1)},t),\ldots,\mathit{not}\ \mathit{HoldsAt}(\overline{\lambda(L_n)},t). \end{array}$$

Intuitively, given Propositions 7.1, 7.2 and 7.3 it is easy to see that the programs described above behave correctly for consistent domain descriptions, since Clauses (LP1a)–(LP4) either correspond exactly to or are just special cases of conditions (1a)–(4) in Definition 7.4. The following proposition[8]confirms this intuition.

Proposition 7.4. Let $P(\Pi,\preceq)$ be an ordering program for $\mathcal{E}$, and let D be a finite domain description. Then for any fluent literal L of $\mathcal{E}$ and any $T \in \Pi$, if

$$LP[D,P(\Pi,\preceq)] \vdash_{SLDNF} \mathit{HoldsAt}(\lambda(L),T)$$

then

$$D \models L\ \textbf{holds-at}\ T$$

Proof. See Appendix B.4

To a certain extent, the above logic programs overcome the two limitations of formalizations of action in normal logic programming identified by Gelfond and Lifschitz in [16]. That is to say, (i) if the values of some fluents at one or more time points are given, they facilitate automated reasoning about what holds at other time points before, afterwards or in between, and (ii) as shown by Proposition 7.4, the programs behave correctly even when the information entailed by their Language $\mathcal{E}$ specifications is incomplete.

However, although they are sound, the above logic programs do not compute all consequences of every domain under its semantics as given by Definition 2.10. This

[8] In this and subsequent propositions, the symbol $\vdash_{SLDNF}$ signifies the existence of an SLDNF derivation, defined as for example in [28]. However, for clarity of presentation, in proposition proofs we avoid working directly with this type of definition, and instead refer to SLDNF derivations using the usual (Prolog) programming terminology.

potential incompleteness is illustrated by the following example.

Example 7.1. Let $\mathcal{E}_v$ and D_v be the domain language and domain description respectively of Example 2.1, and suppose that $P(\Re, \leq)$ is an ordering program for $\mathcal{E}_v$. Then the logic program $LP[D_v, P(\Re, \leq)]$ consists of the program $P(\Re, \leq)$ together with clauses (LP1a)–(LP9) of Definition 7.9 and the domain-specific clauses

$$\begin{aligned}&Initiates(InjectA, Protected, t) \leftarrow \\ &\quad HoldsAt(Pos(TypeO), t).\end{aligned}$$

$$\begin{aligned}&PossiblyInitiates(InjectA, Protected, t) \leftarrow \\ &\quad not\ HoldsAt(Neg(TypeO), t).\end{aligned}$$

$$\begin{aligned}&Initiates(InjectB, Protected, t) \leftarrow \\ &\quad HoldsAt(Neg(TypeO), t).\end{aligned}$$

$$\begin{aligned}&PossiblyInitiates(InjectB, Protected, t) \leftarrow \\ &\quad not\ HoldsAt(Pos(TypeO), t).\end{aligned}$$

$$HappensAt(InjectA, 2).$$

$$HappensAt(InjectB, 3).$$

$$Given(Neg(Protected), 1).$$

As it stands, the query $HoldsAt(Pos(Protected), 4)$ will fail on this program even though the corresponding t-proposition is entailed by its specification. Notice however that the query can be made to succeed by adding either $Given(Pos(TypeO), n)$ or $Given(Neg(TypeO), n)$ to the program for some time point $n < 2$. The example also illustrates the necessity of using the *PossiblyInitiates* and *PossiblyTerminates* predicates. Had the program definition of *ClippedBetween* been given simply in terms of *Initiates* and *Terminates*, the query $HoldsAt(Neg(Protected), 4)$ would succeed, even though the t-proposition "$\neg Protected$ **holds-at** 4" is not entailed by D_v. Finally, notice that the success of the goal $HoldsAt(Neg(Protected), 0)$ (trivially) demonstrates the utility of this type of program for reasoning backwards in time.

8. META-LEVEL PROGRAMS FOR COMPUTING I-ENTAILMENT

In this section we show how, for a class of domains, we can exploit the meta-level characterisation of i-entailment given in Section 6 to build meta-level programs which facilitate a more 'complete' form of reasoning (both backwards and forwards in time) than the object-level programs of the previous section. The important characteristic of the programs given below is that they maintain the distinction between observations, which are dealt with at the meta-level, and other parts of the domain theory.

The object-level programs upon which our meta-level implementation is built are simplified versions of the programs described in Section 7. The simplification is possible because of the following property of projection domain descriptions. If D is a projection domain description which contains either the i-proposition "**initially** F" or the i-proposition "**initially** $\neg F$" for every fluent constant F, then D is 'complete' in the sense that it has at most one model (this follows from Proposition 2.1). If $\mathcal{E}$ is non-converging, and D is fluent-independent, D has exactly one model (this follows from Proposition 5.1). Hence, in the case where D is also finite (so that there are only a finite number of fluent constants in $\mathcal{E}$), it is not hard to construct a simplified Event Calculus style program for D enabling complete automated reasoning forwards in time from the initial time point T_0. For reasons which will shortly become apparent, we will represent the ('complete' set of) i-propositions of D in list form inside a three-argument version of the *HoldsAt* predicate, rather than with a *Given* predicate as previously. In Definition 8.2 below, $\mathit{HoldsAt}(M, L, T)$ should be read as "$D \models L$ **holds-at** T, where M is (a list representation of) the set of i-propositions in D".

It is easy to see that clauses (EC7)–(EC11) which define *HoldsAt* are a simplification of clauses (LP1a)–(LP9) in Definition 7.9. Here it is not necessary to distinguish between the predicates *Initiates* (resp. *Terminates*) and *PossiblyInitiates* (resp. *PossiblyTerminates*), and clauses (LP1a), (LP1b) and (LP4) can be condensed into the single clause (EC7). This is because, at the (object) level of calls to the *HoldsAt* predicate, complete information is available about what holds at the initial time point T_0, and we are only interested in reasoning forwards in time from this point. Incompleteness and reasoning backwards in time are instead dealt with at the meta-level.

We wish to construct a program able to test whether a particular t-proposition is i-entailed by some projection domain description D' (which, unlike D above, may not have an i-proposition for each fluent) together with some observation set τ_{ob}. All that remains to be done is to define a meta-level program able to use the *HoldsAt* predicate to test the truth of the t-proposition in each extension of D' with a 'maximal' i-explanation for τ_{ob}. (It is sufficient to consider only maximal i-explanations, i.e. those which mention every fluent in the language, because of the monotonicity of $\mathcal{E}$ as regards addition of t-propositions to any domain description – see the remarks at the end of Section 2.) This is achieved in a straightforward way by clauses (EC1)–(EC6) in Definition 8.2 below. In this definition, $\mathit{IHoldsAt}(L, T)$ should be read as "$D', \tau_{ob} \models_i L$ **holds-at** T". $\mathit{IExplanation}(M)$ should be read as "M is a (maximal) i-explanation for τ_{ob} in D'".

In Definition 8.2, a Prolog-like syntax for lists is used. Thus the term [] represents the empty list and the term $[\mathit{Head}|\mathit{Remainder}]$ represents a non-empty list whose first element is *Head*. Suitable definitions of the standard list predicates *Member* and *Append* are assumed. The two meta-level (or 'second-order') predicates *Setof* and *Forall* are also used. To summarise their functions, $\mathit{Forall}(\mathit{Condition}, \mathit{Goal})$ succeeds if for all solutions of *Condition*, *Goal* succeeds. $\mathit{Setof}(X, \mathit{Goal}, \mathit{Instances})$ succeeds if *Instances* is the set of instances of X for which *Goal* succeeds, where sets are represented as (possibly empty) lists without repetitions. For practical details of the use of these predicates in the context of Prolog programming, the reader may consult [42]. (It is also assumed that the predicates *IHoldsAt*, *IExplanation*, *Permutation*, and *ConsistentWithObservations* do not appear in the ordering program $P(\Pi, \preceq)$.)

Definition 8.1. The domain language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$ is *fluent-finite* iff the set Φ is finite.

Definition 8.2. $[EC[D, \tau_{ob}, P(\Pi, \preceq)]]$
Let $D = \langle \gamma, \eta, \tau \rangle$ be a finite projection domain description written in a fluent-finite projection language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$. Let T_0 be the null element of Π, and let $P(\Pi, \preceq)$ be an ordering program for $\mathcal{E}$. Let τ_{ob} be a finite observation set. The logic program $EC[D, \tau_{ob}, P(\Pi, \preceq)]$ is defined as the program $P(\Pi, \preceq)$ augmented with the following general clauses

$$\begin{aligned}&IHoldsAt(l,t) \leftarrow \\ &\quad Forall(IExplanation(m), HoldsAt(m,l,t)).\end{aligned} \tag{EC1}$$

$$\begin{aligned}&IExplanation(m) \leftarrow \\ &\quad Setof(l, (Initially(l)), i), \\ &\quad Setof(f, \\ &\qquad (Fluent(f),\ not\ Initially(Pos(f)),\ not\ Initially(Neg(f))), \\ &\qquad p), \\ &\quad Permutation(p,c),\ Append(c,i,m), \\ &\quad ConsistentWithObservations(m).\end{aligned} \tag{EC2}$$

$$Permutation([],[]). \tag{EC3}$$

$$Permutation([f|r1],[f|r2]) \leftarrow Permutation(r1,r2). \tag{EC4}$$

$$Permutation([f|r1],[Neg(f)|r2]) \leftarrow Permutation(r1,r2). \tag{EC5}$$

$$\begin{aligned}&ConsistentWithObservations(m) \leftarrow \\ &\quad Forall(Observation(l,t), HoldsAt(m,l,t)).\end{aligned} \tag{EC6}$$

$$\begin{aligned}&HoldsAt(m,l,t_3) \leftarrow \\ &\quad Member(l,m),\ not\ ClippedBetween(m,T_0,l,t_3).\end{aligned} \tag{EC7}$$

$$\begin{aligned}&HoldsAt(m,Pos(f),t_3) \leftarrow \\ &\quad HappensAt(a,t_1),\ t_1 \prec t_3,\ Initiates(m,a,f,t_1), \\ &\quad not\ ClippedBetween(m,t_1,Pos(f),t_3).\end{aligned} \tag{EC8}$$

$$\begin{aligned}&HoldsAt(m,Neg(f),t_3) \leftarrow \\ &\quad HappensAt(a,t_1),\ t_1 \prec t_3,\ Terminates(m,a,f,t_1), \\ &\quad not\ ClippedBetween(m,t_1,Neg(f),t_3).\end{aligned} \tag{EC9}$$

$$\begin{aligned}&ClippedBetween(m,t_1,Pos(f),t_3) \leftarrow \\ &\quad HappensAt(a,t_2),\ t_1 \preceq t_2,\ t_2 \prec t_3, \\ &\quad Terminates(m,a,f,t_2).\end{aligned} \tag{EC10}$$

$$ClippedBetween(m, t_1, Neg(f), t_3) \leftarrow \quad \text{(EC11)}$$
$$Happens(a, t_2),\ t_1 \preceq t_2,\ t_2 \prec t_3,$$
$$Initiates(m, a, f, t_2).$$

and the following domain-specific clauses

- For each fluent constant $F \in \phi$, the clause

  $Fluent(F)$.

- For each i-proposition "**initially** L" in τ_i, the clause

  $Initially(\lambda(L))$.

- For each o-proposition "L **holds-at** T" in τ_{ob}, the clause

  $Observation(\lambda(L), T)$.

- For each h-proposition "A **happens-at** T" in η, the clause

  $HappensAt(A, T)$.

- For each c-proposition "A **initiates** F **when** $\{L_1, \ldots, L_n\}$" in γ, the clause

  $Initiates(m, A, F, t) \leftarrow$
  $HoldsAt(m, \lambda(L_1), t), \ldots, HoldsAt(m, \lambda(L_n), t)$.

- For each c-proposition "A **terminates** F **when** $\{L_1, \ldots, L_n\}$" in γ, the clause

  $Terminates(m, A, F, t) \leftarrow$
  $HoldsAt(m, \lambda(L_1), t), \ldots, HoldsAt(m, \lambda(L_n), t)$.

Example 8.1. Let $\mathcal{E}_{ysp}$, D_{ysp} and τ_{ysp} be the domain language, domain description and observation set respectively of Example 6.1. Let $P(\Re^+, \leq)$ be an ordering program for $\mathcal{E}_{ysp}$. Then the logic program $EC[D_{ysp}, \tau_{ysp}, P(\Re^+, \leq)]$ consists of the program $P(\Re^+, \leq)$ together with clauses (EC1)–(EC11) of Definition 8.2 and the domain-specific clauses

$Terminates(m, Shoot, Alive, t) \leftarrow HoldsAt(m, Pos(Loaded), t)$.

$Fluent(Alive)$.

$Fluent(Loaded)$.

$Observation(Pos(Alive), 1).$

$Observation(Neg(Alive), 3).$

$HappensAt(Shoot, 2).$

Although potentially somewhat inefficient, the programs described in Definition 8.2 are of interest because they enable sound derivations[9] of t-propositions which would not be possible with the object-level logic programs given in the previous section. For example, it is easy to verify, either by inspection or using a Prolog interpreter, that

$$EC[D_{ysp}, \tau_{ysp}, P(\Re^{+}, \leq)] \vdash_{SLDNF} IHoldsAt(Pos(Loaded), 0)$$

Indeed, for a wide class of domains they are both sound and 'complete', in the sense of Proposition 8.1 below. Since any finite, consistent Language $\mathcal{A}$ domain as defined in [16] may be translated directly into a Language $\mathcal{E}$ projection domain description together with an observation set (see Section 3), finite $\mathcal{A}$ domains may also be given a meta-level implementation of this type[10].

Proposition 8.1. Let $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$ be a fluent-finite, non-converging projection language, let $P(\Pi, \preceq)$ be an ordering program for $\mathcal{E}$, and let $D = \langle \gamma, \eta, \tau_i \rangle$ be a finite, initially-consistent, fluent-independent projection domain description in $\mathcal{E}$. Let τ_{ob} be a finite observation set. Then for any fluent literal L of $\mathcal{E}$ and any $T \in \Pi$,

$$EC[D, \tau_{ob}, P(\Pi, \preceq)] \vdash_{SLDNF} IHoldsAt(\lambda(L), T)$$

if and only if

$$D, \tau_{ob} \models_i L \textbf{ holds-at } T$$

PROOF. See Appendix B.5

As stated in Proposition 5.1, it is possible to verify the consistency of the class of projection domain descriptions described in Proposition 8.1 by a syntactic check. Note that we can now build on this proposition to check for i-consistency with a given observation set, simply by verifying the success of the unground call $IExplanation(m)$. Finally, the style of the programs described in this section offers some clue as to how we might in principle implement a preference criterion added to the definition if i-entailment, by appropriately extending the program definition of $IExplanation$.

[9] In order to continue to refer to 'SLDNF derivations', we assume that the meta-level primitives $Setof$ and $Forall$ are appropriately re-interpreted (see [42] for details).

[10] Strictly speaking, the translation method described in Definition 8.2 is not applicable to the Language $\mathcal{A}$ type domain descriptions described in Section 3 Proposition 3.1. This is because the "complete occurrence set" of h-propositions is infinite (see Definition 3.3). But the appropriate modification to Definition 8.2 is trivial, and does not necessitate a significant change to the proof of Proposition 8.1. An example ordering program $P(\Pi_{\Delta}, \leq_{\Delta})$ is given in Appendix C together with the necessary changes to Definition 8.2.

9. CONCLUSIONS AND FURTHER WORK

Following the methodology of the Language $\mathcal{A}$ introduced in [16], we have presented a simple declarative language, $\mathcal{E}$, for describing narratives with actions. $\mathcal{E}$ is based on a narrative ontology inherited from the Event Calculus, thus demonstrating that this methodology is not limited to the particular ontology of $\mathcal{A}$. $\mathcal{E}$'s semantics is based around a simple characterisation of persistence which facilitates a modular approach to extending the expressivity of the language. This characterisation relies heavily on the notion of a flow of time which is independent from any actions which may occur. The benefits of this become particularly apparent when representing domains where periods of time elapse in which a change may or may not have taken place. It is not necessary to 'fill in' time with an artificial 'action' such as a '$Wait$'.

The explicit notion of an action occurrence incorporated in $\mathcal{E}$ allows an important class of 'narrative' explanations (h-explanations) to be characterised in a simple way. These enable us to extend an otherwise inconsistent theory written in $\mathcal{E}$ so as to establish consistency, thus providing a natural method, in many cases, to account for conflicting sets of information about the domain. More generally, our formalisation of various notions of explanation within $\mathcal{E}$ illustrates that commonsense reasoning need not always be modelled as deduction at a single object level. Our results are built upon much previous work concerning the role of abduction in Artificial Intelligence and related areas. Once again we have demonstrated that reasoning from cause to effect can be modelled at the object level, whereas reasoning from effect to cause can be regarded as an essentially meta-level (for example abductive) activity. In the context of reasoning about action, causation is temporally directed. Hence in our work this distinction manifests itself in the fact that reasoning forwards in time is modelled as object-level deduction, whereas reasoning backwards in time is captured at the meta-level. The success of this approach lends extra weight to a developing consensus (see [8] for a general discussion) that observations should somehow be treated separately from other aspects of theories of action.

We have also shown how domains in $\mathcal{E}$ can be implemented in normal logic programming with extended versions of Event Calculus programs that behave correctly even when the knowledge entailed by the domain description is incomplete. These programs have the capability of reasoning backwards as well as forwards in time.

We can envisage at least three areas of future research relating to $\mathcal{E}$. Firstly, in line with the methodology described in our introduction, we could use $\mathcal{E}$ as a 'measuring stick' to show correspondences between various narrative-based formalisms for reasoning about action, perhaps in the manner of Katha in [21]. Example candidates for comparison are the formalisms in [31], [34], [40] and [41].

Secondly, it would be interesting to investigate different styles of implementation as regards $\mathcal{E}$ domains. The approaches followed in this paper are based on a relatively simple use of normal logic programming and standard techniques within this. We could develop a more general implementation for computing entailment or i-entailment using abductive logic programming, building on the work in [10], [11], [19] and [20]. The 'abductive flavour' of our definition of h-entailment suggests that abductive logic programming could be a useful implementation tool here as well.

Thirdly, the expressivity of $\mathcal{E}$ could be increased in various ways. We have already briefly indicated how $\mathcal{E}$ might be extended to partially deal with *ramifications* and *qualifications* (see Appendix A). Since $\mathcal{E}$ already allows for actions to occur *concurrently* within a narrative, it seems likely that the language could also be extended to allow for a theory of cancelling and combined effects of actions similar to that in [4]. It has already been pointed out [10] [12] [33] that a narrative based approach offers alternative ways to model *non-deterministic* effects of actions, and is a natural setting in which to model *continuous change* [39] [43] [32] [41]. Finally, we might extend the syntax and semantics of $\mathcal{E}$ to deal with incomplete information about the order and timing of action occurrences, perhaps building on the ideas in [10] and [12], and perhaps introducing temporal variables into the language in a manner similar to [5] and [6].

The utility and appeal of specialised declarative languages such as $\mathcal{A}$ and $\mathcal{E}$ lies in their simplicity. They are of sufficiently 'high level' to allow various issues to be aired without immediately becoming involved in technical details, and are perhaps best regarded as useful stepping stones towards the ultimate goal of developing comprehensive formal theories of action using general purpose representational mechanisms. Hence their capacity for retaining their simplicity when extended to cover more complex domains is a crucial measure of their utility.

Acknowledgements

We would particularly like to thank Marc Denecker, Yannis Dimopoulos, David Evans, Alessandro Provetti, Alessandra Russo, Fariba Sadri, Murray Shanahan and Eugenia Ternovskaia for helpful discussions on the topic of this paper, and for careful reading of earlier drafts. Thanks as well to the anonymous referees for their thoughtful and detailed comments, which helped considerably in improving the paper.

REFERENCES

1. James Allen, *Towards a General Theory of Action and Time*, Artificial Intelligence 23, Elsevier Science Publishers, pages 123-154, 1984.
2. Jonathan Amsterdam, *Temporal Reasoning and Narrative Conventions*, Proceedings of the 2nd International Conference on Knowledge Representation (KR 91), ed.s J. Allen, R. Fikes and E. Sandewall, Morgan Kaufmann, pages 15–21, 1991.
3. Andrew Baker, *Nonmonotonic Reasoning in the Framework of the Situation Calculus*, Artificial Intelligence 49, Elsevier Science Publishers, page 5, 1991.
4. Chitta Baral and Michael Gelfond, *Representing Concurrent Actions in Extended Logic Programming*, Proceedings IJCAI 93, Morgan Kaufmann, page 866, 1993.
5. Chitta Baral, Michael Gelfond and Alessandro Provetti, *Representing Actions - I: Laws, Observations and Hypotheses (Short Version)*, in Working Notes of the AAAI Spring Symposium: Extending Theories of Action - Formal Theory and Practical Applications, Stanford University, California, USA, 1995.
6. Chitta Baral, Michael Gelfond and Alessandro Provetti, *Representing Actions: Laws, Observations and Hypotheses*, Journal of Logic Programming, Special Issue on Reasoning about Actions (this issue), 1996.
7. Iliano Cervesato, Luca Chittaro and Angelo Montanari, *A Modal Calculus of Partially Ordered Events in a Logic Programming Framework*, in Proceedings ICLP'95, pages 299-313, 1995.

8. James Crawford and David Etherington, *Observations on Observations in Action Theories*, in Working Notes of the AAAI Spring Symposium: Extending Theories of Action - Formal Theory and Practical Applications, Stanford University, California, USA, 1995.

9. Ernest Davis, *Infinite Loops in Finite Time: Some Observations*, Proceedings KR 92 (3rd International Conference on Principles of Knowledge Representation and Reasoning), Cambridge, Massachusetts, ed.s B. Nebel, C. Rich and W. Swartout, Morgan Kaufmann, 1992.

10. Marc Denecker, Lode Missiaen and Maurice Bruynooghe, *Temporal Reasoning with Abductive Event Calculus*, in Proceedings ECAI 92, Vienna, 1992.

11. Marc Denecker and Danny De Schreye, *Reperesenting Incomplete Knowledge in Abductive Logic Programming*, in Proceedings of the International Symposium on Logic Programming, 1993.

12. Marc Denecker, *Knowledge Representation and Reasoning in Incomplete Logic Programming*, PhD thesis, Department of Computer Science, K.U.Leuven, 1993.

13. Marc Denecker, *A Terminological Interpretation of (Abductive) Logic Programming*, in Proceedings of the Third International Conference on Logic Programming and Non-monotonic Reasoning, Lexington, KY, USA, Springer Verlag, 1995.

14. Phan Minh Dung, *Representing Actions in Logic Programming and its Applications in Database Updates*, Proceedings of the Tenth International Conference on Logic Programming, ed David S. Warren, MIT Press, pages 222-238, 1993.

15. Kave Eshghi, *Abductive Planning with Event Calculus*, Proceedings of the 5th International Conference and Symposium on Logic Programming, ed.s Robert Kowalski and Kenneth Bowen, MIT Press, pages 562–579, 1988.

16. Michael Gelfond and Vladimir Lifschitz, *Representing Actions in Extended Logic Programming*, Proceedings of the Joint International Conference and Symposium on Logic Programming, ed. Krzysztof Apt, MIT Press, page 560, 1992.

17. Michael Gelfond and Vladimir Lifschitz, *Representing Action and Change by Logic Programs*, Journal of Logic Programming, volume 17 numbers 2, 3 and 4, North-Holland, pages 301-322, 1993.

18. David Harel, *Dynamic Logic*, In Handbook of Philosophical Logic II: Extensions of Classical Logic, ed.s D. Gabbay and F. Guenther, Reidel, Boston, USA, pages 497-604, 1984.

19. Antonios Kakas and Paolo Mancarella, *Generalized Stable Models: a Semantics for Abduction.* Proceedings of the 9th European Conference on Artificial Intelligence, Stockholm, ed. L. Aiello, pages 385-391, 1990.

20. Antonios Kakas, Robert Kowalski and Francesca Toni, *Abductive logic programming*, Journal of Logic and Computation vol. 2 no. 6, pages 719-770, 1993.

21. G. Neelakantan Kartha, *Soundness and Completeness Theorems for Three Formalizations of Action*, Proceedings IJCAI 93, page 724, 1993.

22. G. Neelakantan Kartha, *Two Counterexamples Related to Baker's Approach to the Frame Problem*, Artificial Intelligence 69, Elsevier Science Publishers, pages 379-392, 1994.

23. Henry Kautz, *The Logic of Persistence*, Proceedings AAAI 86, page 401, 1986.

24. Robert A. Kowalski and Marek J. Sergot, *A Logic-Based Calculus of Events*, New Generation Computing, vol 4, page 267, 1986.

25. Robert A. Kowalski and Fariba Sadri, *The Situation Calculus and Event Calculus Compared*, in Proceedings of the International Logic Programming Symposium (ILPS'94), 1994.

26. Vladimir Lifschitz, *A Language for Describing Actions*, in Working Papers of Common Sense '93: The Second Symposium on Logical Formalizations of Commonsense Reasoning, pages 103-113, Austin, Texas, U.S.A., 1992.

27. Fangzhen Lin and Yoav Shoham, *Provably Correct Theories of Action*, in Proceedings AAAI 91, page 349, MIT Press, 1991.

28. John W. Lloyd, *Foundations of Logic Programming*, Springer Verlag, 1984.

29. John McCarthy and Patrick Hayes, *Some Philosophical Problems from the Standpoint of Artificial Intelligence*, in Machine Intelligence 4, ed.s D. Michie and B. Meltzer, Edinburgh University Press, 1969.

30. Rob Miller, *Situation Calculus Specifications for Event Calculus Logic Programs*, in Proceedings of the Third International Conference on Logic Programming and Non-monotonic Reasoning, Lexington, KY, USA, Springer Verlag, 1995.

31. Rob Miller and Murray Shanahan, *Narratives in the Situation Calculus*, in Journal of Logic and Computation, Special Issue on Actions and Processes, vol 4 no 5, Oxford University Press, 1994.

32. Rob Miller and Murray Shanahan,, *Reasoning about Discontinuities in the Event Calculus*, in L. C. Aiello, J. Doyle & S. C. Shapiro, ed.s, Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96), Morgan Kaufmann, 1996.

33. Javier Pinto, *Temporal Reasoning in the Situation Calculus*, PhD. Thesis, University of Toronto, 1994.

34. Javier Pinto and Raymond Reiter, *Temporal Reasoning in Logic Programming: A Case for the Situation Calculus*, Proceedings ICLP 93, page 203, 1993.

35. Alessandro Provetti, *Hypothetical Reasoning about Actions: From Situation Calculus to Event Calculus*, Computational Intelligence, volume 12, number 2, 1995.

36. Erik Sandewall, *Feature and Fluents*, Oxford University Press, 1994.

37. Marek Sergot, *An Introduction to the Event Calculus*, in Lecture Notes of the GULP Advanced School on Foundations of Logic Programming, (unpublished), Alghero, Sardinia, 1990.

38. Murray Shanahan, *Prediction Is Deduction but Explanation Is Abduction*, Proceedings IJCAI 89, pages 1055-1060, 1989.

39. Murray Shanahan, *Representing Continuous Change in the Event Calculus*, Proceedings ECAI'90, pages 598-603, 1990.

40. Murray Shanahan, *A Circumscriptive Calculus of Events*, Artificial Intelligence, vol. 77, pages 249-284, Elsevier Science Publishers, 1995.

41. Murray Shanahan, *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*, MIT Press (to appear), 1997.

42. Leon Sterling and Ehud Shapiro, *The Art of Prolog*, MIT Press, 1986.

43. Kristof Van Belleghem, Marc Denecker and Danny De Schreye, *Representing Continuous Change in the Abductive Event Calculus*, in Proceedings 1994 International Conference on Logic Programming, ed. P. Van Hentenrijck, pages 225-240, 1994.

44. Kristof Van Belleghem, Marc Denecker and Danny De Schreye, *On the Relation Between Situation Calculus and Event Calculus*, Journal of Logic Programming, Special Issue on Reasoning about Actions (this issue), 1996.

APPENDICES

A. EXTENDING THE EXPRESSIVITY OF $\mathcal{E}$

Two extensions to the syntax and semantics of $\mathcal{E}$ are given in this appendix. This is in order to illustrate that the basic notion of a model, encapsulated in Definitions 2.7 to 2.10, may be modified to accommodate extra types of propositions, without altering the basic principle of persistence captured in conditions (1)-(3) of Definition 2.10. Both extensions are very simple, and although they are obviously related to aspects of the *qualification problem* and *ramification problem* respectively, it is not our intention to suggest that, in this short space, we have developed a comprehensive approach to these subtle and complex issues.

A.1. Describing Conditions under which an Action Cannot Occur

In some circumstances it may be possible to infer knowledge about the conditions at the time of an action occurrence from the fact that the action did occur. For example, given that we know that "at 2 o'clock the caretaker unlocked the door", we might typically infer that (at 2 o'clock) "she had the key". This is because it is impossible to unlock a door without a key, which might be expressed by a proposition such as

$$Unlock\ \textbf{impossible-if}\ \{\neg HasKey\}$$

This motivates a general definition for a new type of proposition:

Definition A.1. [q-proposition] A *q-proposition* in $\mathcal{E}$ is an expression of the form

$$A\ \textbf{impossible-if}\ C$$

where $A \in \Delta$, and C is a set of fluent literals of $\mathcal{E}$.

Such propositions may be accommodated in the semantics of $\mathcal{E}$ by strengthening condition (4) of Definition 2.10 (which expresses simple pointwise constraints on a model), without changing in the basic notion of persistence encapsulated in conditions (1)-(3). Assuming that domain descriptions are now defined as a quadruple $\langle \gamma, \eta, \tau, \kappa \rangle$, where γ, η and τ are as before, and κ is a set of q-propositions[11], the condition now becomes:

(a) For all t-propositions in τ of the form "F **holds-at** T", $H(F,T) = true$, and for all t-propositions of the form "$\neg F$ **holds-at** T", $H(F,T) = false$.

(b) For all pairs of h-propositions and q-propositions in $\eta \times \kappa$ of the form "A **happens-at** T" and "A **impossible-if** C", H does not satisfy C at T.

[11] Note that, if for every c-proposition "A **initiates** F **when** C" there is a q-proposition "A **impossible-if** $C \cup \{F\}$", in any model all initiation points for F are actual points of change for F. An analogous observation holds for termination points. Thus Sergot's notions of *strong initiation* and *strong termination* [37] can be incorporated into $\mathcal{E}$ by addition of q-propositions where appropriate.

A.2. Describing Indirect Effects of Actions

The following extension to $\mathcal{E}$ is included to illustrate how Definition 2.9 of an initiation or termination point might be refined, without necessitating a change in the basic notion of persistence encapsulated in conditions (1)-(3) of Definition 2.10. Suppose that we wish to express simple constraints between fluents. To take a canonical example, suppose that we want to express that a room is stuffy when the window is closed and the ventilator blocked. This might be expressed by a proposition such as

$$\mathit{Stuffy}\ \textbf{whenever}\ \{\mathit{Closed}, \mathit{Blocked}\}$$

Hence we define a new type of proposition as follows:

Definition A.2. [r-proposition] An *r-proposition* in $\mathcal{E}$ is an expression of the form

$$L\ \textbf{whenever}\ C$$

where L is a fluent literal and C is a set of fluent literals of $\mathcal{E}$.

We now need a recursive definition of an initiation point and of a termination point, since, for example, if the ventilator is blocked, the action of closing the window will (indirectly) initiate the property of the room being stuffy. The modified definitions below assume that domain descriptions are defined as a tuple $\langle \gamma, \eta, \tau, \kappa, \rho \rangle$, where γ, η and τ are as before, κ is a set of q-propositions and ρ is a set of r-propositions.

Definition A.3. [Initiation/termination point for domains with r-propositions] Let H be an interpretation of $\mathcal{E}$, let $D = \langle \gamma, \eta, \tau, \kappa, \rho \rangle$ be a domain description, let $F \in \Phi$ and let $T \in \Pi$. T is an *initiation-point* (respectively *termination-point*) *for F in H relative to D* iff one of the following two conditions holds.

1. There is an $A \in \Delta$ such that (i) there is both an h-proposition in η of the form "A **happens-at** T" and a c-proposition in γ of the form "A **initiates** F **when** C" (respectively "A **terminates** F **when** C") and (ii) H satisfies C at T.

2. There is an r-proposition in ρ of the form "F **whenever** C" (respectively "$\neg F$ **whenever** C") and a partition $\{C_1, C_2\}$ of C such that (i) C_1 is non-empty, for each fluent constant $F' \in C_1$, T is an initiation point for F', and for each fluent literal $\neg F' \in C_1$, T is a termination point for F', and (ii) there is some $T_2 \in \Pi$, $T \prec T_2$, such that for all T_1, $T \preceq T_1 \preceq T_2$, H satisfies C_2 at T_1.

Intuitively, condition (2) above states that in order find time points at which the fluent F becomes indirectly initiated via the r-proposition F **whenever** C, we need to look for points at which one or more of the conditions in C become satisfied, and at which the remaining conditions were already satisfied (and continue to be satisfied up to some point T_2 beyond the point in question).

Condition (4) of Definition 2.10 is now:

(a) For all t-propositions in τ of the form "F **holds-at** T", $H(F,T) = \mathit{true}$, and for all t-propositions of the form "$\neg F$ **holds-at** T", $H(F,T) = false$.

(b) For all pairs of h-propositions and q-propositions in $\eta \times \kappa$ of the form "A **happens-at** T" and "A **impossible-if** C", H does not satisfy C at T.

(c) For all r-propositions in ρ of the form "L **whenever** C", if H satisfies C at T then H satisfies $\{L\}$ at T.

B. PROPOSITION PROOFS

B.1. Proof of Proposition 2.1

PROPOSITION STATEMENT: *Let D be an occurrence-sparse domain description written in a language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$, and let $T_1, T_2 \in \Pi$ be such that $T_1 \preceq T_2$. Let H and H' be models of D such that for all $F \in \Phi$, $H(F,T_1) = H'(F,T_1)$. Then for all $F \in \Phi$, $H(F,T_2) = H'(F,T_2)$.*

PROOF: Proof is by induction on the number n of h-propositions in η of the form A **happens-at** T such that $T_1 \preceq T \prec T_2$.

Base Case: If $n = 0$ then by the first condition in the definition of a model, for all $F \in \Phi$, $H(F,T_2) = H(F,T_1) = H'(F,T_1) = H'(F,T_2)$.

Inductive Step: Suppose that $n > 0$ and that the lemma is true for all $m < n$. Let $F' \in \Phi$ be an arbitrary fluent constant. It is sufficient to show that $H(F',T_2) = H'(F',T_2)$. Since D is occurrence-sparse there exists at least one $T \in \Pi$ such that (i) $T_1 \preceq T \prec T_2$, (ii) there is at least one h-proposition in η of the form "A **happens-at** T", and (iii) there is no h-proposition in η of the form "A **happens-at** T'" such that $T \prec T' \prec T_2$. Let T_h be such a time point. By the inductive hypothesis, for all $F \in \Phi$, $H(F,T_h) = H'(F,T_h)$ and by construction there are no initiation or termination points T' (for any fluent) in H or H' such that $T_h \prec T' \prec T_2$. There are three cases to consider (since in any model, F' must be unaffected, initiated or terminated at T_h):

Case one: There is not both an h-proposition in η of the form "A **happens-at** T_h" and a c-proposition in γ either of the form "A **initiates** F' **when** C" or of the form "A **terminates** F' **when** C" such that H (and thus H') satisfies C at T_h. Hence by the first condition in the definition of a model, $H(F',T_2) = H(F',T_h) = H'(F',T_h) = H'(F',T_2)$.

Case two: There is both an h-proposition in η of the form "A **happens-at** T_h" and a c-proposition in γ of the form "A **initiates** F' **when** C" such that H (and thus H') satisfies C at T_h. Hence by the second condition in the definition of a model, $H(F',T_2) = true = H'(F',T_2)$.

Case three: There is both an h-proposition in η of the form "A **happens-at** T_h" and a c-proposition in γ of the form "A **terminates** F' **when** C" such that H (and thus H') satisfies C at T_h. Hence by the third condition in the definition of a model, $H(F',T_2) = false = H'(F',T_2)$.

B.2. Proof of Proposition 5.1

PROPOSITION STATEMENT: *Let D be an occurrence-sparse, non-converging, initially-consistent, fluent-independent projection domain description. Then D is consistent.*

PROOF: Let $D = \langle \gamma, \eta, \tau_i \rangle$ be written in the projection language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$, and let T_0 be the null element of Π.

Let $M : \Phi \mapsto \{true, false\}$ be defined as follows. For each $F \in \Phi$,

- $M(F) = true$ if there is an i-proposition in τ_i of the form "**initially** F",
- $M(F) = false$ otherwise.

M will be used to construct a model H of D such that for all $F \in \Phi$, $H(F, T_0) = M(F)$. Notice that since D is initially consistent, then for each $F \in \Phi$ such that there is an i-proposition in τ_i of the form "**initially** $\neg F$", $M(F) = false$.

Since D is occurrence-sparse and non-converging, each time point $T \in \Pi$ has a unique, maximal, finite (possibly empty) sequence $T_1, \ldots, T_n$ associated with it such that $T_1 \prec \ldots \prec T_n \prec T$ and such that for each T_i there is an h-proposition in η of the form "A **happens-at** T_i". Moreover, the unique such sequence associated with T_n is $T_1, \ldots, T_{n-1}$. Therefore, it is possible to define an interpretation H of D inductively as follows. For each $T \in \Pi$ and $F \in \Phi$,

1. $H(F, T) = M(F)$ if $n = 0$ (i.e. the sequence associated with T is empty),
2. $H(F, T) = H(F, T_n)$ if $n > 0$ and there is no $A \in \Delta$ such that there is both an h-proposition in η of the form "A **happens-at** T_n" and a c-proposition in γ either of the form "A **initiates** F **when** C" or of the form "A **terminates** F **when** C" such that H satisfies C at T_n,
3. $H(F, T) = true$ if $n > 0$ and there is an $A \in \Delta$ such that there is both an h-proposition in η of the form "A **happens-at** T_n" and a c-proposition in γ of the form "A **initiates** F **when** C" such that H satisfies C at T_n,
4. $H(F, T) = false$ if $n > 0$ and there is an $A \in \Delta$ such that there is both an h-proposition in η of the form "A **happens-at** T_n" and a c-proposition in γ of the form "A **terminates** F **when** C" such that H satisfies C at T_n.

The fact that D is fluent-independent guarantees that no fluent/time-point pair (F, T) satisfies both of conditions 3 and 4 above. Hence H is well defined, and is clearly a model of D.

B.3. Proof of Proposition 6.1

PROPOSITION STATEMENT: *Let $D = \langle \gamma, \eta, \tau_i \rangle$ be an occurrence-sparse projection domain description and let τ_{ob} be an observation set. Then H is a model of $\langle \gamma, \eta, \tau_i \cup \tau_{ob} \rangle$ if and only if H is an i-model of D with τ_{ob}.*

PROOF: Let D be written in the projection language $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$, and let T_0 be the null element of Π.

"If" half: Suppose H is an i-model of D with τ_{ob}. Then by the definitions of an model and of an i-model there exists some set $\tau_{i\epsilon}$ of i-propositions such that H is a model of $\langle \gamma, \eta, \tau_i \cup \tau_{i\epsilon} \cup \tau_{ob} \rangle$. Hence, by the monotonicity of $\mathcal{E}$ as regards addition of t-propositions to domain descriptions (see the remarks at the end of Section 2), H is a model of $\langle \gamma, \eta, \tau_i \cup \tau_{ob} \rangle$.

"Only if" half: Suppose H is a model of $\langle \gamma, \eta, \tau_i \cup \tau_{ob} \rangle$. Let the set τ_H of i-propositions be defined as follows. For each $F \in \Phi$,

- **initially** $F \in \tau_H$ iff $H(F, T_0) = true$
- **initially** $\neg F \in \tau_H$ iff $H(F, T_0) = false$

Clearly H is a model of $\langle \gamma, \eta, \tau_i \cup \tau_H \rangle$ and, by Proposition 2.1, $\langle \gamma, \eta, \tau_i \cup \tau_H \rangle \models p$ for each $p \in \tau_{ob}$, so that τ_H is an i-explanation for τ_{ob} in D. Hence H is an i-model of D with τ_{ob}.

B.4. Proof of Proposition 7.4

PROPOSITION STATEMENT:
Let $P(\Pi, \preceq)$ be an ordering program for $\mathcal{E}$, and let D be a finite domain description. Then for any fluent literal L of $\mathcal{E}$ and any $T \in \Pi$, if

$$LP[D, P(\Pi, \preceq)] \vdash_{SLDNF} HoldsAt(\lambda(L), T)$$

then

$$D \models L \textbf{ holds-at } T$$

The proof of this proposition which is given below uses induction on the 'length' $length(\alpha)$ of the SLDNF derivation α of $HoldsAt(\lambda(L), T)$, where $length(\alpha)$ is defined in Definition B.1 below in terms of successful calls to $HappensAt$. It is defined so that each SLDNF sub-derivation of a $HoldsAt$ sub-goal within α (which must have occurred within some call to $Initiates$, $Terminates$, $PossiblyInitiates$ or $PossiblyTerminates$) has 'length' less than the top-level derivation α.

Definition B.1. [$length(\alpha)$] Let α be a successful SLDNF derivation of the goal $HoldsAt(\lambda(L), T)$ in $LP[D, P(\Pi, \preceq)]$. $length(\alpha)$ is defined inductively as follows:

$$length(\alpha) = S + \sum_{\beta \in B_\alpha} size(\beta)$$

where

- S is the number of successful calls to $HappensAt$ at the top level of α, i.e. not called within a (negation-as-failure) finitely-failed subsidiary derivation of α.
- B_α is the set of all finitely-failed subsidiary derivations from negative calls (to $ClippedBetween$ or $AffectedBetween$) appearing at the top level of α.
- $size(\beta) = 0$ if there is no successful call to $HappensAt$ in any branch of β.

- $size(\beta) = 1 + \max(\{length(\alpha') \mid \alpha' \in A_\beta\})$ if there is a successful call to *HappensAt* inside β, where A_β is the set of all successful SLDNF derivations of a *HoldsAt* goal, called as a negated sub-goal in one of the branches of β[12].

PROOF OF PROPOSITION. Let α be a successful SLDNF derivation of the goal $HoldsAt(\lambda(L), T)$ in $LP[D, P(\Pi, \preceq)]$. We will use induction on $length(\alpha)$ to show that, given the fixed point I^+ as defined in Proposition 7.3, if $LP[D, P(\Pi, \preceq)] \vdash_{SLDNF} HoldsAt(\lambda(L), T)$ then if $L = F$ for some $F \in \Phi$ then $I^+(F, T) = true$, and if $L = \neg F'$ for some $F' \in \Phi$ then $I^+(F', T) = false$. The proposition will then follow directly from Proposition 7.3.

Base Case ($length(\alpha) = 0$):
Clearly, if $length(\alpha) = 0$ the query $HoldsAt(Pos(F), T)$ (respectively $HoldsAt(Neg(F'), T)$) can succeed only on clauses (LP1a), (LP1b) or (LP4), as success on (LP2) or (LP3) would require $length(\alpha) \geq 1$. We consider each of these possibilities in turn:

(i) Success on (LP1a): Clearly, the success of $Given(\lambda(L), T_1)$ for some $T_1 < T$ means that "L **holds-at** T_1" $\in D$ and so $I^+(F, T_1) = true$ (respectively $I^+(F', T_1) = false$) by rule (4) in the definition of $\mathcal{F}$. Also, since the call β to $AffectedBetween$ fails with $size(\beta) = 0$, the unground sub-goal $HappensAt(a, t_2)$ fails, so that there are no h-propositions in D. Hence there are no possible initiation points or possible termination points between T_1 and T. Therefore, since I^+ is a fixed point of $\mathcal{F}$, rule (1a) of $\mathcal{F}$ applies to give $I^+(F, T) = true$ (respectively $I^+(F', T) = false$).

(ii) Success on (LP1b): The proof is exactly analogous to (i), but using rule (1b) of $\mathcal{F}$ in place of rule (1a).

(iii) Success on (LP4): Trivially, $I^+(F, T) = true$ (respectively $I^+(F', T) = false$) by rule (4) in the definition of $\mathcal{F}$.

Inductive Step ($length(\alpha) = n$):
Suppose that the statement which we wish to prove is true for all SLDNF derivations α' of all *HoldsAt* goals such that $length(\alpha') < n$. The query $HoldsAt(Pos(F), T)$ (respectively$HoldsAt(Neg(F'), T)$) can succeed only on clauses (LP4), (LP1a), (LP1b) or (LP2) (respectively (LP3)). Again, we consider each of these possibilities in turn:

(i) Success on (LP4): Trivially, $I^+(F, T) = true$ (respectively $I^+(F', T) = false$) by rule (4) in the definition of $\mathcal{F}$.

(ii) Success on (LP1a): Since the sub-goals $Given(\lambda(L), t_1), t_1 \prec T$ succeed, $I^+(F, T_1) = true$ (respectively $I^+(F', T_1) = false$) by rule (4) in the definition of $\mathcal{F}$ (where T_1 is the binding to t_1). It remains to show that there is no possible initiation point or possible termination point for F (respectively F') between T_1 and T, so that rule (1a) in the definition of $\mathcal{F}$ may be applied. Trivially, if the sub-goals $HappensAt(a, t_2)$, $T_1 \preceq t_2$ and $t_2 \prec T$ in the body of each $ClippedBetween$ clause collectively fail, there can be no such point. Now suppose these sub-goals succeed, binding t_2 to T_2. Since the call $AffectedBetween(T_1, \lambda(L), T_3)$ fails (so that the top level goal succeeds on clause (LP1a)), both of the calls $PossiblyInitiates(A, F, T_2)$ and $PossiblyTerminates(A, F, T_2)$ (respectively $PossiblyTerminates(A, F', T_2)$ and $PossiblyInitiates(A, F', T_2)$) fail. Hence, if there is a c-proposition in D of the

[12] We assume the convention that $\max(\emptyset) = 0$.

form "A **initiates** F **when** C" or "A **terminates** F **when** C" (respectfully "A **terminates** F' **when** C" or "A **initiates** F' **when** C"), there is a fluent literal $L_p \in C$ such that *not* $HoldsAt(\lambda(L_p), T_2)$ fails, i.e. $HoldsAt(\lambda(L_p), T_2)$ succeeds, say with SLDNF derivation α', where $length(\alpha') < n$. By the induction hypothesis, if $L_p = F_p$ then $I^+(F_p, T_2) = false$, and if $L_p = \neg F'_p$ then $I^+(F'_p, T_2) = true$. In either case, the c-proposition is therefore not applicable in the definition of a possible initiation point or possible termination point of F (respectively F') relative to I^+. Hence rule (1a) in the definition of $\mathcal{F}$ applies to give $I^+(F,T) = true$ (respectively $I^+(F',T) = false$) as required.

(iii) Success on (LP1b): The argument in this case is exactly analogous to case (ii), but using rule (1b) (instead of rule (1a)) in the definition of $\mathcal{F}$.

(iv) Success on (LP2): In this case there exists a $T_1 \in \Pi$, $T_1 \prec T$, such that for some $A \in \Delta$ the propositions "A **happens-at** T_1" and "A **initiates** F **when** $\{L_1, \ldots, L_k\}$" belong to D, and each of the calls $HoldsAt(\lambda(L_1), T_1), \ldots,$ $HoldsAt(\lambda(L_k), T_1)$ succeed. The successful SLDNF derivations of each of these calls are of length strictly less than n due to the successful call of $HappensAt(A, T_1)$ in the root SLDNF derivation of $HoldsAt(Pos(F), T)$. Hence by the inductive hypothesis, for each $L_i = F_i$, $I^+(F_i, T_1) = true$, and for each $L_j = \neg F_j$, $I^+(F_j, T_1) = false$. Hence T_1 is an initiation point for F relative to I^+. By an argument exactly analogous to that in case (ii) above, we can show from the finite failure of $ClippedBetween(T_1, Pos(F), T)$ that there are no possible termination points for F between T_1 and T, so that rule (2) in the definition of $\mathcal{F}$ applies to give $I^+(F,T) = true$ as required.

(v) Success on (LP3): The argument in this case is exactly analogous to case (iv), but using rule (3) (instead of rule (2)) in the definition of $\mathcal{F}$ to show that $I^+(F',T) = false$.

B.5. Proof of Proposition 8.1

Lemma B.1. Let $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$ be a fluent-finite, non-converging projection language, let $P(\Pi, \preceq)$ be an ordering program for $\mathcal{E}$, and let $D = \langle \gamma, \eta, \tau_i \rangle$ be a finite, initially-consistent, fluent-independent projection domain description in $\mathcal{E}$. Let H be a model of D. Let M be a ground list term such that the ground query $Member(\lambda, M)$ succeeds iff there is a fluent constant $F' \in \Phi$ such that either $\lambda = Neg(F')$ and $H(F', T_0) = false$ or $\lambda = F'$ and $H(F', T_0) = true$. Then for all $F \in \Phi$ and $T \in \Pi$,

$$EC[D, \emptyset, P(\Pi, \preceq)] \vdash_{SLDNF} HoldsAt(M, Pos(F), T)$$

if and only if $H(F,T) = true$, and

$$EC[D, \emptyset, P(\Pi, \preceq)] \vdash_{SLDNF} HoldsAt(M, Neg(F), T)$$

if and only if $H(F,T) = false$.

Proof. Let T and F be an arbitrary time-point and fluent constant. Since D is finite and non-converging, T has a unique, maximal, finite (possibly empty) sequence $T_1, \ldots, T_n$ associated with it such that $T_1 \prec \ldots \prec T_n \prec T$ and such that for each T_i there is an h-proposition in η of the form "A **happens-at** T_i". Proof is by induction on the length n of this sequence.

Base Case:
Clearly, if $n = 0$ the queries $HoldsAt(M, Pos(F), T)$ and $HoldsAt(M, Neg(F), T)$ can succeed only on clause (EC7), and will succeed if and only if the queries $Member(Pos(F), M)$ and $Member(Neg(F), M)$ succeed respectively. By the second condition in Definition 2.10 of a model, $H(F,T) = H(F,T_0)$, so that by definition of the list term M the lemma is true in the base case.

Inductive Step:
Suppose that the lemma is true for all time-points whose associated sequences are of length $m < n$. Then in particular it is true for T_n whose associated sequence is of length $n-1$. There are three cases to consider:

Case one: There is both an h-proposition in η of the form "A **happens-at** T_n" and a c-proposition in γ of the form "A **initiates** F **when** C" such that H satisfies C at T_n. Hence by the third condition in the definition of a model, $H(F,T) = true$.

In this case, by the inductive hypothesis and the program definition of $Initiates$, the query $HoldsAt(M, Pos(F), T)$ will succeed on clause (EC8) with the program variable t_1 in the body of the clause bound to T_n. The query $HoldsAt(M, Neg(F), T)$ will fail on clause (EC7) because the sub-goal $ClippedBetween(M, T_0, Neg(F), T)$ will succeed on clause (EC11) with the program variable t_2 in the body of the clause bound to T_n. The query $HoldsAt(M, Neg(F), T)$ will fail on clause (EC9) because by the inductive hypothesis and fluent-independence of D it will fail on the sub-goal $Terminates(M, a, F, T_n)$ for all bindings of the variable a provided by solutions to $Happens(a, T_n)$. Hence in this case the lemma is true.

Case two: There is both an h-proposition in η of the form "A **happens-at** T_n" and a c-proposition in γ of the form "A **terminates** F **when** C" such that H satisfies C at T_n. Hence by the third condition in the definition of a model, $H(F,T) = false$.

In this case, by the inductive hypothesis and the program definition of $Initiates$, the query $HoldsAt(M, Neg(F), T)$ will succeed on clause (EC9) with the program variable t_1 in the body of the clause bound to T_n. The query $HoldsAt(M, Pos(F), T)$ will fail on clause (EC7) because the sub-goal $ClippedBetween(M, T_0, Pos(F), T)$ will succeed on clause (EC10) with the program variable t_2 in the body of the clause bound to T_n. The query $HoldsAt(M, Pos(F), T)$ will fail on clause (EC8) because by the inductive hypothesis and fluent-independence of D it will fail on the sub-goal $Initiates(M, a, F, T_n)$ for all bindings of the variable a provided by solutions to $Happens(a, T_n)$. Hence in this case the lemma is also true.

Case three: There is not both an h-proposition in η of the form "A **happens-at** T_n" and a c-proposition in γ either of the form "A **initiates** F **when** C" or of the form "A **terminates** F **when** C" such that H satisfies C at T_n. Hence by the second condition in the definition of a model, $H(F,T) = H(F,T_n)$.

Clearly in this case, by the inductive hypothesis and by the program definitions of $Initiates$ and $Terminates$, the queries $HoldsAt(M, Pos(F), T)$ and $HoldsAt(M, Neg(F), T)$ can succeed on the clauses (EC8) and (EC9) respectively only with the variable t_1 in the body of each clause bound to some time-point $T_i < T_n$. Moreover, by the same argument, for all $T' < T_n$, the queries $ClippedBetween(M, T', Pos(F), T)$ and $ClippedBetween(M, T', Neg(F), T)$ succeed if and only if the queries $ClippedBetween(M, T', Pos(F), T_n)$ and $ClippedBetween(M, T', Neg(F), T_n)$ succeed respectively. Hence the queries $HoldsAt(M, Pos(F), T)$ and $HoldsAt(M, Neg(F), T)$ succeed if and only if the queries $HoldsAt(M, Pos(F), T_n)$ and $HoldsAt(M, Neg(F), T_n)$ succeed respectively. Hence in this case the lemma is also true.

STATEMENT OF MAIN PROPOSITION: *Let $\mathcal{E} = \langle \Pi, \preceq, \Delta, \Phi \rangle$ be a fluent-finite, non-converging projection language, let $P(\Pi, \preceq)$ be an ordering program for $\mathcal{E}$, and let $D = \langle \gamma, \eta, \tau_i \rangle$ be a finite, initially-consistent, fluent-independent projection domain description in $\mathcal{E}$. Let τ_{ob} be a finite observation set. Then for any fluent literal L of $\mathcal{E}$ and any $T \in \Pi$,*

$$EC[D, \tau_{ob}, P(\Pi, \preceq)] \vdash_{SLDNF} IHoldsAt(\lambda(L), T)$$

if and only if

$$D, \tau_{ob} \models_i L \textbf{ holds-at } T$$

PROOF:
Let an *initial assignment* of D be defined as a function $M : \Phi \mapsto \{true, false\}$ such that $M(F) = true$ whenever there is an i-proposition in D of the form "**initially** F", and $M(F) = false$ whenever there is an i-proposition in D of the form "**initially** $\neg F$". Since D is initially consistent there exists at least one such function, and by Propositions 2.1 and 5.1 there is a one-to-one correspondence between initial assignments of D and models of D. We may therefore unambiguously refer to the model H *generated by* the initial assignment M.

Clearly, successive solutions to the sub-goals

$Setof(l, (Initially(l)), i),$
$Setof(f, (Fluent(f),\ not\ Initially(Pos(f)),\ not\ Initially(Neg(f))), p),$
$Permutation(p, c),\ Append(c, i, m),$

in clause (EC2) bind the variable m to an appropriate list representation of each such initial assignment in turn. Given such a ground list term M', by Lemma B.1, clause (EC6) and the definition of $Forall$, the goal

$$ConsistentWithObservations(M')$$

will succeed if and only if the model of D its corresponding initial assignment generates is consistent with each o-proposition in τ_{ob}. Hence successive solutions to the goal $IExplanation(m)$ bind the variable m to a list representation of each initial assignment which generates an i-model of D with τ_{ob}. Hence the proposition is true by clause (EC1), Lemma B.1 and the definition of $Forall$.

C. AN ORDERING PROGRAM FOR $\langle \Pi_\Delta, \leq_\Delta \rangle$

This appendix concerns the practical details of implementing Language $\mathcal{A}$ domains as Event Calculus style Prolog programs in the manner of Section 8, given the intermediate translations to $\mathcal{E}$ domain descriptions as defined in Section 3.

The following ordering program $P(\Pi_\Delta, \leq_\Delta)$ uses the Situation Calculus style terms

$$Result(A_n, Result(\ldots, Result(A_1, S0)\ldots))$$

and

$$Branch(A', Result(A_n, Result(\ldots, Result(A_1, S0)\ldots)))$$

to represent the Δ-sequences "$A_1, \ldots, A_n$" and "$A_1, \ldots, A_n, |A'|$" respectively:

$$t_1 \prec_\Delta t_2 \leftarrow ListForm(t_2, l_2),\ Append([h|r], l_1, l_2),\ ListForm(t_1, l_1).$$

$$t \preceq_\Delta t.$$

$$t_1 \preceq_\Delta t_2 \leftarrow t_1 \prec_\Delta t_2.$$

$$ListForm(S0, []).$$

$$ListForm(Branch(a,t), [B(a)|l]) \leftarrow ListForm(t, l).$$

$$ListForm(Result(a,t), [R(a), B(a)|l]) \leftarrow ListForm(t, l).$$

In addition, the complete occurrence set of Δ is represented by a single clause which replaces all the domain-dependent ground *HappensAt* clauses of Definition 8.2:

$$HappensAt(a, Branch(a,t)).$$

The important feature of the above ordering program is that not only does it correctly deal with ground queries of the form "$T \preceq T'$" (where T and T' are Situation Calculus style representations of Δ-sequences as described above), but it also gives all correct solutions to queries of the form "$t \preceq T'$" (where t is a variable). This enables the sub-goals in clauses (EC8)–(EC11) to be re-ordered as follows:

$$HoldsAt(m, Pos(f), t_3) \leftarrow \qquad \text{(EC8}'\text{)}$$
$$t_1 \prec t_3,\ HappensAt(a, t_1),\ Initiates(m, a, f, t_1),$$
$$not\ ClippedBetween(m, t_1, Pos(f), t_3).$$

$$HoldsAt(m, Neg(f), t_3) \leftarrow \qquad \text{(EC9}'\text{)}$$
$$t_1 \prec t_3,\ HappensAt(a, t_1),\ Terminates(m, a, f, t_1),$$
$$not\ ClippedBetween(m, t_1, Neg(f), t_3).$$

$$ClippedBetween(m, t_1, Pos(f), t_3) \leftarrow \qquad \text{(EC10}'\text{)}$$
$$t_2 \prec t_3,\ t_1 \preceq t_2,\ HappensAt(a, t_2),\ Terminates(m, a, f, t_2).$$

$$ClippedBetween(m, t_1, Neg(f), t_3) \leftarrow \qquad \text{(EC11}'\text{)}$$
$$t_2 \prec t_3,\ t_1 \preceq t_2,\ Happens(a, t_2),\ Initiates(m, a, f, t_2).$$

This re-ordering avoids problems that would otherwise arise from calls to *HappensAt* with an unground second argument.