A Simple Declarative Language for Describing Narratives with Actions

Antonios Kakas
Dept. of Computer Science
University of Cyprus
75 Kallipoleos Street
P.O. Box 537
CY-1678 Nicosia
CYPRUS
antonis@turing.cs.ucy.ac.cy

Rob Miller
Department of Computing
Imperial College
180 Queen’s Gate
London SW7 2BZ, U.K.
rsm@doc.ic.ac.uk
http://laozu.doc.ic.ac.uk/
UserPages/stuff/rsm/rsm.html

August 1995

Joint
Imperial College / University of Cyprus
Research Report

Imperial College Research Report Number DoC 95/12
(also submitted for journal publication, August 1995)

Abstract

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
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.

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 recognized elsewhere (see [2], [27] or [25]). 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

$\text{Result(Shoot, Result(Wait, Result(Compartment, S)))}$

unless there is some extra mechanism to relate the sequence of actions $\text{Compartment} - \text{Wait} - \text{Shoot}$ 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] and Kowalski and Sergot’s Event Calculus [20]. On the other hand, formalisms such as the McCarthy and Hayes’ Situation Calculus [23], Dynamic Logics (see for example [14]) and Gelfond and Lifschitz’s Language $\mathcal{A}$ [13] are not narrative-based. This is not to say that they cannot be extended to deal with narrative information (see for example [27], [25] or [21]). But the notions of an independent flow of time and of an action occurrence are not central to their underlying ontology.

In [13], 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 two papers by Kartha [17], [18]. The first of these papers uses translations from $\mathcal{A}$ to show the equivalence of three well-known characterizations of the Situation Calculus for a whole class
of domains, and the second uses $\mathcal{A}$ to reveal a potential incompatibility between Baker's circumscriptive formalism [3] and a deductive view of explanation. 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 formalism described in [30].

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 $\mathcal{L}_8$, a narrative extension of $\mathcal{A}$ recently proposed by Baral, Gelfond and Provetti [5]. 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 $\text{HoldsAt}$ 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 1** [Domain Language] A domain language is a tuple $(\Pi, \preceq, \Delta, \Phi)$, where $\preceq$ is a partial (possibly total) ordering$^1$ defined over the non-empty set $\Pi$ of time points, $\Delta$ is a non-empty set of action constants, and $\Phi$ 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 $\mathcal{E} = (\Pi, \preceq, \Delta, \Phi)$. We will often write $T_1 \preceq T_2$ to mean $T_1 \leq T_2$ and $T_1 \neq T_2$.

$^1$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 $\mathcal{A}$. 

4
Definition 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 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 3 [c-proposition] A $c$-proposition in $\mathcal{E}$ is an expression either of the form

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

or of the form

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

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 "$A \text{ initiates } F \text{ when } \emptyset$" and "$A \text{ terminates } F \text{ when } \emptyset$" as "$A \text{ initiates } F$" and "$A \text{ terminates } F$" respectively.

Definition 4 [h-proposition] An $h$-proposition in $\mathcal{E}$ is an expression of the form

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

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

Definition 5 [t-proposition] A $t$-proposition in $\mathcal{E}$ is an expression of the form

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

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

Definition 6 [Domain description] A domain description in $\mathcal{E}$ is a triple $(\gamma, \eta, \tau)$, where $\gamma$ is a set of $c$-propositions, $\eta$ is a set of $h$-propositions, and $\tau$ 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. This is stated explicitly in condition (1) of Definition 10. Conditions (2) and (3) confirm that the terms initiate and terminate have their intended meanings, relative to this ‘commonsense’ principle.

**Definition 7** [Interpretation] An interpretation of \( E \) is a mapping

\[ H : \Phi \times \Pi \rightarrow \{ \text{true}, \text{false} \} \]

**Definition 8** [Point satisfaction] Given a set of fluent literals \( C \) of \( E \) and a time point \( T \in \Pi \), an interpretation \( H \) satisfies \( C \) at \( T \) iff for each fluent constant \( F \in C \), \( H(F, T) = \text{true} \), and for each fluent constant \( F' \) such that \( \neg F' \in C \), \( H(F', T) = \text{false} \).

**Definition 9** [Initiation/termination point] Let \( H \) be an interpretation of \( E \), let \( D = (\gamma, \eta, \tau) \) 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 \( \eta \) of the form “\( A \) happens-at \( T \)” and a \( c \)-proposition in \( \gamma \) of the form “\( A \) initiates \( F \) when \( C \)” (respectively “\( A \) terminates \( F \) when \( C \)”); and (ii) \( H \) satisfies \( C \) at \( T \).

**Definition 10** [Model] Given a domain description \( D = (\gamma, \eta, \tau) \) in \( E \), an interpretation \( H \) of \( E \) is a model of \( D \) iff, for every \( F \in \Phi \) and \( T, T', T_1, T_3 \in \Pi \) such 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 \( T_1 \prec T_2 \prec T_3 \), 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 \( H(F, T_3) = \text{true} \).
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 \( H(F, T_3) = \text{false} \).

---

2 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 (\( \text{frame fluents} \) in Lifschitz’s notation) and those which do not.

3 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 [28] the semantics uses “weak” initiation and termination (but see footnote, Appendix A.1).
4. For all t-propositions in \( \tau \) of the form "\( F \) holds-at \( T \)", \( H(F, T) = true \), and for all t-propositions of the form "\( \neg F \) holds-at \( T' \)", \( H(F, T') = false \).

\[ \square \]

**Definition 11** [Consistency] A domain description is **consistent** iff it has a model.

\[ \square \]

**Definition 12** [Entailment] A domain description \( D \) **entails** the t-proposition "\( F \) 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 \( D \), \( H(F', T) = false \).

\[ \square \]

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 \( c \)-propositions in a Language \( A \) 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 \( A \) 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\(^4\).

Condition (4) in Definition 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 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.

\(^4\)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.
Example 1 This example is intended to illustrate the necessity of including the first condition in Definition 10 of a Language $E$ 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, $E_v = \langle \mathbb{R}, \le, \{\text{InjectA}, \text{InjectB}, \{\text{Protected}, \text{TypeO}\} \rangle$. The domain description $D_v$ consists of two c-propositions, two h-propositions and a single t-proposition:

\begin{align*}
\text{InjectA} & \text{ initiates } \text{ Protected when } \{\text{TypeO}\} \\
\text{InjectB} & \text{ initiates } \text{ Protected when } \{\neg\text{TypeO}\} \\
\text{InjectA} & \text{ happens-at 2} \\
\text{InjectB} & \text{ happens-at 3} \\
\neg\text{Protected} & \text{ holds-at 1}
\end{align*}

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 type 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,

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

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 $\neg\text{TypeO}$ to $\text{TypeO}$ at 2.30, thus rendering both vaccinations ineffective. \hfill \Box

Example 2 This example shows that the Language $E$ 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. Let $E_y = \langle \mathbb{R}^+, \le, \{\text{Shoot}, \{\text{Alive}, \text{Loaded}\}\rangle$, where $\mathbb{R}^+$ signifies the non-negative real numbers, and let the domain description $D_y$ consist of a single c-proposition, a single h-proposition and twot-propositions:

\begin{align*}
\text{Shoot} & \text{ terminates } \text{ Alive when } \{\text{Loaded}\} \\
\text{Shoot} & \text{ happens-at 2} \\
\text{Alive} & \text{ holds-at 1} \\
\neg\text{Alive} & \text{ holds-at 3}
\end{align*}
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

\[ D_{ys} \models Loaded \text{ holds-at } 2 \]

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

\[ D_{ys} \models Loaded \text{ holds-at } n \]

\[ \square \]

Two properties of \(E\) will prove useful later. The first is that \(E\) 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 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 [22]. 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 13** [Occurrence Sparsity] Let \(D = \langle \gamma, \eta, \tau \rangle\) be a domain description written in a language \(E = (\Pi, \succeq, \Delta, \Phi)\). \(D\) and \(\eta\) are occurrence-sparse iff for any two points \(T_1, T_2 \in \Pi\) there are only a finite number of \(h\)-propositions in \(\eta\) of the form “A happens-at \(T\)” such that \(T_1 \preceq T < T_2\).

**Proposition 1** Let \(D\) be an occurrence-sparse domain description written in a language \(E = (\Pi, \succeq, \Delta, \Phi)\), 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 [7]. 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 1.
3 Simulating the Language $A$ as a Class of Language $E$ Domains

Because they use different ontologies, any correspondence between the Languages $A$ and $E$ must inevitably be expressed in rather artificial terms. In this section we show that Language $A$ domains can be simulated or reformulated as Language $E$ domains with a branching structure $(\Pi, \leq)$ of time points, analogous to the branching 'tree' of situations often incorporated in formulations of the Situation Calculus. Since the Language $A$ is not narrative-based and so does not directly include the notion of action occurrences, in our reformulation 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 $\text{Result}(A, S)$. To express this, we need to insert an extra time-point between $S$ and $\text{Result}(A, S)$ – 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 14** $[\Delta$-sequence$]$ Given a set $\Delta$ of action constants, a $\Delta$-sequence of $\Delta$ is defined inductively as follows:

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

\[ \square \]

**Definition 15** $[\Delta$-ordering$]$ Given a set $\Delta$ of action constants and the corresponding set $\Pi_\Delta$ of all $\Delta$-sequences, the $\Delta$-ordering $\leq_\Delta$ over $\Pi_\Delta$ is defined as follows:

- For all $S \in \Pi_\Delta$, $\langle \rangle \leq_\Delta S$.
- For all $\langle a_1, \ldots, a_n \rangle \in \Pi_\Delta$ and for all $m$ such that $1 \leq m \leq n$, $\langle a_1, \ldots, a_m \rangle \leq_\Delta \langle a_1, \ldots, a_n \rangle$.

\[ \square \]
Examples:
Suppose $\Delta = \{\text{Wait}, \text{Load}, \text{Shoot}\}$. The $\Delta$-sequence
$$\langle [\text{Load}], \text{Load}, [\text{Wait}], \text{Wait}, [\text{Shoot}], \text{Shoot} \rangle$$
corresponds to the Situation Calculus term
$$\text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0)))$$
and (regarding situations as arranged in a branching tree structure) the $\Delta$-sequence
$$\langle [\text{Load}], \text{Load}, [\text{Wait}], \text{Wait}, [\text{Shoot}] \rangle$$
corresponds to the inter-connecting arc between the situations
$$\text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0))$$
and
$$\text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0)))$$
Regarding the ordering $\preceq_\Delta$, it is easy to see that, for example
$$\langle [\text{Load}] \rangle \preceq_\Delta \langle [\text{Load}], \text{Load} \rangle$$
$$\langle [\text{Load}], \text{Load} \rangle \preceq_\Delta \langle [\text{Load}], \text{Load}, [\text{Wait}], \text{Wait}, [\text{Shoot}], \text{Shoot} \rangle$$

Notation:
We shall sometimes refer to the $\Delta$-sequence
$$\langle [A_1], A_1, \ldots, [A_n], A_n \rangle$$
simply as $A_1, \ldots, A_n$ and refer to the $\Delta$-sequence
$$\langle [A_1], A_1, \ldots, [A_n], A_n, [A'] \rangle$$
as $A_1, \ldots, A_n, [A']$. Notice that in this notation
$$A_1, \ldots, A_n, [A'] \preceq_\Delta A_1, \ldots, A_n, A'$$
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 16 [Complete occurrence set] Let $\mathcal{E} = \langle \Pi_\Delta, \preceq_\Delta, \Delta, \Phi \rangle$, where $\Pi_\Delta$ is the set of $\Delta$-sequences of $\Delta$ and $\preceq_\Delta$ is the $\Delta$-ordering over $\Pi_\Delta$. The set $\eta_\Delta$, called the complete occurrence set of $\Delta$, is the set of all $h$-propositions of $\mathcal{E}$ either of the form
$$A \text{ happens-at } \langle [A] \rangle$$
or of the form
\[ A_n \text{ happens-at } \langle |A_1|, A_1, \ldots, |A_{n-1}|, A_{n-1}, |A_n| \rangle \]

The following proposition shows a sense in which the Language \( A \) may be regarded as a special case of the Language \( E \).

**Proposition 2** Let \( D_A \) be a consistent theory written in a language \( A \) in the sense of \([13]\), with a set of action constants \( \Delta \) and a set of fluent constants \( \Phi \). Let \( E = \langle \Pi_\Delta, \leq_\Delta, \Delta, \Phi \rangle \), where \( \Pi_\Delta \) is the set of \( \Delta \)-sequences of \( \Delta \) and \( \leq_\Delta \) is the \( \Delta \)-ordering over \( \Pi_\Delta \). Let \( D_E \) be the domain description \( \langle \gamma, \eta_\Delta, \tau \rangle \) in \( E \) defined as follows:

- \( \eta_\Delta \) is the complete occurrence set of \( \Delta \)
- For each \( v \)-proposition in \( D_A \) of the form \( L \) after \( A_1; \ldots; A_m \) there is a \( t \)-proposition in \( \tau \) of the form \( L \text{ holds-at } A_1, \ldots, A_m \), and for each \( v \)-proposition in \( D_A \) of the form \( \text{initially } L \) there is a \( t \)-proposition in \( \tau \) of the form \( L \text{ holds-at } \langle \rangle \)
- For each \( F \in \Phi \), then for each \( e \)-proposition in \( D_A \) of the form \( A \text{ causes } F \text{ if } L_1, \ldots, L_n \) there is a \( e \)-proposition in \( \gamma \) of the form \( A \text{ initiates } F \text{ when } \{ L_1, \ldots, L_n \} \), and for each \( e \)-proposition in \( D_A \) of the form \( A' \text{ causes } \neg F \text{ if } L'_1, \ldots, L'_n \) there is an \( e \)-proposition in \( \gamma \) of the form \( A' \text{ terminates } F \text{ when } \{ L'_1, \ldots, L'_n \} \)

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

- \( D_E \models F \text{ holds-at } \langle \rangle \) if and only if \( D_A \) entails \( \text{initially } F \)
- \( D_E \models \neg F \text{ holds-at } \langle \rangle \) if and only if \( D_A \) entails \( \text{initially } \neg F \)
- \( D_E \models F \text{ holds-at } A_1, \ldots, A_n \) if and only if \( D_A \) entails \( F \text{ after } A_1; \ldots; A_n \)
- \( D_E \models \neg F \text{ holds-at } A_1, \ldots, A_n \) if and only if \( D_A \) entails \( \neg F \text{ after } A_1; \ldots; A_n \)

**Proof** Let \( R \) be the unique transition function such that there is a model \((\sigma_0, R)\) of \( D_A \) (for definitions see \([13]\)). 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:

- \( H_{[\sigma, R]}(F, \langle \rangle) = H_{[\sigma, R]}(\langle |A'| \rangle) = \text{true} \) if and only if \( F \in \sigma \)
Clearly, for each \( \sigma \subseteq \Phi \), \( H_{[\varepsilon, R]} \) is a model of \( \langle \gamma, \eta, \emptyset \rangle \), and \( H_{[\varepsilon, R]} \) is a model of \( D_E = \langle \gamma, \eta, \tau \rangle \) if and only if \( (\sigma, R) \) is a model of \( D_A \) in the sense of [13]. Since \( D_E \) is occurrence-sparse, by Proposition 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 \( H_{[\varepsilon, R]} \).

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]. \( \mathcal{L}_0 \) 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 [27]. 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 \((\Psi, \Sigma)\). The “\( \Psi \)” component roughly corresponds to the notion of a (partial) “transition function” in a Language \( \mathcal{A} \) model. For a given domain, \( \Psi \) 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 \( \langle \gamma, \eta, \tau \rangle \) as described in Proposition 2. Given such a translation, the “\( \Sigma \)” 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 \( \Delta \)-sequence \( \delta_N \) in \( \Pi_\Delta \), together with an assignment of each other situation symbol \( s_i \) to a \( \Delta \)-sequence \( \delta_i \) such that \( \delta_i \leq_\Delta \delta_N \). (In Pinto and Reiter’s terms, \( \delta_N \) represents the ‘actual path of situations’.) To be acceptable, the assignment \( \delta_N \) 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 \( \delta_N \) to be of the form \( \langle a_1, \ldots, a_i, [A_1], A_1, [A_2], A_2, \ldots, a_n \rangle \) (where \( \langle a_1, \ldots, a_i \rangle \) is the \( \Delta \)-sequence \( \delta_i \) assigned to \( s_i \)), and the precedence fact “\( s_i \) precedes \( s_j \)” expresses the constraint \( \delta_i \leq_\Delta \delta_j \). The minimal length requirement for \( \delta_N \) corresponds to the minimisation of the \text{occurs} predicate in Pinto and Reiter’s extended Situation Calculus.

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 nonmonotonocity 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 [19] as an illustration.

**Example 3** Let $\mathcal{E}_s = \langle \mathcal{N}, \leq, \{\text{Park, Steal}\}, \{\text{Parked}\} \rangle$, where $\mathcal{N}$ signifies the natural numbers, and let $D_s$ be the domain description consisting of the following two c-propositions, single h-proposition and single t-proposition:

- **Park initiates Parked**
- **Steal terminates Parked**
- **Park happens-at 2**
- **¬Parked holds-at 6**

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

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

\[
\{ \text{Steal happens-at 3, Steal happens-at 4} \}
\]
\[
\{ \text{Steal happens-at 4, Park happens-at 8} \}
\]
\[
\{ \text{Steal happens-at 5} \}
\]

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 18** [Preferable \( h \)-explanation]\(^5\)

Let \( \eta \) and \( \eta' \) be \( h \)-explanations for \( D \). \( \eta \) is preferable to \( \eta' \) iff \( \eta \subseteq \eta' \). \( \square \)

Having identified a class of explanations, such as the class of \( h \)-explanations in Definition 17, and a preference criterion such as that of Definition 18, it is possible to construct a corresponding meta-level, explanation-based ‘semantics’ similar to the semantics of Abductive Logic Programming [15], [16], [10]. 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 19** [Optimal \( h \)-explanation] \( \eta_c \) is an optimal \( h \)-explanation for \( D \) iff \( \eta_c \) is an \( h \)-explanation for \( D \) and there is no other \( h \)-explanation \( \eta'_c \) for \( D \) such that \( \eta'_c \) is preferable to \( \eta_c \). \( \square \)

**Definition 20** [\( h \)-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 \( \eta_c \) for \( D \) such that \( H \) is a model of \( \langle \gamma, \eta \cup \eta_c, \tau \rangle \). \( \square \)

**Definition 21** [\( h \)-consistency] A domain description is \( h \)-consistent iff it has at least one \( h \)-model. \( \square \)

\(^5\)In this definition and throughout the paper “\( \subset \)” is intended to mean “is contained in and not equal to”.

15
**Definition 22 [h-entailment]** The domain description $D = \langle \gamma, \eta, \tau \rangle$ \textit{h-entails} the $t$-proposition “$F \text{ holds-at } T$”, written “$D \models_h F \text{ holds-at } T$”, iff for every $h$-model $H$ of $D$, $H(F, T) = true$. $D$ \textit{h-entails} the $t$-proposition “$\neg F' \text{ holds-at } T$” 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 19 to 22 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 22 above, this is indeed the case.

**Proposition 3** 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:

\[
\{ \text{Steal happens-at } 3 \} \\
\{ \text{Steal happens-at } 4 \} \\
\{ \text{Steal happens-at } 5 \}
\]

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

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

and for all $n \geq 6$,

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

16
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 [29]. 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 [12] 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 $\tau_I$ and $\tau_G$ of t-propositions, domain information can be modelled as a set $\gamma$ of o-propositions, and a plan can be regarded as a single (optimal) h-explanation for $\langle \gamma, \emptyset, \tau_I \cup \tau_G \rangle$.

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] (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 minimal.

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

- $F_1$ holds-at 4
- $F_2$ holds-at 5
- $\neg F_2$ 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 23** [hc-explanation] Let $D = \langle \gamma, \eta, \tau \rangle$ be a domain description. An *hc-explanation* for $D$ is a pair $\langle \gamma, \eta, \tau \rangle$, where $\gamma$ is a (possibly empty) set of c-propositions and $\eta, \tau$ are (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 23 the following are all hc-explanations for $D_{ex}$:

1. $\langle \{A_1 \text{ terminates } F_2 \text{ when } \{F_1\}\}, \{A_1 \text{ happens-at 8}\} \rangle$
2. $\langle \{A_1 \text{ terminates } F_2\}, \{A_1 \text{ happens-at 8}\} \rangle$
3. $\langle \{A_1 \text{ terminates } F_2 \text{ when } \{\neg F_1\}, A_2 \text{ terminates } F_1\}$
   \hspace{1cm} $\{A_2 \text{ happens-at 7}, A_1 \text{ happens-at 8}\} \rangle$

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 $\epsilon$ over hc-explanation $\epsilon'$ only if $\epsilon$'s set of c-propositions was strictly contained in $\epsilon'$'s. However, this policy would not yield any preferences between the three hc-explanations in Example 4 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.

For the purposes of illustration, we will formulate a definition of preference between two hc-explanations which gives preference to hc-explanation (1) above over both (2) and (3). For the sake of argument, we will suppose that we want to prefer the inclusion in an hc-explanation of the c-proposition “$A_1 \text{ terminates } F_2 \text{ when } \{F_1\}$” to the inclusion of the c-proposition “$A_1 \text{ terminates } F_2$” because the former is more specific – it necessitates changes in a smaller variety of circumstances. We will also suppose that we wish to prefer hc-explanation (1) to hc-explanation (3) because the latter contains the ‘extra’ c-proposition “$A_2 \text{ terminates } F_1$”. To state this last type of preference we will need a way of expressing that two hc-propositions such as “$A_1 \text{ terminates } F_2 \text{ when } \{F_1\}$” and “$A_1 \text{ terminates } F_2 \text{ when } \{\neg F_1\}$” are in some sense equivalent, or can be “traded-off” with each other when evaluating a potential preference between two hc-explanations. These considerations motivate the following definitions.

**Definition 24** [fluent complement] Given a fluent literal $L$ of $E = \langle \Pi, \leq, \Delta, \Phi \rangle$, the *complement* of $L$, written $\overline{L}$, is defined to be

- $\neg F$ if $L = F$ for some $F \in \Phi$
- $F'$ if $L = \neg F'$ for some $F' \in \Phi$
Definition 25 [c-specificity] Given the four c-propositions
\[ P_1 = A \text{ initiates } F \text{ when } C_1 \]
\[ P_2 = A \text{ initiates } F \text{ when } C_2 \]
\[ P_3 = A \text{ terminates } F \text{ when } C_1 \]
\[ P_4 = A \text{ terminates } F \text{ when } C_2 \]

\( P_1 \) is as c-specific as \( P_2 \), and \( P_3 \) is as c-specific as \( P_4 \), iff for every fluent literal \( L \in C_2 \), either \( L \in C_1 \) or \( \overline{L} \in C_1 \). □

Definition 26 [c-containment] Let \( \gamma \) and \( \gamma' \) be two sets of c-propositions. Then \( \gamma \) is c-contained in \( \gamma' \), written \( \gamma \subseteq_c \gamma' \) iff for every \( P \in \gamma \) there is a \( P' \in \gamma' \) such that \( P \) is as c-specific as \( P' \). Otherwise, \( \gamma \) is not c-contained in \( \gamma' \), written \( \gamma \not\subseteq_c \gamma' \). □

We can now state our desired preference criterion between hc-explanations. Clearly, the definition of hc-preference given below could be used as a substitute for h-preference in Definitions 19 to 22 of the previous section, giving rise to an analogous notion of “hc-entailment”.

Definition 27 [Preferable hc-explanation] Let \( \epsilon = \langle \gamma_\epsilon, \eta_\epsilon \rangle \) and \( \epsilon' = \langle \gamma'_\epsilon, \eta'_\epsilon \rangle \) be hc-explanations for \( D \). \( \epsilon \) is hc-preferable to \( \epsilon' \) iff either (i) \( \gamma_\epsilon \subseteq_c \gamma'_\epsilon \) and \( \gamma'_\epsilon \not\subseteq_c \gamma_\epsilon \), or (ii) \( \gamma_\epsilon \subseteq_c \gamma'_\epsilon, \gamma'_\epsilon \subseteq_c \gamma_\epsilon \), and \( \eta_\epsilon \subseteq \eta'_\epsilon \). □

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 respectively⁶. 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.

⁶Note that the class of Situation-Calculus-style languages of the form \((\Pi_\Delta, \leq_\Delta, \Delta, \Phi)\) defined in Proposition 2 of Section 3 are all examples of projection languages.
Definition 28 [Projection Language] A projection language is a domain language $\langle \Pi, \preceq, \Delta, \Phi \rangle$, where $\Pi$ 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 $\Pi$. It will be convenient to identify a particular type of t-proposition which we will call an i-proposition ("i" for "initial").

Definition 29 [i-proposition] An i-proposition in $\mathcal{E}$ is a t-proposition of the form

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

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

initially $L$

Definition 30 [Projection domain description] A projection domain description in $\mathcal{E}$ is a triple $\langle \gamma, \eta, \pi \rangle$, where $\gamma$ is a set of c-propositions, $\eta$ is a set of h-propositions, and $\pi$ is a set of i-propositions in $\mathcal{E}$.

Example 5 The following projection domain description uses the projection domain language $\mathcal{E}_{\text{s}} = \langle \mathbb{R}^+, \preceq, \{\text{Shoot}\}, \{\text{Alive, Loaded}\} \rangle$ of Example 2, where $\mathbb{R}^+$ signifies the non-negative real numbers (so that $T_0 = 0$).

Shoot terminates Alive when \{Loaded\}

Shoot happens-at 2

initially Alive

initially Loaded

At first sight it appears that the restriction of the set $\pi$ 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 31** [Initial consistency] Let \( D = (\gamma, \eta, \tau) \) 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 \( \tau \).

The next definition is related to the notion of “e-consistency” defined by Denekerk and De Schreye [9] in the context of the Language \( \mathcal{A} \).

**Definition 32** [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 \( E \) 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 33** [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 34** [Non-convergence] Let \( D \) be a domain description written in a language \( \mathcal{E} = (\Pi, \preceq, \Delta, \Phi) \). \( D \) and \( \mathcal{E} \) are non-converging iff for every three (not necessarily distinct) time points \( T_1, T_2 \) and \( T_3 \) in \( \Pi \) 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 4** 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 model this type of explanation by regarding 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. Thus we re-introduce \( t \)-propositions not simply as additional statements of the domain description but as observations that need to be explained.
This separation of observations from other parts of the domain theory appears elsewhere (see [6] for a discussion). In particular, it is used in [9] in the study of translations of $\mathcal{A}$ domains into logic programs.

**Definition 35** [\(\alpha\)-proposition] An \(\alpha\)-proposition in $\mathcal{E}$ is a t-proposition of the form

\[
L \text{ holds-at } T
\]

where $T \neq T_0$. □

**Definition 36** [Observation set] An observation set is a non-empty set of \(\alpha\)-propositions. □

**Definition 37** [i-explanation] Let $D = \langle \gamma, \eta, \tau_i \rangle$ be a projection domain description and let $\tau_{ob}$ be an observation set. An i-explanation for $\tau_{ob}$ in $D$ is a set $\tau_c$ of i-propositions such that $\langle \gamma, \eta, \tau_i \cup \tau_c \rangle$ is consistent and such that for each $p \in \tau_{ob}$,

\[
\langle \gamma, \eta, \tau_i \cup \tau_c \rangle \models p
\]

□

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

**Definition 39** [i-consistency] Let $D$ be a projection domain description and let $\tau_{ob}$ be an observation set. $D$ is i-consistent with $\tau_{ob}$ iff there is at least one i-model of $D$ with $\tau_{ob}$. □
Definition 40 [i-entailment] Let $D$ be a projection domain description and let $\tau_{ob}$ be an observation set. $D$ with $\tau_{ob}$ 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 $\tau_{ob}$, $H(F,T) = true$. $D$ with $\tau_{ob}$ i-entails the t-proposition “$\neg F'$ holds-at $T$” iff for every i-model $H$ of $D$ with $\tau_{ob}$, $H(F',T) = false$. \hfill \Box

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-perference in the definitions above.

Proposition 5 Let $D = \langle \gamma, \eta, \pi \rangle$ be an occurrence-sparse projection domain description and let $\tau_{ob}$ be an observation set. Then $H$ is a model of $\langle \gamma, \eta, \pi \cup \tau_{ob} \rangle$ if and only if $H$ is an i-model of $D$ with $\tau_{ob}$.

Proof See Appendix B.3 \hfill \Box

Example 6 Let $\mathcal{E}_{ys}$ be $\langle \mathbb{R}^+, \leq, \{\text{Shoot}\}, \{\text{Alive}, \text{Loaded}\} \rangle$, where $\mathbb{R}^+$ signifies the non-negative real numbers, and let the projection description $D_{ys}$ consist of a single c-proposition and a single h-proposition:

 Shoot terminates Alive when \{Loaded\}

Shoot happens-at 2

Let $\tau_{ys}$ be the observation set containing the following two o-propositions:

 Alive holds-at 1

$\neg$Alive holds-at 3

It is easy to see that there is a unique i-explanation for $\tau_{ys}$ in $D_{ys}$:

\{ initially Loaded, initially Alive \}

Hence for all $n \geq 0$

$D_{ys}, \tau_{ys} \models_{i} \text{Loaded holds-at } n$. \hfill \Box

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. After giving the appropriate definitions below, we conclude this section with an example involving explanations in terms of both i-propositions and h-propositions.
**Definition 41** [ih-explanation] Let $D = \langle \gamma, \eta, \tau_i \rangle$ be a projection domain description and let $\tau_{ob}$ be an observation set. An *ih-explanation* for $\tau_{ob}$ in $D$ is a pair $\langle \eta_c, \tau_{ic} \rangle$, where $\eta_c$ is an occurrence-sparse set of h-propositions and $\tau_{ic}$ is a set of i-propositions, such that $\langle \gamma, \eta \cup \eta_c, \tau_i \cup \tau_{ic} \rangle$ is consistent and such that for each $p \in \tau_{ob}$,

$$
\langle \gamma, \eta \cup \eta_c, \tau_i \cup \tau_{ic} \rangle \models p
$$

**Definition 42** [Preferable ih-explanation]

Let $\epsilon = \langle \eta_c, \tau_{ic} \rangle$ and $\epsilon' = \langle \eta'_c, \tau'_{ic} \rangle$ be ih-explanations for $\tau_{ob}$ in $D$. $\epsilon$ is ih-preferable to $\epsilon'$ iff $\eta_c \subset \eta'_c$.

**Definition 43** [Optimal ih-explanation] $\epsilon$ is an *optimal ih-explanation* for $\tau_{ob}$ in $D$ iff $\epsilon$ is an ih-explanation for $\tau_{ob}$ in $D$ and there is no other ih-explanation $\epsilon'$ for $\tau_{ob}$ in $D$ such that $\epsilon'$ is ih-preferable to $\epsilon$.

**Definition 44** [ih-model] Let $D = \langle \gamma, \eta, \tau_i \rangle$ be a projection domain description and let $\tau_{ob}$ be an observation set. $H$ is an *ih-model* of $D$ with $\tau_{ob}$ if there exists an optimal ih-explanation $\epsilon = \langle \eta_c, \tau_{ic} \rangle$ for $\tau_{ob}$ in $D$ such that $H$ is a model of $\langle \gamma, \eta \cup \eta_c, \tau_i \cup \tau_{ic} \rangle$.

**Definition 45** [ih-entailment] Let $D$ be a projection domain description and let $\tau_{ob}$ be an observation set. $D$ with $\tau_{ob}$ 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 $\tau_{ob}$, $H(F, T) = true$. $D$ with $\tau_{ob}$ ih-entails the t-proposition “$\neg F'$ holds-at $T$” iff for every ih-model $H$ of $D$ with $\tau_{ob}$, $H(F', T) = false$.

**Example 7** 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. Initially (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 regard the time that Fred left home as the least time point $T_0$ in the language, and represent Fred’s domain knowledge with a projection domain description $D_{rec}$ consisting of two c-propositions and an i-proposition:

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

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

initially $\neg$LightOn
Fred’s observation that the recording light is on when he returns home is represented by the singleton observation set $\tau_{rec}$:

$$\{\text{LightOn holds-at } T_R\}$$

where $T_R \in \mathbb{I}$ represents the time at which Fred returned home. It is easy to see that all optimal ih-explanations for $\tau_{rec}$ in $D_{rec}$ are of the form

$$\langle\{\text{TimerTrigger happens-at } T\}, \{\text{initially Working}\}\rangle$$

for some $T \in \mathbb{I}$ such that $T_0 \leq T < T_R$, so that

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

and hence the notion of ih-entailment correctly models Fred’s reasoning. \qed

## 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 and analogous assumption for h-propositions and c-propositions, this assumption does not extend to t-propositions (equivalent to $\text{Holds}$ or $\text{HoldsAt}$ 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 $\text{HoldsAt}$ predicate. In the program translations defined below, the t-proposition “$\neg F$ holds-at $T$” is represented by the positive literal $\text{HoldsAt}(\text{Neg}(F), T)$, whereas the negative literal not $\text{HoldsAt}(F, T)$ 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 [24]. Analogous techniques are used in [13], [11] 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 46 [Partial interpretation] A partial interpretation of \( \mathcal{E} \) is a partial mapping
\[
I : \Phi \times \Pi \rightarrow \{ \text{true}, \text{false} \}
\]

In the discussion which follows, we assume that Definitions 8 and 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 47 [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 48 [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 \( \eta \) of the form “\( A \) happens-at \( T \)” and a \( c \)-proposition in \( \gamma \) 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 10 of a model for a domain \( D \), we can define an associated (partial) operator on \( I \) as follows.

Definition 49 [Operator \( F \)] Given a domain description \( D = (\gamma, \eta, \tau) \) the partial operator \( F : \mathcal{I} \rightarrow \mathcal{I} \) 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 \( T_1 \leq T_2 \prec T \), then \( (F)(I)(F,T) = I(F,T_1) \).

(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 \( T \leq T_1 \prec T_2 \), then \( (F)(I)(F,T) = I(F,T_2) \).

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 \( (F)(I)(F,T) = true \).

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 \( (F)(I)(F,T) = false \).
4. If there is a t-proposition in \( \tau \) of the form "\( F \) holds-at \( T \)" , then
\[
(F)(I)(F,T) = \text{true},
\]
and if there is a t-proposition of the form "\( \neg F \) holds-at \( T' \)”,
\[
(F)(I)(F,T') = \text{false}.
\]
5. Otherwise \((F)(I)(F,T)\) is undefined.

\[\square\]

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 \((F)(I)(F,T)\) 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 10 of a model. The following three propositions show the relationship between Language \( \mathcal{E} \) models and the operator \( \mathcal{F} \).

**Proposition 6** Let \( D \) be a domain description. If \( D \) is consistent then any model \( H \) of \( D \) is a greatest (with respect to subset relation\(^7\)) 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** 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 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 8** Let \( D \) be a consistent domain description and let \( I^+ \) be the least fixed point of the following sequence of partial interpretations:

\[^7\text{For any two partial interpretations } l_1, l_2 \in \mathcal{I}, \text{ is a subset of } l_2, \text{ written } l_1 \subseteq l_2, \text{ iff for any } F \text{ and } T \text{ if } l_1(F,T) = \text{true then } l_2 = \text{true and if } l_1(F,T) = \text{false then } l_2(F,T) = \text{false.}\]
\[ I_0 = \emptyset \]
\[ I_{n+1} = I_n \cup F(I_n) \text{ for each countable ordinal } n > 0. \]

Then \( D \) entails any \( t \)-proposition of the form "\( F \) holds-at \( T' \)" (resp. "\( \neg F \) holds-at \( T' \)"") such that \( I^+(F, T) = \text{true} \) (resp. \( I^+(F, T) = \text{false} \)).

**Proof.** By Proposition 7 when \( D \) is consistent this sequence is well defined and \( I^+ \subseteq H \) for any model \( H \) of \( D \). \( \square \)

Given Proposition 8, we can use Definition 49 to ‘read off’ a logic program that implements \( F \) 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 50** [Ordering program] Given the language \( \mathcal{E} = \{ \Pi, \preceq, \Delta, \Phi \} \), 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 \( P(\Pi, \preceq) \):
  
  \( \text{HoldsAt}, \text{Given}, \text{ClippedBetween}, \text{HappensAt}, \text{PossiblyInitiates}, \text{Initiates}, \text{PossiblyTerminates}, \text{Terminates}, \text{AffectedBetween}. \)

\( \square \)

We will also need the following preliminary definitions.

**Definition 51** [lp-term and lp-complement] Given a fluent literal \( L \) of \( \mathcal{E} = \{ \Pi, \preceq, \Delta, \Phi \} \), the \( \text{lp-term} \) of \( L \), written \( \lambda(L) \), is defined to be

- \( F \) if \( L = F \) for some \( F \in \Phi \)
- \( \neg g(F') \) if \( L = \neg F' \) for some \( F' \in \Phi \)

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

- \( \neg g(F) \) if \( L = F \) for some \( F \in \Phi \)
- \( F' \) if \( L = \neg F' \) for some \( F' \in \Phi \)

\( \square \)

**Definition 52** [Finite domain description] The domain description \( \langle \gamma, \eta, \tau \rangle \) is finite iff \( \gamma \), \( \eta \) and \( \tau \) are all finite, and for each \( c \)-proposition in \( \gamma \) either of the form "\( A \text{ initiates } F \text{ when } C \)" or of the form "\( A \text{ terminates } F \text{ when } C \)". \( C \) is also finite. \( \square \)
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 49 of the operator $F$. The use of negation-as-failure and fluent converses in the domain-specific clauses defining the predicates PossiblyInitiates and PossiblyTerminates reflects Definition 48 of a possible initiation and possible termination point.

**Definition 53** \(LP[D, P(\Pi, \preceq)]\) Given a finite domain description \(D = (\gamma, \eta, \tau)\) written in the language \(\mathcal{E} = (\Pi, \preceq, \Delta, \Phi)\), 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

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

(LP1a)

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

(LP1b)

\[
HoldsAt(l, t_3) \leftarrow \\
\quad l \neq \text{Neg}(f), \text{HappensAt}(a, t_1), t_1 \prec t_3, \text{Initiates}(a, l, t_1), \text{not ClippedBetween}(t_1, l, t_3).
\]

(LP2)

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

(LP3)

\[
HoldsAt(l, t) \leftarrow \text{Given}(l, t).
\]

(LP4)

\[
\text{ClippedBetween}(t_1, l, t_3) \leftarrow \\
\quad l \neq \text{Neg}(f), \text{HappensAt}(a, t_2), t_1 \preceq t_2, t_2 \prec t_3, \text{PossiblyTerminates}(a, l, t_2).
\]

(LP5)

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

(LP6)

\[
\text{AffectedBetween}(t_1, l, t_3) \leftarrow \text{ClippedBetween}(t_1, l, t_3).
\]

(LP7)

\[
\text{AffectedBetween}(t_1, \text{Neg}(f), t_3) \leftarrow \text{ClippedBetween}(t_1, f, t_3).
\]

(LP8)

\[
\text{AffectedBetween}(t_1, f, t_3) \leftarrow \\
\quad f \neq \text{Neg}(f_1), \text{ClippedBetween}(t_1, \text{Neg}(f), t_3).
\]

(LP9)

and the following domain-specific clauses

...
• For each t-proposition “$L \text{ holds-at } T$” in $\tau$, the clause

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

• For each h-proposition “A happens-at $T$” in $\eta$, the clause

$$HappensAt(A, T).$$

• For each c-proposition “A initiates $F$ when $\{L_1, \ldots, L_n\}$” in $\gamma$, the clause

$$\begin{align*}
\text{Initiates}(A, F, t) &\leftarrow \\
\text{HoldsAt}(\lambda(L_1), t), \ldots, \text{HoldsAt}(\lambda(L_n), t).
\end{align*}$$

and the clause

$$\begin{align*}
\text{PossiblyInitiates}(A, F, t) &\leftarrow \\
\text{not HoldsAt}(\lambda(L_1), t), \ldots, \text{not HoldsAt}(\lambda(L_n), t).
\end{align*}$$

• For each c-proposition “A terminates $F$ when $\{L_1, \ldots, L_n\}$” in $\gamma$, the clause

$$\begin{align*}
\text{Terminates}(A, F, t) &\leftarrow \\
\text{HoldsAt}(\lambda(L_1), t), \ldots, \text{HoldsAt}(\lambda(L_n), t).
\end{align*}$$

and the clause

$$\begin{align*}
\text{PossiblyTerminates}(A, F, t) &\leftarrow \\
\text{not HoldsAt}(\lambda(L_1), t), \ldots, \text{not HoldsAt}(\lambda(L_n), t).
\end{align*}$$

Intuitively, given Propositions 6, 7 and 8 it is easy to see that the programs described above behave correctly for consistent domain descriptions, since Clauses (LP1a)-(LP4) correspond exactly to conditions (1a)-(4) in Definition 49. The following proposition confirms this intuition.

**Proposition 9** 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

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

then

30
Proof See Appendix B.4

To a certain extent, the above logic programs overcome the limitations of formalizations of action in normal logic programming identified by Gelfond and Lifschitz in [13]. 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. As shown by Proposition 9, the programs behave correctly even when the information entailed by their Language $E$ 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 10. This potential incompleteness is illustrated by the following example.

**Example 8** Let $E_v$ and $D_v$ be the domain language and domain description respectively of Example 1, and suppose that $P(\mathcal{R}, \leq)$ is an ordering program for $E_v$. Then the logic program $LP[D_v, P(\mathcal{R}, \leq)]$ consists of the program $P(\mathcal{R}, \leq)$ together with clauses (LP1a)-(LP9) of Definition 53 and the domain-specific clauses

\[
\text{Initiates}(InjectA, Protected, t) \leftarrow \text{HoldsAt}(TypeO, t).
\]

\[
\text{PossiblyInitiates}(InjectA, Protected, t) \leftarrow \\
\text{not} \text{HoldsAt}(\text{Neg}(TypeO), t).
\]

\[
\text{Initiates}(InjectB, Protected, t) \leftarrow \\
\text{HoldsAt}(\text{Neg}(TypeO), t).
\]

\[
\text{PossiblyInitiates}(InjectB, Protected, t) \leftarrow \\
\text{not} \text{HoldsAt}(TypeO, t).
\]

\[
\text{HappensAt}(InjectA, 2).
\]

\[
\text{HappensAt}(InjectB, 3).
\]

\[
\text{Given}(\text{Neg}(Protected), 1).
\]

As it stands, the query $\text{HoldsAt}(Protected, 4)$ 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 $\text{Given}(TypeO, n)$ or $\text{Given}(\text{Neg}(TypeO), n)$ to the program for some time point $n < 2$. The
example also illustrates the necessity of using the predicates PossiblyInitiates and PossiblyTerminates. Had the program definition of ClippedBetween been given simply in terms of Initiates and Terminates, the query

\[ \text{HoldsAt}(\text{Neg}(\text{Protected}), 4) \]

would succeed, even though the t-proposition \( \neg \text{Protected holds-at } 4 \) is not entailed by \( D_v \). Finally, notice that the success of the goal

\[ \text{HoldsAt}(\text{Neg}(\text{Protected}), 0) \]

(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 “\( \text{Initially } F \)" or the i-proposition “\( \text{Initially } \neg F \)" for every fluent constant \( F \), then \( D \) is ‘complete’ in the sense that it has at most one model (this follows from Proposition 1). If \( E \) is non-converging, and \( D \) is fluent-independent, \( D \) has exactly one model (this follows from Proposition 4). Hence, in the case where \( D \) is also finite (so that there are only a finite number of fluent constants in \( E \)), 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 \( \text{HoldsAt} \) predicate, rather than with a \( \text{Given} \) predicate as previously. In Definition 55 below, \( \text{HoldsAt}(M, F, T) \) should be read as “\( D \models F \) 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 \( \text{HoldsAt} \) are a simplification of clauses (LP1a)-(LP9) in Definition 53. 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 $\tau_{ob}$. 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 $\tau_{ob}$. (It is sufficient to consider only maximal i-explanations, i.e. those which mention every fluent in the language, because of the monotonicity of $E$ 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 55 below. In this definition, $I$HoldsAt($F, T$) should be read as “$D', \tau_{ob} \models_i F$ holds-at $T$”. $IExplanation(M)$ should be read as “$M$ is a (maximal) i-explanation for $\tau_{ob}$ in $D'$”.

In Definition 55, a Prolog-like syntax for lists is used. Thus the term [] represents the empty list and the term [Head|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, Forall(Condition, Goal) succeeds if for all solutions of Condition, Goal succeeds. Setof($X$, Goal, 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 [31]. (It is also assumed that the predicates holdsAt, Explanation, Permutation, and ConsistentWithObservations do not appear in the ordering program $P(\Pi, \preceq)$.)

**Definition 54** The domain language $E = \langle \Pi, \preceq, \Delta, \Phi \rangle$ is fluent-finite iff the set $\Phi$ is finite.

**Definition 55** $[EC[D, \tau_{ob}, P(\Pi, \preceq)]]$

Let $D = \langle \gamma, \eta, \tau \rangle$ be a finite projection domain description written in a fluent-finite projection language $E = \langle \Pi, \preceq, \Delta, \Phi \rangle$. Let $T_0$ be the null element of $\Pi$, and let $P(\Pi, \preceq)$ be an ordering program for $E$. Let $\tau_{ob}$ 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

\[
I\text{holsAt}(l, t) \leftarrow \quad \text{(EC1)}
\]

\[
\text{Forall}(IE\text{xplanation}(m), \text{HoldsAt}(m, l, t)).
\]
\( IExplanation(m) \) —
\( \text{Setof}(l, (\text{Initially}(l)), i), \)
\( \text{Setof}(f, (\text{Fluent}(f), \text{not Initially}(f), \text{not Initially}(\text{Neg}(f))), p), \)
\( \text{Permutation}(p, c), \text{Append}(c, i, m), \)
\( \text{ConsistentWithObservations}(m). \)

\( \text{Permutation}([], []). \) —
\( \text{Permutation}([f][r1],[f][r2]) \rightarrow \text{Permutation}(r1, r2). \)
\( \text{Permutation}([f][r1],[\text{Neg}(f)][r2]) \rightarrow \text{Permutation}(r1, r2). \)
\( \text{ConsistentWithObservations}(m) \) —
\( \text{Forall}(\text{Observation}(l, t), \text{HoldsAt}(m, l, t)). \)

\( \text{HoldsAt}(m, l, t3) \leftarrow \)
\( \text{Member}(l, m), \text{not ClippedBetween}(m, T_0, l, t3). \)

\( \text{HoldsAt}(m, l, t3) \leftarrow \)
\( l \neq \text{Neg}(f), \text{HappensAt}(a, t1), t1 < t3, \text{Initiates}(m, a, l, t1), \)
\( \text{not ClippedBetween}(m, t1, l, t3). \)

\( \text{HoldsAt}(m, \text{Neg}(f), t3) \leftarrow \)
\( \text{HappensAt}(a, t1), t1 < t3, \text{Terminates}(m, a, f, t1), \)
\( \text{not ClippedBetween}(m, t1, \text{Neg}(f), t3). \)

\( \text{ClippedBetween}(m, t1, l, t3) \leftarrow \)
\( l \neq \text{Neg}(f), \text{HappensAt}(a, t2), t1 \leq t2, t2 < t3, \)
\( \text{Terminates}(m, a, l, t2). \)

\( \text{ClippedBetween}(m, t1, \text{Neg}(f), t3) \leftarrow \)
\( \text{Happens}(a, t2), t1 \leq t2, t2 < t3, \)
\( \text{Initiates}(m, a, f, t2). \)

and the following domain-specific clauses

- For each fluent constant \( F \in \phi \), the clause
  \( \text{Fluent}(F). \)

- For each \( i \)-proposition "initially \( I \)" in \( \pi \), the clause
  \( \text{Initially}(\lambda(I)). \)
• For each o-proposition "L holds-at T" in τ_ob, the clause
  \[\text{Observation}(\lambda(L), T)\].

• For each h-proposition "A happens-at T" in η, the clause
  \[\text{HappensAt}(A, T)\].

• For each c-proposition "A initiates F when \{L_1, \ldots, L_n\}" in γ, the clause
  \[
  \text{Initiates}(m, A, F, t) \leftarrow
  \text{HoldsAt}(m, \lambda(L_1), t), \ldots, \text{HoldsAt}(m, \lambda(L_n), t).
  \]

• For each c-proposition "A terminates F when \{L_1, \ldots, L_n\}" in γ, the clause
  \[
  \text{Terminates}(m, A, F, t) \leftarrow
  \text{HoldsAt}(m, \lambda(L_1), t), \ldots, \text{HoldsAt}(m, \lambda(L_n), t).
  \]

\[\Box\]

Example 9 Let \(\mathcal{E}_{gsp}\), \(D_{gsp}\) and \(\tau_{gsp}\) be the domain language, domain description and observation set respectively of Example 6. Let \(P(\mathcal{H}^+; \leq)\) be an ordering program for \(\mathcal{E}_{gsp}\). Then the logic program \(\mathcal{P}(\mathcal{H}^+; \leq)\) consists of the program \(\mathcal{P}(\mathcal{H}^+; \leq)\) together with clauses (EC1)-(EC11) of Definition 55 and the domain-specific clauses

\[\text{Terminates}(m, \text{Shoot, Alive, t}) \leftarrow \text{HoldsAt}(m, \text{Loaded, t}).\]

\[\text{Fluent(Alive)}.\]

\[\text{Fluent(Loaded)}.\]

\[\text{Observation(Alive, 1)}.\]

\[\text{Observation(Neg(Alive), 3)}.\]

\[\text{HappensAt(Shoot, 2)}.\]

\[\Box\]
Although potentially somewhat inefficient, the programs described in Definition 55 are of interest because they enable sound derivations\footnote{In order to continue to refer to ‘SLDNF derivations’, we assume that the meta-level primitives \textit{Setof} and \textit{For all} are appropriately reinterpreted (see \cite{31} for details).} of i-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

\[
EC[D_{ysy}, \tau_{ys}, P(\mathbb{R}^+, \leq)] \vdash_{SLDNF} \text{IHoldsAt}(\text{Loaded}, 0)
\]

Indeed, for a wide class of domains they are both sound and ‘complete’, in the sense of Proposition 10 below. Since any finite, consistent Language $A$ domain as defined in \cite{13} may be translated directly into a Language $E$ projection domain description together with an observation set (see Section 3), finite $A$ domains may also be given a meta-level implementation of this type\footnote{Strictly speaking, the translation method described in Definition 55 is not applicable to the Language $A$ type domain descriptions described in Section 3 Proposition 2. This is because the “complete occurrence set” of i-propositions is infinite (see Definition 16). But the appropriate modification to Definition 55 is trivial, and does not necessitate a significant change to the proof of Proposition 10. An example ordering program $P(\Pi_\Delta, \leq_\Delta)$ is given in Appendix C together with the necessary changes to Definition 55.}.

**Proposition 10** 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 $D = \langle \gamma, \eta, \tau \rangle$ be a finite, initially-consistent, fluent-independent projection domain description in $\mathcal{E}$. Let $\tau_\Delta$ be a finite observation set. Then for any fluent literal $L$ of $\mathcal{E}$ and any $T \in \Pi$,

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

if and only if

\[
D, \tau_\Delta \models_i L \text{ holds-at } T
\]

**Proof** See Appendix B.5

As stated in Proposition 4, it is possible to verify the consistency of the class of projection domain descriptions described in Proposition 10 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 $IExplanation$. 

36
9 Conclusions and Further Work

Following the methodology of the Language $\mathcal{A}$ introduced in [13], 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 (n-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 [6] 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 [17]. Example candidates for comparison are the formalisms in [25], [27] and [30].

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 [8], [9], [15] and [16]. 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 $E$ could be increased in various ways. We have already briefly indicated how $E$ might be extended to partially deal with ramifications and qualifications (see Appendix A). Since $E$ 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 in [26] that a narrative based approach offers alternative ways to model non-deterministic effects of actions. Finally, we might extend the syntax and semantics of $E$ to deal with incomplete information about the order and timing of action occurrences, perhaps introducing temporal variables into the language in a manner similar to [5].

The utility and appeal of specialised declarative languages such as $A$ and $E$ 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 Deneker, Yannis Dimopoulos, David Evans, Alessandra Russo, Fariba Sadri, and Murray Shanahan for helpful discussions on the topic of this paper, and for careful reading of earlier drafts.

References


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 7 to 10, may be modified to accommodate extra types of propositions, without altering the basic principle of persistence captured in conditions (1)-(3) of Definition 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 not 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

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

This motivates a general definition for a new type of proposition:
Definition 56 [q-proposition] A \textit{q-proposition} in \( \mathcal{E} \) is an expression of the form

\[ A \text{ impossible-if } C \]

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 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 \( \gamma \), \( \eta \) and \( \tau \) are as before, and \( \kappa \) is a set of q-propositions\(^1\), the condition now becomes:

(a) For all t-propositions in \( \tau \) of the form \( \text{"F holds-at T"} \), \( H(F,T) = true \), and for all t-propositions of the form \( \text{"\neg F holds-at T'"} \), \( H(F,T') = false \).

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

A.2 Describing Indirect Effects of Actions

The following extension to \( \mathcal{E} \) is included to illustrate how Definition 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 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

\textit{Stuffy whenever} \{Closed, Blocked\}

Hence we define a new type of proposition as follows:

Definition 57 [r-proposition] An \textit{r-proposition} in \( \mathcal{E} \) is an expression of the form

\[ L \text{ whenever } C \]

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

\(^1\)Note that, if for every c-proposition \( \text{"A initiates F when C"} \) there is a q-proposition \( \text{"A impossible-if C \cup \{F\}"} \), 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 \textit{strong initiation} and \textit{strong termination} [28] can be incorporated into \( \mathcal{E} \) by addition of q-propositions where appropriate.
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 $(\gamma, \eta, \tau, \kappa, \rho)$, where $\gamma$, $\eta$, and $\tau$ are as before, $\kappa$ is a set of $q$-propositions and $\rho$ is a set of $r$-propositions.

**Definition 58** [Initiation/termination point for domains with $r$-propositions]

Let $H$ be an interpretation of $E$, let $D = (\gamma, \eta, \tau, \kappa, \rho)$ 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 $\eta$ of the form “$A$ happens-at $T$” and a $q$-proposition in $\gamma$ 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 $\rho$ 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) $H$ satisfies $C_2$ at $T$.

Condition (4) of Definition 10 is now:

(a) For all $t$-propositions in $\tau$ 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 $T$.

(c) For all $r$-propositions in $\rho$ 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 1**

**Proposition statement:** Let $D$ be an occurrence-sparse domain description written in a language $E = (\Pi, \preceq, \Delta, \Phi)$, 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 \( \eta \) of the form \( A \text{ happens-at} T \) 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 \( \eta \) of the form \( A \text{ happens-at} T' \), and (iii) there is no \( h \)-proposition in \( \eta \) of the form \( A \text{ happens-at} T'' \) such that \( T \preceq T' \prec T'' \). Let \( T_b \) be such a time point. By the inductive hypothesis, for all \( F \in \Phi \), \( H(F,T_b) = H'(F,T_b) \) and by construction there are no initiation or termination points \( T' \) (for any fluent) in \( H \) or \( H' \) such that \( T_b \preceq T' \prec T_2 \). There are three cases to consider:

**Case one:** There is not both an \( h \)-proposition in \( \eta \) of the form \( A \text{ happens-at} T_b \) and a \( c \)-proposition in \( \gamma \) either of the form \( A \text{ initiates} F' \text{ when} C \) or of the form \( A \text{ terminates} F' \text{ when} C \) such that \( H \) (and thus \( H' \)) satisfies \( C \) at \( T_b \). Hence by the first condition in the definition of a model, \( H(F',T_2) = H(F',T_b) = H'(F',T_b) = H'(F',T_2) \).

**Case two:** There is both an \( h \)-proposition in \( \eta \) of the form \( A \text{ happens-at} T_b \) and a \( c \)-proposition in \( \gamma \) of the form \( A \text{ initiates} F' \text{ when} C \) such that \( H \) (and thus \( H' \)) satisfies \( C \) at \( T_b \). 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 \( \eta \) of the form \( A \text{ happens-at} T_b \) and a \( c \)-proposition in \( \gamma \) of the form \( A \text{ terminates} F' \text{ when} C \) such that \( H \) (and thus \( H' \)) satisfies \( C \) at \( T_b \). 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 4

**Proposition statement:** Let \( D \) be an occurrence-sparse, non-converging, initially-consistent, fluent-independent projection domain description. Then \( D \) is consistent.

**Proof:** Let \( D = \langle \gamma,\eta,\tau,\iota \rangle \) be written in the projection language \( E = \langle \Pi,\preceq,\Delta,\Phi \rangle \), and let \( T_b \) be the null element of \( \Pi \).

Let \( M : \Phi \mapsto \{true,false\} \) be defined as follows. For each \( F \in \Phi \),
• \(M(F) = \text{true}\) if there is an i-proposition in \(\tau_i\) of the form “\text{initially } F”,

• \(M(F) = \text{false}\) 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 \(\tau_i\) of the form “\text{initially } \neg F”, \(M(F) = \text{false}\).

Since \(D\) is occurrence-sparse and non-converging, each time point \(T \in \Sigma\) 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 \(\eta\) of the form “\text{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 \(T \in \Sigma\) 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 \(\eta\) of the form “\text{A happens-at } T_n” and a c-proposition in \(\gamma\) either of the form “\text{A initiates } F\text{ when } C\)” or of the form “\text{A terminates } F\text{ when } C\)” such that \(H\) satisfies \(C\) at \(T_n\),

3. \(H(F, T) = \text{true}\) if \(n > 0\) and there is an \(A \in \Delta\) such that there is both an h-proposition in \(\eta\) of the form “\text{A happens-at } T_n” and a c-proposition in \(\gamma\) of the form “\text{A initiates } F\text{ when } C\)” such that \(H\) satisfies \(C\) at \(T_n\),

4. \(H(F, T) = \text{false}\) if \(n > 0\) and there is an \(A \in \Delta\) such that there is both an h-proposition in \(\eta\) of the form “\text{A happens-at } T_n” and a c-proposition in \(\gamma\) of the form “\text{A terminates } F\text{ 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 5**

**Proposition statement:** Let \(D = \langle \gamma, \eta, \tau_i \rangle\) be an occurrence-sparse projection domain description and let \(\tau_{ob}\) 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 \(\tau_{ob}\).

**Proof:** Let \(D\) be written in the projection language \(E = \langle \Sigma, \leq, \Delta, \Phi \rangle\), and let \(T_0\) be the null element of \(\Sigma\).

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

“Only if” half: Suppose $H$ is a model of $\langle \gamma, \eta, \pi \cup \tau_{ob} \rangle$. Let the set $\tau_H$ 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 $\langle \gamma, \eta, \pi \cup \tau_H \rangle$ and, by Proposition 1, $\langle \gamma, \eta, \pi \cup \tau_H \rangle \models p$ for each $p \in \pi_{ob}$, so that $\tau_H$ is an i-explanation for $\pi_{ob}$ in $D$. Hence $H$ is an i-model of $D$ with $\pi_{ob}$.

B.4 Proof of Proposition 9

Proposition statement:
Let $P(\Pi, \leq)$ 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

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

then

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

The proof of this proposition which is given below uses induction on the ‘length’ $\text{length}(a)$ of the SLDNF derivation $a$ of $HoldsAt(\lambda(L), T)$, where $\text{length}(a)$ is defined in Definition 59 below in terms of successful calls to $\text{HappensAt}$. It is defined so that each SLDNF sub-derivation of a $HoldsAt$ sub-goal within $a$ (which must have occurred within some call to $\text{Initiates}$, $\text{Terminates}$, $\text{PossiblyInitiates}$ or $\text{PossiblyTerminates}$) has ‘length’ less than the top-level derivation $a$.

Definition 59 [length(a)] Let $a$ be a successful SLDNF derivation of the goal $HoldsAt(\lambda(L), T)$ in $LP[D, P(\Pi, \leq)]$. $\text{length}(a)$ is defined inductively as follows:

$$\text{length}(a) = S + \sum_{\beta \in A_a} \text{size}(\beta)$$

where

- $S$ is the number of successful calls to $\text{HappensAt}$ at the top level of $a$, i.e. not called within a (negation-as-failure) finitely-failed subsidiary derivation of $a$. 

46
• $B_\alpha$ is the set of all finitely-failed subsidiary derivations from negative calls (to \textit{ClippedBetween} or \textit{AffectedBetween}) appearing at the top level of $\alpha$.

• size($\beta$) = 0 if there is no successful call to \textit{HappensAt} in any branch of $\beta$.

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

Proof of proposition. Let $\alpha$ be a successful SLDNF derivation of the goal \textit{HoldsAt}(\textit{\lambda}(L),T) in \textit{LP}[D,P(\Pi,\leq)]$. We will use induction on \text{length}(\alpha) to show that, given the fixed point $I^+$ as defined in Proposition 8, if \text{LP}[D,P(\Pi,\leq)] |-_{\text{SLDNF}} \text{HoldsAt}(\textit{\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 8.

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

(i) Success on (LP1a): Clearly, the success of \textit{Given}(\textit{\lambda}(L),T_1) for some $T_1 < T$ means that “\textit{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 $F$. Also, since the call $\beta$ to \textit{AffectedBetween} fails with size($\beta$) = 0, the unground sub-goal \textit{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 $F$, rule (1a) of $F$ 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 $F$ 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 $F$.

Inductive Step (\text{length}(\alpha) = n):
Suppose that the statement which we wish to prove is true for all SLDNF derivations $\alpha'$ of all \textit{HoldsAt} goals such that \text{length}(\alpha') < n. the query \textit{HoldsAt}(F,T) (respectively\textit{HoldsAt}(\text{Neg}(F'),T)) can succeed only on clauses (LP4), (LP1a),

\textsuperscript{11}We assume the convention that $\max(\emptyset) = 0$. 

47
(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 \( Given(L, t_1), t_1 \prec T \) 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 \( AffectedBetween(T_1, L, T_3) \) fails (so that the top level goal succeeds on clause (LP1a)), both of the calls \( PossiblyInitiates(A, F, T_3) \) and \( PossiblyTerminates(A, F, T_2) \) fail. Hence, if there is a \( c \)-proposition in \( D \) of the form “\( A \) initiates \( F \) when \( C \)” or “\( A \) terminates \( F \) when \( C \)” (respectively “\( A \) initiates \( F' \) when \( C \)” or “\( A \) terminates \( F' \) when \( C \)”), there is a fluent literal \( L_\varphi \in C \) such that \( not \ HoldsAt(L_\varphi, T_2) \), i.e. \( HoldAt(L_\varphi, T_2) \) succeeds, say with SLDNF derivation \( \alpha' \), where \( length(\alpha') < n \). By the inductive hypothesis, if \( L_\varphi = F_\varphi \) then \( I^+(F_\varphi, T_2) = false \), and if \( L_\varphi = \neg F_\varphi \) then \( I^+(F_\varphi, T_2) = true \). 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 \( A \), and each of the calls \( HoldAt(A, L_1, T_1), \ldots, HoldAt(A, L_k, T_1) \) 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 \( HoldAt(A, F, T) \). Hence by the inductive hypothesis, for each \( L_i = F_i, I^+(F_i, T_1) = true \), and for each \( L_i = \neg F_i, I^+(F_i, T_1) = false \). 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 \( ClippedBetween(T_1, F, T) \) 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 \( I^+(F', T) = false \).
Proof of Proposition 10

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

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

if and only if $H(F, T) = \text{true}$, and

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

if and only if $H(F, T) = \text{false}$.

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 $\eta$ of the form “$A \text{ happens-at }T_i$”. Proof is by induction of the length $n$ of this sequence.

Base Case:

Clearly, if $n = 0$ the queries $\text{HoldsAt}(M, F, T)$ and $\text{HoldsAt}(M, \text{Neg}(F), T)$ can succeed only on clause (EC7), and will succeed if and only if the queries $\text{Member}(F, M)$ and $\text{Member}(\text{Neg}(F), M)$ succeed respectively. By the second condition in Definition 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 $\eta$ of the form “$A \text{ happens-at }T_n$” and a $\gamma$-proposition in $\gamma$ of the form “$A \text{ initiates }F \text{ when }C$” such that $H$ satisfies $C$ at $T_n$. Hence by the third condition in the definition of a model, $H(F, T) = \text{true}$.

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

Case two: There is both an h-proposition in } \eta \text{ of the form "A } \textit{happens-at } T_n\text{" and a c-proposition in } \gamma \text{ of the form "A } \textit{terminates } F \text{ when } C\text{" such that } H \text{ satisfies } C \text{ at } T_n. \text{ Hence by the third condition in the definition of a model, } H(F, T) = false. \\
\text{In this case, by the inductive hypothesis and the program definition of } \textit{Initiates}, the query } \textit{HoldsAt}(M, \textit{Neg}(F), T) \text{ will succeed on clause (EC7) because the sub-goal} \\
\textit{ClippedBetween}(M, T_0, F, T) \text{ will succeed on clause (EC10) with the program variable } t_2 \text{ in the body of the clause bound to } T_n. \text{ The query} \\
\textit{HoldsAt}(M, F, T) \text{ will fail on clause (EC8) because by the inductive hypothesis and} \\
\text{fluent-independence of } D \text{ it will fail on the sub-goal } \textit{Initiates}(M, e, F, T_n) \text{ for all bindings of the variable } e \text{ provided by solutions to } \textit{Happens}(e, T_n). \text{ Hence in this case the lemma is also true.}

Case three: There is not both an h-proposition in } \eta \text{ of the form "A } \textit{happens-at } T_n\text{" and a c-proposition in } \gamma \text{ either of the form "A } \textit{initiates } F \text{ when } C\text{" or of the form "A } \textit{terminates } F \text{ when } C\text{" such that } H \text{ satisfies } C \text{ at } T_n. \text{ Hence by the second condition in the definition of a model, } H(F, T) = H(F, T_n). \\
\text{Clearly in this case, by the inductive hypothesis and by the program definitions of } \textit{Initiates } \text{ and } \textit{Terminates}, the queries } \textit{HoldsAt}(M, F, T) \text{ and} \\
\textit{HoldsAt}(M, \textit{Neg}(F), T) \text{ can succeed on the clauses (EC8) and (EC9) respectively only with the variable } t_1 \text{ in the body of each clause bound to some time-point } T_i < T_n. \text{ Moreover, by the same argument, for all } T' < T_n \text{, the queries} \\
\textit{ClippedBetween}(M, T', F, T) \text{ and } \textit{ClippedBetween}(M, T', \textit{Neg}(F), T) \text{ succeed if and only if the queries} \\
\textit{ClippedBetween}(M, T', F, T_n) \text{ and} \\
\textit{ClippedBetween}(M, T', \textit{Neg}(F), T_n) \text{ succeed respectively. Hence the queries} \\
\textit{HoldsAt}(M, F, T) \text{ and } \textit{HoldsAt}(M, \textit{Neg}(F), T) \text{ succeed if and only if the queries} \\
\textit{HoldsAt}(M, F, T_n) \text{ and } \textit{HoldsAt}(M, \textit{Neg}(F), T_n) \text{ succeed respectively. Hence in this case the lemma is also true.} \quad \Box

\textbf{Statement of Main Proposition:} \text{ Let } \mathcal{E} = (\Pi, \leq, \Delta, \Phi) \text{ be a fluent-finite, } \\
\text{non-converging projection language, let } P(\Pi, \leq) \text{ be an ordering program for } \mathcal{E}, \text{ and let } D = (\gamma, \eta, \tau) \text{ be a finite, initially-consistent, fluent-independent projection} \\
domain description in } \mathcal{E}. \text{ Let } \tau_{st} \text{ be a finite observation set. Then for any} \\
frequent literal } L \text{ of } \mathcal{E} \text{ and any } T \in \Pi, \\
EC[D, \tau_{st}, P(\Pi, \leq)] \vdash_{SLDF} \exists \textit{HoldsAt}(\lambda(L), T)
\text{ if and only if}
Proof:
Let an initial assignment of $D$ be defined as a function $M : \Phi \rightarrow \{true, false\}$ 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 1 and 4 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

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

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 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 $\tau_{ab}$. 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 $\tau_{ab}$. Hence the proposition is true by clause (EC1), Lemma 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 $A$ domains as Event Calculus style Prolog programs in the manner of Section 8, given the intermediate translations to $E$ 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 $\Delta$-sequences “$A_1, \ldots, A_n$” and “$A_1, \ldots, A_n, [A']$” respectively:

\[
t_1 \prec_\Delta t_2 \Leftrightarrow ListForm(t_2, l_2), Append([h|r], l_1, l_2), ListForm(t_1, l_1).
\]
In addition, the complete occurrence set of $\Delta$ is represented by a single clause which replaces all the domain-dependent ground \texttt{HappensAt} clauses of Definition 55:

\[
\text{HappensAt}(a, \text{Branch}(a, t)).
\]

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

\[
\text{HoldsAt}(m, l, t_3) \leftarrow \text{HappensAt}(a, t_1), \text{Initiates}(m, a, l, t_1), \\
\text{not ClippedBetween}(m, t_1, l, t_3).
\] (EC8')

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

\[
\text{ClippedBetween}(m, t_1, l, t_3) \leftarrow \text{HappensAt}(a, t_2), \\
\text{not ClippedBetween}(m, t_1, l, t_2).
\] (EC10')

\[
\text{ClippedBetween}(m, t_1, \text{Neg}(f), t_3) \leftarrow \text{HappensAt}(a, t_2), \\
\text{Initiates}(m, a, f, t_2).
\] (EC11')

This re-ordering avoids problems that would otherwise arise from calls to \texttt{HappensAt} with an ungrounded second argument.