PMON$^+$: A Fluent Logic for Action and Change
Formal Specification, Version 1.0

Patrick Doherty

Abstract

This report describes the current state of work with PMON, a logic for reasoning about action and change, and its extensions. PMON has been assessed correct for the $\mathcal{K} - IA$ class using Sandewall's Features and Fluents framework which provides tools for assessing the correctness of logics of action and change. A syntactic characterization of PMON has previously been provided in terms of a circumscription axiom which is shown to be reducible to a first-order formula. This report introduces a number of new extensions which are also reducible and deal with ramification. The report is intended to provide a formal specification for the PMON family of logics and the surface language $\mathcal{L}(SD)$ used to represent action scenario descriptions. It should be considered a working draft. The title of the report has a version number because both the languages and logics used are continually evolving. Since this document is intended as a formal specification which is used by our group as a reference for research and implementation, it is understandably brief as regards intuitions and applications of the languages and logics defined. We do provide a set of benchmarks and comments concerning these which can serve as a means of comparing this formalism with others. The set of benchmarks is not complete and is only intended to provide representative examples of the expressivity and use of this particular family of logics. We describe its features and limitations in other publications by our group which can normally be found at `http://www.ida.liu.se/labs/kplab/`.

Supported in part by the Swedish Research Council for the Engineering Sciences (TFR).

IDA Technical Report 1996
LiTH-IDA-R-96-33
ISSN-0281-4250

Department of Computer
and Information Science,
Linköping University,
S-581 83 Linköping, Sweden

PMON$^+$: A Fluent Logic for Action and Change Formal Specification, Version 1.0

Patrick Doherty*

Department of Computer and Information Science
Linköping University
S-58183 Linköping, Sweden
patdo@ida.liu.se

Abstract

This report describes the current state of work with PMON, a logic for reasoning about action and change, and its extensions. PMON has been assessed correct for the $\mathcal{K} - IA$ class using Sandewall's Features and Fluents framework which provides tools for assessing the correctness of logics of action and change. A syntactic characterization of PMON has previously been provided in terms of a circumscription axiom which is shown to be reducible to a first-order formula. This report introduces a number of new extensions which are also reducible and deal with ramification. The report is intended to provide a formal specification for the PMON family of logics and the surface language $\mathcal{L}(SD)$ used to represent action scenario descriptions. It should be considered a working draft. The title of the report has a version number because both the languages and logics used are continually evolving. Since this document is intended as a formal specification which is used by our group as a reference for research and implementation, it is understandably brief as regards intuitions and applications of the languages and logics defined. We do provide a set of benchmarks and comments concerning these which can serve as a means of comparing this formalism with others. The set of benchmarks is not complete and is only intended to provide representative examples of the expressivity and use of this particular family of logics. We describe its features and limitations in other publications by our group which can normally be found at `http://www.ida.liu.se/labs/kplab/`.

*Supported in part by the Swedish Research Council for Engineering Sciences (TFR).

Contents

1 Introduction

The following report is intended to provide a formal specification for the PMON family of logics and the surface language $\mathcal{L}(SD)$ used to represent action scenario descriptions. The title of the report has a version number because both the languages and logics used are continually evolving. Since this document is intended as a formal specification which is used by our group as a reference for research and implementation, it is understandably brief as regards intuitions and applications of the languages and logics defined. We do provide a set of benchmarks and comments concerning these which can serve as a means of comparing this formalism with others. The set of benchmarks is not complete and is only intended to provide representative examples of the expressivity and use of this particular family of logics. We describe its features and limitations in other publications by our group which can normally be found at `http://www.ida.liu.se/labs/kplab/`.

The Features and Fluents project began in 1987 and was initially based on a number of technical reports by Sandewall (e.g. [22, 21, 24, 25]) and published articles (e.g. [23, 26, 28]) that resulted in a book by Sandewall [27] which covers part of the research done during this period. Two concepts of fundamental importance, the use of filtering and occlusion, were introduced at an early stage in the project [21, 23], and in the current research with PMON are proving to be quite versatile

in dealing with many of the problems which arise when reasoning about action and change. For details about the relation between filtering and occlusion and other approaches, see Sandewall [27].

Doherty [4, 3, 5, 6] , took an approach somewhat different from Sandewall and generated syntactic characterizations of many of the definitions of preferential entailment introduced in Features and Fluents by translating scenario descriptions into a standard sorted first-order logic with circumscription axioms in an attempt to provide a basis for implementing some of these logics. It turns out that PMON is also well-behaved in this respect; any scenario description in PMON is reducible to a first-order theory. In the current report, we continue the development of PMON and show that it continues to have a number of nice mathematical properties even when extended for ramification [8]. In related work, Karlsson [10, 11, 13, 12] uses PMON as a basis for formal specification of plans.

2 The Base Logic FL

There are a number of different possibilities for choosing a base logic in which to compile scenario descriptions which are described using the scenario description language $\mathcal{L}(SD)$. Sandewall uses a specialized logic called *discrete feature logic* with a number of interesting non-standard features which can potentially aid in developing efficient implementations for the various action classes. In this section, we introduce an alternative, which we call *fluent logic* and denote FL. FL will be used as a basis for formalizing reasoning about action and change. The language for FL, denoted $\mathcal{L}(FL)$, is a many-sorted first-order language. All scenario descriptions described in $\mathcal{L}(SD)$ have a modular translation into $\mathcal{L}(FL)$.

2.1 Syntax

The language $\mathcal{L}(FL)$, is a many-sorted first-order language with equality. We use the standard connectives $\neg$ (negation), $\wedge$ (conjunction), $\rightarrow$ (implication), $\equiv$ (equivalence), $\vee$ (disjunction), and the standard quantifiers $\forall$ and $\exists$. In addition, scoping will be indicated by standard use of parentheses or dot notation.

We associate a many-sorted signature in $\mathcal{L}(FL)$ for each vocabulary and action similarity type in $\mathcal{L}(SD)$ as follows.

Let $\nu = \langle \sigma, \mathcal{O}, \mathcal{V}, \mathcal{D} \rangle$ be a vocabulary in $\mathcal{L}(SD)$, where

$$\mathcal{O} = \langle \mathcal{O}_1, \ldots, \mathcal{O}_l \rangle,$$

$$\mathcal{V} = \langle \mathcal{V}_1, \ldots, \mathcal{V}_p \rangle,$$

$$\sigma = \{ f_1^{i_1} : \mathcal{D}_1, f_2^{i_2} : \mathcal{D}_2, \ldots f_n^{i_n} : \mathcal{D}_n \},$$

and

$$\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n \rangle, \text{ where } \mathcal{D}_i \in \mathcal{O} \text{ or } \mathcal{V}.$$

In addition, let

$$\pi = \{ A_1^{i_1}, A_2^{i_2}, \ldots A_m^{i_m} \},$$

be an action similarity type for $\mathcal{L}(SD)$. The many sorted-signature $\mathcal{S} = (S, \Sigma)$, is a pair where S is a finite set of sort symbols

$$S = \{ \mathcal{F}, \mathcal{A}, \mathcal{T} \} \cup S_o \cup S_v \,,$$

$$S_o = \{ O_1, \ldots, O_l \} \text{ and } S_v = \{ V_1, \ldots, V_p \} \text{ and } S_d = S_o \cup S_v \,.$$

and Σ is a denumerable set of sort strings describing the type for the function and relation symbols in $\mathcal{L}(FL)$ associated with the feature symbols in σ, the action symbols in π, the function and

relation symbols associated with the temporal sort $\mathcal{T}$, the domain independent relation symbols, and the constant symbols for each of the sorts. Σ contains the following domain dependent strings :

- For each feature symbol $f_k^{i_k}$ in σ with a non-boolean value domain sort $dom_\nu(f_k^{i_k}) = \mathcal{D}_k$, $\mathcal{L}(FL)$ contains a function symbol f_k^j of arity $j = i_k + 1$ and Σ contains the string $f_k^j : s_1 \times \ldots \times s_j \rightarrow \mathcal{F}$, where $s_i \in S_o$, $1 \leq i \leq j-1$, and $s_j \in S_d$.
- For each feature symbol $f_k^{i_k}$ in σ with a boolean value domain sort $dom_\nu(f_k^{i_k}) = \mathcal{D}_k$, $\mathcal{L}(FL)$ contains a function symbol f_k^j of arity $j = i_k$ and Σ contains the string $f_k^j : s_1 \times \ldots \times s_j \rightarrow \mathcal{F}$, where $s_i \in S_o$, $1 \leq i \leq j$.
- For each action symbol $A_k^{i_k}$ in π, $\mathcal{L}(FL)$ contains a function symbol A_k^j of arity $j = i_k$ and Σ contains the string $A_k^j : s_1 \times \ldots \times s_j \rightarrow \mathcal{A}$, where $s_i \in S_d$, $1 \leq i \leq j$.
- For each constant c in $\mathcal{L}(FL)$ of sort O_i, Σ contains a string $c : O_i$. For each constant c in $\mathcal{L}(FL)$ of sort V_i, Σ contains a string $c : V_i$.

In addition, Σ contains the following domain independent strings:

- $Holds : \mathcal{T} \times \mathcal{F}$,
- $Occlude : \mathcal{T} \times \mathcal{F}$,
- $Observe : \mathcal{T} \times \mathcal{F}$,
- $Occurs : \mathcal{T} \times \mathcal{T} \times \mathcal{A}$,
- $<: \mathcal{T} \times \mathcal{T}$, $\leq: \mathcal{T} \times \mathcal{T}$,
- $+ : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$, $- : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$.

2.1.1 The Object Sorts S_o

For each sort O_i in S_o:

- We use the letter o^i, possibly subscripted for variables of sort O_i.
- For each object name n_j in $\mathcal{O}_i$, we associate an *object name constant* n_j^i of sort O_i of the same name in $\mathcal{L}(FL)$.
- We use the letter o^i, possibly subscripted for object constants of sort O_i.

Only constants and variables of sorts O_i are object terms. A unique names assumption will be applied to the object name constants, while the object constants will be allowed to vary among object names.[1]

2.1.2 The Value Sorts S_v

For each sort V_i in S_v:

- We use the letter v^i, possibly subscripted for variables of sort V_i.
- For each value name ν_j in $\mathcal{V}_i$, we associate an *value name constant* v_j^i of sort V_i of the same name in $\mathcal{L}(FL)$.
- We use the letter u^i, possibly subscripted for *value constants* of sort V_i.

[1] For convenience, we may sometimes violate naming conventions for the different sorts and use other letters when there is no danger of ambiguity.

Only constants and variables of sorts V_i are value terms. A unique names assumption will be applied to the value name constants, while the value constants will be allowed to vary among value names.

2.1.3 The Fluent Sort $\mathcal{F}$

- We use the letters f or z, possibly subscripted for variables of sort $\mathcal{F}$.
- For each feature symbol $f_k^{i_k}$ in σ, we associate a function symbol f_k in $\mathcal{L}(FL)$ of the same name and having the sort described in Σ.

Fluent terms are constructed in the usual manner, but note that embedded terms $f(f(-))$ are not legal terms in the language. A unique names assumption will be applied to the fluents terms in sort $\mathcal{F}$.

2.1.4 The Action Sort $\mathcal{A}$

- We use the letter a, possibly subscripted for variables of sort $\mathcal{A}$.
- For each action symbol $A_k^{i_k}$ in σ, we associate a function symbol A_k in $\mathcal{L}(FL)$ of the same name and having the sort described in Σ.

Action terms are constructed in the usual manner, but note that embedded terms $A(A(-))$ are not legal terms in the language. A unique names assumption will be applied to the actions terms in sort $\mathcal{A}$.

2.1.5 The Temporal Sort $\mathcal{T}$

- We use the letter t, possibly subscripted for variables of sort $\mathcal{T}$.
- We use the letter t, possibly subscripted, and the numerals (0, 1, 2, ...) for temporal constants of sort $\mathcal{T}$.

The intended interpretation of $\mathcal{T}$ is as a linear discrete time line where $\mathcal{T}$ is considered isomorphic to the natural numbers. Although we assume an interpreted theory, using the standard interpretation for the natural numbers, it turns out that when reasoning with $\mathcal{K}-IA$ or its current extensions, we use no more than Presburger arithmetic. In fact, we use a sub-theory of Presburger arithmetic, the first-order theory of point constraints over integers [15].

The relation symbols $<$ and $\leq$ are interpreted as the usual "less than" and "less than or equal to" relations on natural numbers. The function symbols $+$, and $-$ are interpreted as the usual "plus" and "minus" functions on natural numbers.

In what follows, $\rho < \rho' < \rho''$, $\rho \leq \rho' < \rho''$, $\rho < \rho' \leq \rho''$ and $\rho \leq \rho' \leq \rho''$, stand for $\rho < \rho' \wedge \rho' < \rho''$, $\rho \leq \rho' \wedge \rho' < \rho''$, $\rho < \rho' \wedge \rho' \leq \rho''$ and $\rho \leq \rho' \wedge \rho' \leq \rho''$, respectively, where ρ,ρ' and ρ'' are terms of sort $\mathcal{T}$.

3 Foundational Axioms in FL

In this section we describe a number of domain independent axioms that are always assumed to be part of any translation of action scenarios into $\mathcal{L}(FL)$.

In the following, we will use the meta variables

- $d_1, \ldots$, possibly superscripted, to denote object or value variables of sorts inS_o, or S_v.
- $\mathsf{d}_1, \ldots$, possibly superscripted, to denote object or value constants, or name constants of sorts S_o or S_v.

3.1 Axioms for the Value Sorts S_v

For each fluent $f_k^{i_k}$ with value domain $D_k \in S_d$, the following *unique values axioms* are assumed which state that at a specific time-point t, a fluent $f_k^{i_k}$ can only be assigned one value from its value domain for a particular set of object arguments $o_1, \ldots, o_{i_k} - 1$:

$$\begin{aligned} &\forall \bar{o}, t, d_1^k, d_2^k . (d_1^k \neq d_2^k \rightarrow \\ &\quad \neg(Holds(t, f_k^{i_k}(o_1, \ldots, o_{i_k-1}, d_1^k)) \wedge Holds(t, f_k^{i_k}(o_1, \ldots, o_{i_k-1}, d_2^k)))), \end{aligned} \tag{1}$$

where $\bar{o} = o_1, \ldots, o_{i_k-1}$. We will denote the set of unique values axioms by Γ_{uva}.

For each fluent $f_k^{i_k}$ with value domain $D_k \in S_d$, the following *value existence axioms* are assumed which state that at a specific time-point t, a fluent $f_k^{i_k}$ must be assigned at least one value from its value domain for a particular set of object arguments $o_1, \ldots, o_{i_k}$:

$$\forall \bar{o} \forall t \exists d_1^k . Holds(t, f_k^{i_k}(o_1, \ldots, o_{i_k-1}, d_1^k)) \tag{2}$$

where $\bar{o} = o_1, \ldots, o_{i_k-1}$. We will denote the set of value existence axioms by Γ_{vea}.

For theories with finite value domains, the following *unique name axioms* are assumed for each of the value sorts $V_j \in S_v$:

$$\bigwedge_{1 \leq i < k \leq |\mathcal{V}_j|} \mathsf{v}_i^j \neq \mathsf{v}_k^j . \tag{3}$$

Note that this axiom applies to *value name constants* and not *value constants*, which may be bound to the interpretation of any value name constant. We will denote the set of unique names axioms for value domains by Γ_{una_v}.

For theories with finite value domains, the following *domain closure axioms* are assumed for each of the value sorts $V_j \in S_v$:

$$\forall v^j . \bigvee_{i=1}^{|\mathcal{V}_j|} v^j = \mathsf{v}_i^j . \tag{4}$$

Similar axioms for value domains D_i of type S_o will be defined in Section 3.2 We will denote the set of domain closure axioms for value domains by Γ_{dca_v}.

3.2 Axioms for the Objects Sorts S_o

For theories with finite object domains, the following *unique name axioms* are assumed for each of the object sorts $O_j \in S_o$:

$$\bigwedge_{1 \leq i < k \leq |\mathcal{O}_j|} \mathsf{n}_i^j \neq \mathsf{n}_k^j . \tag{5}$$

We will denote the set of unique names axioms for object domains by Γ_{una_o}.

For theories with finite object domains, the following *domain closure axioms* are assumed for each of the object sorts $O_j \in S_o$:

$$\forall o^j \,.\, \bigvee_{i=1}^{|\mathcal{O}_j|} o^j = \mathsf{n}_i^j. \tag{6}$$

We will denote the set of domain closure axioms for object domains by Γ_{dca_o}.

3.3 Axioms for the Fluent Sort $\mathcal{F}$

For theories with finite fluent domains, the following unique names axioms are assumed:

$$\bigwedge_{1 \le j < k \le n} \forall \bar{o}, d_{i_j}, d_{i_k} . f_j^{i_j}(o_1, \ldots, o_{i_j - 1}, d_{i_j}) \neq f_k^{i_k}(o'_1, \ldots, o'_{i_k - 1}, d'_{i_k}), \tag{7}$$

where d_{i_j}, d'_{i_k} are either object or value variables.

$$\forall \bar{o}, d_{i_k}, d'_{i_k} . (f_k^{i_k}(o_1, \ldots, o_{i_k - 1}, d_{i_k}) = f_k^{i_k}(o'_1, \ldots, o'_{i_k - 1}, d'_{i_k})) \rightarrow (\bigwedge_{j=1}^{i_k - 1} o_j = o'_j) \wedge d_{i_k} = d'_{i_k}. \tag{8}$$

We will assume that similar axioms exist for boolean value domains and will denote the set of unique names axioms for the fluent domain by Γ_{una_f}.

3.4 Axioms for the Action Sort $\mathcal{A}$

For theories with finite action domains, the following unique names axioms are assumed:

$$\bigwedge_{1 \le j < k \le m} \forall \bar{d} . A_j^{i_j}(d_1, \ldots, d_{i_j}) \neq A_k^{i_k}(d'_1, \ldots, d'_{i_k}). \tag{9}$$

$$\forall \bar{d} . (A_k^{i_k}(d_1, \ldots, d_{i_k}) = A_k^{i_k}(d'_1, \ldots, d'_{i_k})) \rightarrow (\bigwedge_{j=1}^{i_k} d_j = d'_j), \tag{10}$$

where $\bar{d} = d_1, \ldots, d_{i_k}, d'_1, \ldots, d'_{i_k}$ are either object or value variables. We will denote the set of unique names axioms for the action domain by Γ_{una_a}.

3.5 Some Useful Notation

The following notation will be useful when describing circumscription policies in a later section.

$$\Gamma_{UNA} = \Gamma_{una_v} \cup \Gamma_{una_o} \cup \Gamma_{una_f} \cup \Gamma_{una_a}. \tag{11}$$

$$\Gamma_{DCA} = \Gamma_{dca_v} \cup \Gamma_{dca_o}. \tag{12}$$

The following collection of Foundational Axioms will be used in the circumscription policies.

$$\Gamma_{FA} = \Gamma_{UNA} \cup \Gamma_{DCA} \cup \Gamma_{uva} \cup \Gamma_{vea}. \tag{13}$$

4 Reducing Scenario Descriptions to $\mathcal{L}(FL)$

Given a scenario description Υ, consisting of statements in the surface language $\mathcal{L}(SD)$, these statements can be translated into formulas in the language $\mathcal{L}(FL)$ via a two-step process. In the first step:

- Action schemas in Υ are instantiated with action occurrence statements, resulting in what are called *schedule statements*. The resulting schedule statements replace the action schemas.
- Observation statements are translated into *fixed observation statements*.

The result is an *expanded (action) scenario description* Υ', consisting of schedule, fixed observation, causal constraint, acausal constraint and dynamic fluent statements. Each of these are statements in $\mathcal{L}(SD)$. In the second step, translation operators mapping statements from $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$ are used to translate statements in Υ' into well-formed formulas in $\mathcal{L}(FL)$. The details are provided in the appendices.

A The Surface Language for Scenario Descriptions

In this appendix, we will define each of the different types of statements in $\mathcal{L}(SD)$ together with the translation operators which map statements from $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$. In addition, we provide a number of Lemmas and Theorems related to the reduction of circumscriptive theories to the first-order case.

In Section 2, we defined the relationship between symbols in $\mathcal{L}(SD)$ and those in $\mathcal{L}(FL)$. In the following, we will additionally assume that there is a straightforward and unambiguous mapping from all logical connectives, quantifiers, delimiters, and variable symbols used in $\mathcal{L}(SD)$ to their correlates in $\mathcal{L}(FL)$. The translation operators $Tran_{EOE}$, $Tran_{EVE}$, and $Tran_{ETE}$ define part of this mapping, but we do not always explicitly apply these operators when the application is clear from the context. This will avoid a certain amount of notational overhead.

The legal syntax for representing scenario descriptions is defined in terms of a surface language $\mathcal{L}(SD)$ consisting of:

- *action occurrence statements*,
- *action law schemas*,
- *observation statements* and *fixed observation statements*,
- *schedule statements*,
- *causal constraint statements*,
- *acausal constraint statements*, and
- *dynamic fluent statements*.

All statements in $\mathcal{L}(SD)$, with the exception of occurrence statements, are constructed using *logical* and *reassignment formulas* in $\mathcal{L}(SD)$. Both logical and reassignment formulas, together with occurrence statements, can be translated to formulas in the base logic $\mathcal{L}(FL)$.

Definition A.1 (Object Domain for $\mathcal{L}(SD)$)
An *object domain* $\mathcal{O} = \langle \mathcal{O}_1, \ldots, \mathcal{O}_n \rangle$ is a finite tuple of object sorts. Each object sort is defined as a non-empty finite set of *object names*.

Definition A.2 (Value Domain for $\mathcal{L}(SD)$)
A *value domain* $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n \rangle$ is a finite tuple of value or object sorts from $\mathcal{O}, \mathcal{V}$. Each object or value sort is defined as a non-empty finite set of *object* or *value names*.

Definition A.3 (Similarity Type for $\mathcal{L}(SD)$)
A *similarity type* σ for $\mathcal{L}(SD)$ is a mapping

$$\{f_1^{i_1} : \mathcal{D}_1, f_2^{i_2} : \mathcal{D}_2, \ldots f_n^{i_n} : \mathcal{D}_n\}$$

where $i_k \geq 0$ for each k and each $f_k^{i_k}$ is a feature symbol with i_k arguments. An object domain sort $\mathcal{O}_j$ from the object domain $\mathcal{O}$ is associated with each of the i_k arguments of $f_k^{i_k}$. Each $\mathcal{D}_k$ is a non-empty finite set of feature values for f_k of value domain sorts $\mathcal{O}_i$ or $\mathcal{V}_i$. $Dom_\nu(f_k)$ will be used to denote the value domain sort for the feature f_k. $Dom_\nu(f_k^j)$ will be used to denote the object domain sort $\mathcal{O}_i$ in $\mathcal{O}$ associated with the jth argument of the feature f_k.

A *vocabulary* $\nu = \langle \sigma, \mathcal{O}, \mathcal{V}, \mathcal{D} \rangle$ is a tuple consisting of a similarity type, an object domain, value sorts, and a value domain.

The *value domain for* ν, $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n \rangle$, is the finite tuple of value sorts in $\mathcal{O}$ or $\mathcal{V}$ associated with the feature symbols in σ.

Remark A.1 *The surface language will be set up so that the definitions can be extended as needed when modeling additional concepts associated with action and change. For example, currently, both the object and value domains are defined as being finite. At a later date, this restriction may be relaxed.*

A.1 Logical Formulas in $\mathcal{L}(SD)$

Let $\nu = \langle \sigma, \mathcal{O}, \mathcal{V}, \mathcal{D} \rangle$ be a vocabulary.

Definition A.4 (Elementary Object Expression (EOE))
An *elementary object expression* is any one of the following:

1. An object constant symbol.
2. An object name symbol.
3. An object variable symbol.

Definition A.5 (Elementary Value Expression (EVE))
An *elementary value expression* is any one of the following:

1. A value constant symbol.
2. A value name symbol.
3. A value variable symbol.

Definition A.6 (Elementary Feature Expression (EFE))
An *elementary feature expression* is any one of the following:

1. A feature symbol f_k^0 from the vocabulary ν.
2. $f_k^{i_k}(\omega_1, \ldots, \omega_{i_k})$ where $f_k^{i_k}$ is from the vocabulary ν and $\omega_1, \ldots, \omega_{i_k}$ are elementary object expressions.

Definition A.7 (Elementary Fluent Formula (EFlF))
An *elementary fluent formula* has the form $f \hat{=} \mathcal{X}$, where f is an elementary feature expression and $\mathcal{X} \subseteq Dom_\nu(f)$.

Definition A.8 (Fluent Formula (FlF))
The set of *fluent formulas* FlF is defined as follows :

1. If ϕ is an elementary fluent formula then ϕ is in FlF.

2. If o is an object variable and ϕ is in FlF, then $\forall o.\phi$ and $\exists o.\phi$ are in FlF.

3. If ϕ and ψ are in FlF then $\neg\phi$ is in FlF and any boolean combination of ϕ and ψ are in FlF.

4. Nothing else is in FlF.

Definition A.9 (Elementary Time-point Expression (ETE))
An *elementary time-point expression* is any one of the following:

1. A temporal constant in $\mathcal{L}(FL)$.

2. A temporal variable in $\mathcal{L}(FL)$.

3. Any temporal term constructed from temporal functions in $\mathcal{L}(FL)$.

Remark A.2 *Note that as regards ETE's, it is generally assumed that any term in $\mathcal{L}(FL)$ can be used in $\mathcal{L}(SD)$. We set up the correspondence in this manner because it should be relatively straightforward to change temporal structures when using Fluent Logic.*

Definition A.10 (Elementary Fixed Fluent Formula (EFFF))
Let τ denote an elementary time-point expression, and γ denote an elementary fluent formula. An *elementary fixed fluent formula* has the following form:

1. $[\tau]\gamma$.

Definition A.11 (Elementary Fixed Formula (EFF))
Let τ and τ' denote elementary time-point expressions, ω and ω' denote elementary object expressions. An *elementary fixed formula* is any one of the following:

1. An elementary fixed fluent formula.

2. $\tau = \tau'$.

3. $\omega = \omega'$.

4. $\tau \otimes \tau'$, where $\otimes$ is any temporal relation defined in $\mathcal{L}(FL)$.

Definition A.12 (Fixed Fluent Formula (FFF))
Let τ denote an elementary time-point expression, and γ denote a fluent formula. A *fixed fluent formula* has the following form:

1. $[\tau]\gamma$.

Definition A.13 (Logic Formula (LF))
The set of *logic formulas* LF is defined as follows :

1. If ϕ is an elementary fixed formula then ϕ is in LF.

2. If t and o are temporal and object variables, respectively, and ϕ is in LF, then $\forall t.\phi, \exists t.\phi, \forall o.\phi, \exists o.\phi$ are in LF.

3. If ϕ and ψ are in LF then $\neg\phi$ is in LF and any boolean combination of ϕ and ψ is in LF.

4. Nothing else is in LF.

Definition A.14 (Restricted Logic Formula (RLF))
The set of *restricted logic formulas* RLF is defined as follows :

1. If ϕ is a fixed fluent formula then ϕ is in RLF.

2. If t and o are temporal and object variables, respectively, and ϕ is in RLF, then $\forall t.\phi, \exists t.\phi, \forall o.\phi, \exists o.\phi$ are in RLF.

3. If ϕ and ψ are in RLF then $\neg\phi$ is in RLF and any boolean combination of ϕ and ψ is in RLF.

4. Nothing else is in RLF.

Definition A.15 (Time-Point Formula (TPF))
A *time-point formula* is a logic formula where the elementary fixed formulas from which it is constructed only contain types 2 and 4 in Definition A.11.

Definition A.16 (Object Formula (OF))
An *object formula* is a logic formula where the elementary fixed formulas from which it is constructed only contain type 3 in Definition A.11.

Any fixed fluent formula can be represented as a composition of elementary fixed fluent formulas (EFFF) using the logical connectives and quantification over objects in the usual fashion.

Definition A.17 (Fixed Fluent Formula Abbreviations)
Let τ be an ETE, o an object variable and γ, γ', fluent formulas. The following reduction rules can be used to translate any FFF into a composition of elementary fixed fluent formulas:[2]

$[\tau]\neg(\gamma \wedge \gamma') \stackrel{\text{def}}{=} [\tau]\neg\gamma \vee \neg\gamma'$.

$[\tau]\neg(\gamma \vee \gamma') \stackrel{\text{def}}{=} [\tau]\neg\gamma \wedge \neg\gamma'$.

$[\tau]\gamma \wedge \gamma' \stackrel{\text{def}}{=} [\tau]\gamma \wedge [\tau]\gamma'$.

$[\tau]\gamma \vee \gamma' \stackrel{\text{def}}{=} [\tau]\gamma \vee [\tau]\gamma'$.

$[\tau]\gamma \rightarrow \gamma' \stackrel{\text{def}}{=} [\tau]\neg\gamma \vee \gamma'$.

$[\tau]\gamma \equiv \gamma' \stackrel{\text{def}}{=} [\tau]\gamma \equiv [\tau]\gamma'$.

$[\tau]\neg\neg\gamma \stackrel{\text{def}}{=} [\tau]\gamma$.

$[\tau]\neg\gamma \stackrel{\text{def}}{=} \neg[\tau]\gamma$.

$[\tau]\forall o.\gamma \stackrel{\text{def}}{=} \forall o.([\tau]\gamma)$

$[\tau]\exists o.\gamma \stackrel{\text{def}}{=} \exists o.([\tau]\gamma)$

Definition A.18 (Interval Fixed Fluent Formula (IFFF))
Let τ, τ' denote elementary time-point expressions, and γ denote a fluent formula. An *interval fixed fluent formula* has any of the following forms:

1. $[\tau,\tau']\gamma$, $(\tau,\tau']\gamma$, $[\tau,\tau')\gamma$, $(\tau,\tau')\gamma$.
2. $(-\infty,\tau']\gamma$, $(-\infty,\tau')\gamma$, $[\tau,\infty)\gamma$, $(\tau,\infty)\gamma$, $(-\infty,\infty)\gamma$.

Definition A.19 (Abbreviations for IFFF's)

Let τ, τ' be ETEs, and γ be a fluent formula. The following reduction rules can be used to translate any IFFF into a logic formula

$[\tau,\tau']\gamma \stackrel{\text{def}}{=} \forall t.\ \tau \leq t \leq \tau' \rightarrow [t]\gamma$.

$[\tau,\tau')\gamma \stackrel{\text{def}}{=} \forall t.\ \tau \leq t < \tau' \rightarrow [t]\gamma$.

$(\tau,\tau']\gamma \stackrel{\text{def}}{=} \forall t.\ \tau < t \leq \tau' \rightarrow [t]\gamma$.

$(\tau,\tau')\gamma \stackrel{\text{def}}{=} \forall t.\ \tau < t < \tau' \rightarrow [t]\gamma$.

[2] Note that the standard application of these rules is assumed, where all negations are first driven inward before applying the double negation elimination and then any single negation left can be driven to the left outside the scope of the [t] notation.

$(-\infty, \tau']\gamma \stackrel{\text{def}}{=} \forall t.\, t \leq \tau' \rightarrow [t]\gamma.$

$(-\infty, \tau')\gamma \stackrel{\text{def}}{=} \forall t.\, t < \tau' \rightarrow [t]\gamma.$

$[\tau, \infty)\gamma \stackrel{\text{def}}{=} \forall t.\, \tau \leq t \rightarrow [t]\gamma.$

$(\tau, \infty)\gamma \stackrel{\text{def}}{=} \forall t.\, \tau < t \rightarrow [t]\gamma.$

$(-\infty, \infty)\gamma \stackrel{\text{def}}{=} \forall t.\, [t]\gamma.$

A.1.1 Reducing Logic Formulas in $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$

In this section, we will provide an algorithm which takes as input a formula in $\mathcal{L}(SD)$ and returns a wff in $\mathcal{L}(FL)$. We will assume that the relation between vocabularies described in Section 2 holds.

Definition A.20 ($Tran_{EOE}$)
Let ω be an elementary object expression (EOE).

1. If ω is an object constant symbol or object name symbol of sort $\mathcal{O}_i$ then $Tran_{EOE}(\omega) = c_i$, where c_i is a constant symbol in $\mathcal{L}(FL)$ of sort O_i, which is the correlate to $\mathcal{O}_i$ and assumed to exist in $\mathcal{L}(FL)$'s similarity type.
2. If ω is a variable symbol of sort $\mathcal{O}_i$ then $Tran_{EOE}(\omega) = o^i$, where o^i is a variable symbol in $\mathcal{L}(FL)$ of sort O_i.

Definition A.21 ($Tran_{EVE}$)
Let ω be an elementary value expression (EVE).

1. If ω is a value constant symbol or value name symbol of sort $\mathcal{V}_i$ then $Tran_{EOE}(\omega) = c_i$, where c_i is a constant symbol in $\mathcal{L}(FL)$ of sort V_i, which is the correlate to $\mathcal{V}_i$ and assumed to exist in $\mathcal{L}(FL)$'s signature.
2. If ω is a variable symbol of sort $\mathcal{V}_i$ then $Tran_{EVE}(\omega) = v^i$, where v^i is a variable symbol in $\mathcal{L}(FL)$ of sort V_i.

Definition A.22 ($Tran_{E(OV)E}$)
Let ω be an EOE or EVE. We will often use the translation operator $Tran_{E(OV)E}$ which is defined as

$$Tran_{E(OV)E}(\omega) = \begin{cases} Tran_{EOE}(\omega) & \text{if } \omega \text{ is of sort } \mathcal{O} \\ Tran_{EVE}(\omega) & \text{if } \omega \text{ is of sort } \mathcal{V} \end{cases}$$

Definition A.23 ($Tran_{ETE}$)
It is assumed that there is a straightforward mapping from temporal constants, variables, and expressions used in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$.

Definition A.24 ($Tran_{EFFF}$)
Let ϕ be an elementary fixed fluent formula $[\tau]f_k(\omega_1, \ldots, \omega_{i_k}) \hat{=} \mathcal{X}$, where $\mathcal{X} = \{\nu_1, \ldots, \nu_n\}$, $Dom_\nu(f_k)$ is non-Boolean, and f_k has arity i_k and sort

$$Dom_\nu(f_k^1) \times \ldots \times Dom_\nu(f_k^{i_k}) \rightarrow Dom_\nu(f_k).$$

$$Tran_{EFFF}(\phi) = \bigvee_{j=1}^{n} Holds(\rho, f_k(\pi_1, \ldots, \pi_{i_k}, \mathsf{d}_j^k)),$$

where $\rho = Tran_{ETE}(\tau)$, each $\mathsf{d}_j^k = Tran_{E(OV)E}(\nu_j)$ is a name constant of sort D_k associated with each name in $Dom_\nu(f_k)$, each $\pi_l = Tran_{EOE}(\omega_l)$ is either a constant or variable of the appropriate

sorts O_l associated with sorts $Dom_\nu(f_k^l)$, and f_k is the function symbol in $\mathcal{L}(FL)$ of arity $i_k + 1$ associated with f_k in $\mathcal{L}(SD)$.

If $Dom_\nu(f_k)$ is Boolean and $\mathcal{X} = \{T, F\}$ then

$$\begin{aligned} Tran_{EFFF}(\phi) = \\ Holds(\rho, f_k(\pi_1, \ldots, \pi_{i_k})) \\ \vee \neg Holds(\rho, f_k(\pi_1, \ldots, \pi_{i_k})). \end{aligned}$$

If $Dom_\nu(f_k)$ is Boolean and $\mathcal{X} = \{T\}$ then

$$Tran_{EFFF}(\phi) = Holds(\rho, f_k(\pi_1, \ldots, \pi_{i_k})).$$

If $Dom_\nu(f_k)$ is Boolean and $\mathcal{X} = \{F\}$ then

$$Tran_{EFFF}(\phi) = \neg Holds(\rho, f_k(\pi_1, \ldots, \pi_{i_k})).$$

Lemma A.1
Let ψ be an arbitrary logic formula in the surface language for scenario descriptions. There is an algorithm that transforms ψ into a wff in $\mathcal{L}(FL)$.

Proof

Since a logic formula is composed of elementary fixed formulas (EFF) using logical connectives and quantifiers in the usual fashion, and we assume the vocabulary of $\mathcal{L}(FL)$ contains the necessary sorts, and logical and non-logical symbols, it suffices to show that elementary fixed formulas can be translated. Let ψ be a logical formula in $\mathcal{L}(SD)$ and ϕ be an arbitrary elementary fixed formula in ψ.according to Definition A.11, there are four cases:

1. If $\phi \equiv [\tau]\gamma$ is an elementary fixed fluent formula then replace ϕ in ψ with $Tran_{EFFF}(\phi)$.
2. If $\phi \equiv \tau = \tau'$ then replace ϕ in ψ with $Tran_{ETE}(\tau) = Tran_{ETE}(\tau')$.
3. If $\phi \equiv \omega = \omega'$ then replace ϕ in ψ with $Tran_{EOE}(\omega) = Tran_{EOE}(\omega')$.
4. If $\phi \equiv \tau \otimes \tau'$ then replace ϕ in ψ with $Tran_{ETE}(\tau) \otimes Tran_{ETE}(\tau')$.

Under the assumption that there is a straightforward mapping of the logical connectives, quantifiers, and delimiters, then after applying all possible substitutions, we are left with a wff in $\mathcal{L}(FL)$.

Remark A.3 *We will assume that $Tran_{LF}(\psi)$ denotes the corresponding wff in $\mathcal{L}(FL)$. In addition, if $Tran_{LF}(\psi)$, where ψ might include sub-formulas where one of the defined transformation operators $Tran_*$ is applied, then it is assumed that the outer $Tran_{LF}$ simply leaves the result of $Tran_*$ unchanged.*

A.2 Reassignment Formulas in $\mathcal{L}(SD)$

Reassignment formulas play a very important role in PMON and represent the assignment of a new value to a fluent. The reassignment operator is denoted by ":=".

Definition A.25 (Elementary Reassignment Expression (ERE))
An *elementary reassignment expression* has the form

- $f_i := \{\nu\}$

where f_i is an EFE and $v \in Dom_\nu(f_i)$.

The following abbreviations will be used. The first two apply to any value domain for f_i while the last two apply when $Dom_\nu(f_i)$ is Boolean:

$f_i := \nu \stackrel{\text{def}}{=} f_i := \{\nu\}$.

$\neg f_i := \nu \stackrel{\text{def}}{=} f_i := Dom_\nu(f_i) \setminus \{\nu\}$.

$\neg f_i := \{T\} \stackrel{\text{def}}{=} f_i := \{F\}$.

$\neg f_i := \{F\} \stackrel{\text{def}}{=} f_i := \{T\}$.

Definition A.26 (Reassignment Expression (RE))
A *reassignment expression* is an expression of the form

- $f_i := \mathcal{X}$,

where f_i is an EFE and $\mathcal{X}$ is a non-empty subset of $Dom_\nu(f_i)$. A reassignment expression is an abbreviation for a disjunction of ERE's. Let ϕ be the RE $f_i := \{\nu_1, \ldots, \nu_n\}$, then

$$\phi \stackrel{\text{def}}{=} \bigvee_{j=1}^{n} f_i := \nu_j .$$

The following abbreviation will be used.

$$\neg f_i := \mathcal{X} \stackrel{\text{def}}{=} f_i := Dom_\nu(f_i) \setminus \mathcal{X}.$$

Remark A.4 *For the boolean value domain $\mathcal{B}$, there are three cases:*

1. $\mathbf{f}_i := \{T\}$.
2. $\mathbf{f}_i := \{F\}$.
3. $\mathbf{f}_i := \{T, F\} \stackrel{\text{def}}{=} \mathbf{f}_i := \{T\} \ \vee \ \mathbf{f}_i := \{F\}$.

Definition A.27 (Elementary Reassignment Formula (ERF))
An *elementary reassignment formula* is a formula of type

- $[\tau, \tau']f_i := \mathcal{X}$, where both τ, τ' are elementary time-point expressions and $f_i := \mathcal{X}$ is a RE. The following abbreviation will be used:

$$[\tau, \tau']f_i := \{\nu_1, \ldots \nu_n\} = [\tau, \tau'] \bigvee_{j=1}^{n} f_i := \nu_j \stackrel{\text{def}}{=} \bigvee_{j=1}^{n} [\tau, \tau']f_i := \nu_j .$$

When $\mathcal{X}$ is a singleton set, $[\tau, \tau']f_i := \mathcal{X}$ will be referred to as a *simple ERF* . It is easily observed that ERF's are simply abbreviations for a disjunction of simple ERF's.

Definition A.28 (Restricted Reassignment Formula (RRF))
A *restricted reassignment formula* is a conjunction of ERF's

$$\left(\bigwedge_{j=1}^{n} [\tau, \tau']f_j := \mathcal{X}_j \right) \wedge \psi(\bar{t}, \bar{o}),$$

together with a logic formula $\psi(\bar{t}, \bar{o})$ representing *duration constraints* for the interval $[\tau, \tau']$, where $[\tau, \tau']$ is the same for each conjunct and no two EFE's, f_j and $f_{j'}$, are the same. In addition, $\bar{t}$ and $\bar{o}$, denoting the free temporal and object variables in ψ, are restricted to be only those variables mentioned freely in any of the f_j, τ, or τ', and constraints in ψ may only be placed on the elementary feature expressions f_j between $\tau + 1$ and τ', inclusively.

The following abbreviation will be used:

$$[\tau, \tau'] \bigwedge_{j=1}^{n} f_j := \mathcal{X}_j \stackrel{\text{def}}{=} \bigwedge_{j=1}^{n} [\tau, \tau']f_j := \mathcal{X}_j$$

Proposition A.1
Let ϕ be an arbitrary restricted reassignment formula in the surface language for scenario descriptions. There is an effective algorithm that transforms ϕ into a formula in conjunctive normal form, where each conjunct is a disjunction of simple elementary reassignment formulas.

Proof

Straightforward. Follows from the definitions.

Definition A.29 (Reassignment Formula (RF))
A *reassignment formula* is a disjunction of restricted reassignment formulas (RRF's)

$$\bigvee_{i=1}^{k} \left(\left(\bigwedge_{j=1}^{n_i} [\tau, \tau'] f_{i_j} := \mathcal{X}_{i_j} \right) \wedge \psi_i(\bar{o}, \bar{t}) \right)$$

where the interval $[\tau, \tau']$ is the same for each RRF.

The following abbreviation will be used:

$$[\tau, \tau'] \bigvee_{i=1}^{k} \left(\left(\bigwedge_{j=1}^{n_i} f_{i_j} := \mathcal{X}_{i_j} \right) \wedge \psi_i(\bar{o}, \bar{t}) \right) \stackrel{\text{def}}{=} \bigvee_{i=1}^{k} \left(\left(\bigwedge_{j=1}^{n_i} [\tau, \tau'] f_{i_j} := \mathcal{X}_{i_j} \right) \wedge \psi_i(\bar{o}, \bar{t}) \right)$$

Remark A.5 *An additional restriction will be placed on reassignment formulas for the class of action theories we deal with.*

If the elementary feature expression f_{m_j} is mentioned in the mth disjunct of a reassignment formula, then it must be mentioned in all the k disjuncts. If this is not the case when using a reassignment formula in an action law specification then it is assumed that during the translation process the elementary reassignment formula $[\tau, \tau'] f_{m_j} := \mathcal{X}$, where $\mathcal{X} = \mathcal{D}_{m_j}$ (the whole value domain for f_{m_j}), is added to each disjunct where f_{m_j} is not mentioned. Note that this does not influence the semantics of the reassignment formula other than to ensure that the same feature expressions are occluded in each disjunct of the reassignment expression. In other words, legally the occlusion specification can not be any other way.

Alternatively, when translating a reassignment formula into a wff in $\mathcal{L}(FL)$, one need only modify the occlusion assertions so that any feature expression mentioned in a reassignment formula is occluded. One simply defines a set of influenced features *as the union of the sets of feature expressions mentioned in each disjunct. When doing the translation, it is assumed that each feature expression in the set is occluded.*

Definition A.30 (Conditional Reassignment Formula (CRF))
A *conditional reassignment formula* is a formula of the following type

- $\forall \bar{t} \forall \bar{o}. ([\tau] \phi(\bar{o}) \rightarrow [\tau, \tau'] \theta(\bar{o}))$,

where $[\tau]\phi(\bar{o})$ is a fixed fluent formula and $[\tau, \tau']\theta(\bar{o})$ is a reassignment formula. The variables $\bar{t}, \bar{o}$ of sorts $\mathcal{O}_i$ and $\mathcal{T}$, respectively, are free in ϕ and θ. The variables $\bar{t}$ are restricted to those that are free in the elementary time-point expressions τ and τ'.

A *restricted conditional reassignment formula* is a CRF where $[\tau, \tau']\theta$ is a restricted reassignment formula.

A.2.1 Reducing Reassignment Formulas in $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$

Definition A.31 ($Tran_{SERF}$, $Tran_{ERF}$)
Let ϕ be a simple elementary reassignment formula $[\tau, \tau']f_k(\omega_1, \ldots, \omega_{i_k}) := \mathcal{X}$, where $\mathcal{X} = \{\nu_j\}$, $Dom_\nu(f_k)$ is non-Boolean, and f_k has arity i_k and sort

$$Dom_\nu(f_k^1) \times \ldots \times Dom_\nu(f_k^{i_k}) \rightarrow Dom_\nu(f_k).$$

$$\begin{aligned} Tran_{SERF}(\phi) = Holds(\rho', f_k(\pi_1, \ldots \pi_n, \mathsf{d}_j^k)) \wedge \\ \forall t \forall d^k(\rho < t \leq \rho' \rightarrow Occlude(t, f_k(\pi_1, \ldots \pi_n, d^k))) \end{aligned}$$

where $\rho = Tran_{ETE}(\tau), \rho' = Tran_{ETE}(\tau')$, $\mathsf{d}_j^k = Tran_{E(OV)E}(\nu_j)$ is a value or object name constant of sort D_k associated with each object of sort $Dom_\nu(f_k) = \mathcal{D}_k$, d^k is a variable of sort D_k, each $\pi_m = Tran_{EOE}(\omega_m)$ is either a constant or variable of the appropriate sorts O_l associated with sorts $Dom_\nu(f_k^l)$, and f_k is the function symbol of arity $i_k + 1$ in $\mathcal{L}(FL)$ associated with f_k in $\mathcal{L}(SD)$.

If $Dom_\nu(f_k)$ is Boolean and $\mathcal{X} = \{T\}$ then

$$\begin{aligned} Tran_{SERF}(\phi) = Holds(\rho', f_k(\pi_1, \ldots \pi_n)) \wedge \\ \forall t(\rho < t \leq \rho' \rightarrow Occlude(t, f_k(\pi_1, \ldots \pi_n))) \end{aligned}$$

If $Dom_\nu(f_k)$ is Boolean and $\mathcal{X} = \{F\}$ then

$$\begin{aligned} Tran_{SERF}(\phi) = \neg Holds(\rho', f_k(\pi_1, \ldots \pi_n)) \wedge \\ \forall t(\rho < t \leq \rho' \rightarrow Occlude(t, f_k(\pi_1, \ldots \pi_n))) \end{aligned}$$

Let ϕ be an elementary reassignment formula $[\tau, \tau']f_k(\omega_1, \ldots, \omega_{i_k}) := \mathcal{X}$, as above, but where $\mathcal{X} = \{\nu_1, \ldots, \nu_n\}$.

$$\begin{aligned} Tran_{ERF}(\phi) = \bigvee_{\nu_j \in \mathcal{X}} Holds(\rho', f(\pi_1, \ldots \pi_n, \mathsf{d}_j^k)) \wedge \\ \forall t \forall d^k(\rho < t \leq \rho' \rightarrow Occlude(t, f(\pi_1, \ldots \pi_n, d^k))) \end{aligned}$$

Let ϕ be an elementary reassignment formula $[\tau, \tau']\neg f_k(\omega_1, \ldots, \omega_{i_k}) := \mathcal{X}$, as above, but where $\mathcal{X} = \{\nu_1, \ldots, \nu_n\}$.

$$\begin{aligned} Tran_{ERF}(\phi) = \bigwedge_{\nu_j \in \mathcal{X}} \neg Holds(\rho', f(\pi_1, \ldots \pi_n, \mathsf{d}_j^k)) \wedge \\ \forall t \forall d^k(\rho < t \leq \rho' \rightarrow Occlude(t, f(\pi_1, \ldots \pi_n, d^k))) \end{aligned}$$

Remark A.6 *Note that both $[\tau, \tau']f_k(\omega_1, \ldots, \omega_{i_k}) := \mathcal{X}$ and $[\tau, \tau']\neg f_k(\omega_1, \ldots, \omega_{i_k}) := \mathcal{X}$ are defined as abbreviations for disjunctions of simple elementary reassignment formulas. We define specific translations for the unabbreviated forms because they are equivalent to the translations for the abbreviated forms when the unique values and existence of values axioms are taken into account, and these translations will also be useful when we choose to use infinite value domains.*

Proposition A.2 ($Tran_{CRF}$)
Let $\psi = \forall \bar{t} \forall \bar{o}.([\tau]\phi(\bar{o}) \rightarrow [\tau, \tau']\theta(\bar{o}))$ be a conditional reassignment formula in $\mathcal{L}(SD)$. There is a transformation operator $Tran_{CRF}$ that transforms ψ into a wff $\alpha = Tran_{CRF}(\psi)$ in $\mathcal{L}(FL)$.

Proof

Since $[\tau]\phi(\bar{o})$ is a logical formula, Lemma (A.1) shows that $[\tau]\phi(\bar{o})$ can be effectively transformed into a wff β in $\mathcal{L}(FL)$. Since $[\tau, \tau']\theta(\bar{o})$ is a reassignment formula, Lemma (A.1) together with Definition A.31 shows that $[\tau, \tau']\theta(\bar{o})$ can be effectively transformed into a wff β' in $\mathcal{L}(FL)$. Let $\alpha = \forall \bar{t} \forall \bar{o}.(\beta \rightarrow \beta')$. α is a wff in $\mathcal{L}(FL)$.

Proposition A.3
Let $\Phi = \left(\bigwedge_{i=1}^{k} Tran_{ERF}(\phi_i)\right) \wedge Tran_{LF}(\psi(\bar{o},\bar{t}))$ be a formula in $\mathcal{L}(FL)$ which is the result of translating a restricted reassignment formula consisting of a conjunction $\bigwedge_{i=1}^{k} \phi_i$ of elementary reassignment formulas together with a duration constraint $\psi(\bar{o},\bar{t})$ in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$. Φ can be transformed into the logically equivalent form,

$$\Gamma \wedge \forall t \forall z \forall \bar{d}.\Theta(t,z,\bar{d},\bar{o},\bar{t}) \rightarrow Occlude(t,z).$$

Proof Any $Tran_{ERF}(\phi_i)$ has the following form,

$$\begin{aligned}&(\bigvee_{\nu_j \in \mathcal{X}_i} Holds(\rho', f_i(\pi_1, \ldots \pi_{i_n}, \mathsf{d}_{i_j})))\wedge \\ &\quad \forall t \forall d_i(\rho < t \leq \rho' \rightarrow Occlude(t, f(\pi_1, \ldots \pi_n, d_i)))\end{aligned}$$

which can be transformed into the logically equivalent form,

$$\begin{aligned}&(\bigvee_{\nu_j \in \mathcal{X}_i} Holds(\rho', f_i(\pi_1, \ldots \pi_n, \mathsf{d}_{i_j})))\wedge \\ &\quad \forall t \forall z \forall d_i((\rho < t \leq \rho' \wedge z = f_i(\pi_1, \ldots \pi_n, d_i)) \rightarrow Occlude(t,z)).\end{aligned}$$

It is easily observed that $\bigwedge_{i=1}^{k} Tran_{ERF}(\phi_i)$ has the following form[3],

$$\begin{aligned}&(\bigwedge_{i=1}^{k} \bigvee_{\nu_j \in \mathcal{X}_i} Holds(\rho', f_i(\pi_1, \ldots \pi_n, \mathsf{d}_{i_j})))\wedge \\ &\quad \forall t \forall z (\bigwedge_{i=1}^{k} \forall d_i)((\rho < t \leq \rho' \wedge (\bigvee_{i=1}^{k} z = f_i(\pi_1, \ldots \pi_n, d_i))) \rightarrow Occlude(t,z)).\end{aligned}$$

Let

$$\Gamma = \left(\bigwedge_{i=1}^{k} \bigvee_{\nu_j \in \mathcal{X}_i} Holds(\rho', f_i(\pi_1, \ldots \pi_n, d_{i_j}))\right) \wedge Tran_{LF}(\psi(\bar{o},\bar{t})),$$

where $\psi(\bar{o},\bar{t})$ is the duration constraint for the restricted reassignment formula being transformed, and

$$\Theta(t,z,\bar{d},\bar{o},\bar{t}) = ((\rho < t \leq \rho' \wedge (\bigvee_{i=1}^{k} z = f_i(\pi_1, \ldots \pi_n, d_i))).$$

Remark A.7 *Note that a restricted reassignment formula is used in the specification of an action law where the postconditions to an action in a particular context ϕ can be a disjunction of restricted reassignment formulas. The proposition above provides a very nice characterization of one alternative in a nondeterministic action. Γ specifies the effects of the alternative, where we observe that even in a specific alternative there may be* new value ambiguity. *$\Theta(t,z,\bar{d},\bar{o},\bar{t})$ specifies the sufficient conditions for locally releasing the global inertia constraint built into the logic's minimization policy.*

Proposition A.4
Let $\Psi = \bigvee_{l=1}^{m} \left(\left(\bigwedge_{i=1}^{k_l} Tran_{ERF}(\phi_i)\right) \wedge Tran_{LF}(\psi_l(\bar{o},\bar{t}))\right)$ be a formula in $\mathcal{L}(FL)$ satisfying the restriction stated in Remark A.5, which is the result of translating a disjunction of restricted reassignment formulas (i.e. reassignment formula) in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$. Ψ can be transformed into the logically equivalent form,

$$(\bigvee_{l=1}^{m} \Gamma_l) \wedge \forall t \forall z \forall \bar{d}.\Theta(t,z,\bar{d},\bar{o},\bar{t}) \rightarrow Occlude(t,z).$$

[3] Note that in the following, we will sometimes abuse the notation and common meaning of the $\bigwedge$ operator by applying it to quantifiers. For example, $(\bigwedge_{i=1}^{k} \forall d_i)$ denotes $\forall d_1, \ldots, \forall d_k$ or equivalently $\forall d_1, \ldots, d_k$.

Proof By Proposition A.3, any two restricted reassignment formulas in Ψ have the following form:

$$\Gamma_i \wedge \forall t \forall z \forall \bar{d}_i . \Theta_i(t, z, \bar{d}_i, \bar{o}_i, \bar{t}) \rightarrow Occlude(t, z).$$

$$\Gamma_j \wedge \forall t \forall z \forall \bar{d}_j . \Theta_j(t, z, \bar{d}_j, \bar{o}_j, \bar{t}) \rightarrow Occlude(t, z).$$

Based on Remark A.5,$\bar{d}_i = \bar{d}_j$ and $\Theta_i(t, z, \bar{d}_i, \bar{o}_i, \bar{t}) = \Theta_j(t, z, \bar{d}_j, \bar{o}_j, \bar{t})$. It is easily observed that the reassignment formula Ψ has the following form,

$$\bigvee_{l=1}^{m} (\Gamma_l \wedge \forall t \forall z \forall \bar{d} . \Theta(t, z, \bar{d}, \bar{o}, \bar{t}) \rightarrow Occlude(t, z)),$$

which is equivalent to

$$(\bigvee_{l=1}^{m} \Gamma_l) \wedge \forall t \forall z \forall \bar{d} . \Theta(t, z, \bar{d}, \bar{o}, \bar{t}) \rightarrow Occlude(t, z).$$

Remark A.8 *Note that a reassignment formula is used in the specification of an action law where the postcondition to an action in a particular context ϕ is a reassignment formula. The proposition above provides a very nice characterization of the postcondition of a nondeterministic action. Each of the Γ_i specifies the effects of an alternative, while $\Theta(t, z, \bar{d}, \bar{o}, \bar{t})$ specifies the sufficient conditions for locally releasing the global inertia constraint built into the logic's minimization policy. In this case, $\Theta(t, z, \bar{d}, \bar{o}, \bar{t})$ is the same for each alternative in an action with precondition ϕ.*

Proposition A.5
Let Ψ be a formula in $\mathcal{L}(FL)$ which is the result of transforming a conditional reassignment formula in $\mathcal{L}(SD)$ satisfying the restriction stated in Remark A.5 using the appropriate transformation operators. Ψ can be transformed into the logically equivalent form,

$$\Gamma \wedge \forall t \forall z \forall \bar{t} \forall \bar{o} . (\gamma_1 \wedge (\forall \bar{d} . \Theta(t, z, \bar{d}, \bar{t}, \bar{o}))) \rightarrow Occlude(t, z)).$$

Proof A conditional reassignment formula has the following form,

$$\forall \bar{t} \forall \bar{o} . ([\tau]\phi(\bar{o}) \rightarrow [\tau, \tau']\theta(\bar{o})).$$

After translating this into a formula in $\mathcal{L}(FL)$ and applying the transformation in Proposition A.4, it has the following form,

$$\forall \bar{t} \forall \bar{o} . \gamma_1 \rightarrow \left((\bigvee_{l=1}^{m} \Gamma_l) \wedge \forall t \forall z \forall \bar{d} . \Theta(t, z, \bar{d}, \bar{o}, \bar{t}) \rightarrow Occlude(t, z) \right).$$

where $\gamma_1 = Tran_{LF}([\tau]\phi(\bar{o}))$, and we assume that all variables, logical connectives, and delimiters have been translated appropriately. It is easily observed that this formula is equivalent to the following conjunction,

$$\forall \bar{t} \forall \bar{o} . [\gamma_1 \rightarrow \bigvee_{l=1}^{m} \Gamma_l] \wedge$$
$$\forall t \forall z \forall \bar{t} \forall \bar{o} . [(\gamma_1 \wedge (\forall \bar{d} . \Theta(t, z, \bar{d}, \bar{o}, \bar{t}))) \rightarrow Occlude(t, z)].$$

Let

$$\Gamma = \forall \bar{t} \forall \bar{o} . [\gamma_1 \rightarrow \bigvee_{l=1}^{m} \Gamma_l].$$

Remark A.9 *Proposition A.5 provides a nice characterization of one context in a context dependent action. γ_1 is the precondition, which if satisfied has the effect*

$$\bigvee_{l=1}^{m} \Gamma_l,$$

where $\bigvee_{l=1}^{m} \Gamma_l$ describes the effects of the action after it is executed, including the duration constraints. If $m > 1$ then the action is nondeterministic in context γ_1 and has alternative effects.

Lemma A.2
Let $\Psi = \bigwedge_{i=1}^{k} Tran_{CR}(\phi_i)$ be a formula in $\mathcal{L}(FL)$ satisfying the restriction stated in Remark A.5, which is the result of translating a conjunction $\bigwedge_{i=1}^{k} \phi_i$ of conditional reassignment formulas in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$.

Ψ can be transformed into the logically equivalent form,

$$(\bigwedge_{j=1}^{k} \Gamma_j) \wedge \forall t \forall z \forall \bar{t} (\bigwedge_{j=1}^{k} \forall \bar{o}_j \forall \bar{d}_j).[\bigvee_{j=1}^{k} (\gamma_j \wedge \Theta_j(t, z, \bar{d}_j, \bar{t}, \bar{o}_j))] \rightarrow Occlude(t, z)).$$

Proof By Proposition A.5, any translation of a conditional reassignment formula can be transformed into the following form,

$$\Gamma_j \wedge \forall t \forall z \forall \bar{t} \forall \bar{o}_j .(\gamma_j \wedge (\forall \bar{d}_j .\Theta(t, z, \bar{d}_j, \bar{t}, \bar{o}_j))) \rightarrow Occlude(t, z)).$$

A conjunction of such formulas has the form,

$$\bigwedge_{j=1}^{k} (\Gamma_j \wedge \forall t \forall z \forall \bar{t} \forall \bar{o}_j .(\gamma_j \wedge (\forall \bar{d}_j .\Theta_j(t, z, \bar{d}_j, \bar{t}, \bar{o}_j))) \rightarrow Occlude(t, z))).$$

It is easily observed that this is equivalent to

$$(\bigwedge_{j=1}^{k} \Gamma_j) \wedge \forall t \forall z \forall \bar{t} (\bigwedge_{j=1}^{k} \forall \bar{o}_j \forall \bar{d}_j).[\bigvee_{j=1}^{k} (\gamma_j \wedge \Theta_j(t, z, \bar{d}_j, \bar{t}, \bar{o}_j))] \rightarrow Occlude(t, z)).$$

Remark A.10 *The result of instantiating an action law schema is a conjunction of conditional reassignment formulas, Proposition A.4 provides a nice modular characterization of an action law which separates the specification of the effects of the action for each context γ_j,*

$$\bigwedge_{j=1}^{k} \Gamma_j,$$

from the specification $\Theta_j(t, z, \bar{d}_j, \bar{t}, \bar{o}_j)$, of the sufficient conditions for relaxing the inertia policy for the specified set of fluents influenced by the action in each context γ_j,

$$\forall t \forall z \forall \bar{t} (\bigwedge_{j=1}^{k} \forall \bar{o}_j \forall \bar{d}_j).[\bigvee_{j=1}^{k} (\gamma_j \wedge \Theta_j(t, z, \bar{d}_j, \bar{t}, \bar{o}_j))] \rightarrow Occlude(t, z)).$$

Any expanded action scenario description contains a finite set of instantiations of action law schemas. The set of instantiations is denoted by Γ_{SCD}. The next Lemma will show that a conjunction of schedule statements can be put into a form which will be useful when reducing the circumscription of Γ_{SCD} to the first-order case.

Lemma A.3
Let $\Psi = \bigwedge_{l=1}^{m} \bigwedge_{i=1}^{k_l} Tran_{CR}(\phi_i)$ be a formula in $\mathcal{L}(FL)$ satisfying the restriction stated in Remark A.5, which is the result of translating a conjunction $\bigwedge_{l=1}^{m} \bigwedge_{i=1}^{k_l} \phi_i$ of conjunctions of conditional reassignment formulas in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$.

Ψ can be transformed into the logically equivalent form,

$$\Gamma \wedge \forall t \forall z.(\Delta \rightarrow Occlude(t,z)).$$

Proof By Lemma A.2, a conjunction of conditional reassignment formulas (i.e. schedule statement) has the following form,

$$(\bigwedge_{j=1}^{k} \Gamma_j) \wedge \forall t \forall z \forall \bar{t}(\bigwedge_{j=1}^{k} \forall \bar{o}_j \forall \bar{d}_j).[\bigvee_{j=1}^{k} (\gamma_j \wedge \Theta_j(t,z,\bar{d}_j,\bar{t},\bar{o}_j))] \rightarrow Occlude(t,z)).$$

It is easily observed that a conjunction of such formulas has the following form,

$$(\bigwedge_{l=1}^{m} \bigwedge_{j=1}^{k_l} \Gamma_{j_l}) \wedge$$
$$(\forall t \forall z(\bigwedge_{l=1}^{m} \forall \bar{t}_l(\bigwedge_{j=1}^{k_l} \forall \bar{o}_{j_l} \forall \bar{d}_{j_l})).[\bigvee_{l=1}^{m} \bigvee_{j=1}^{k_l} (\gamma_{j_l} \wedge \Theta_{j_l}(t,z,\bar{d}_{j_l},\bar{t}_l,\bar{o}_{j_l}))]$$
$$\rightarrow Occlude(t,z)).$$

Let

$$\Gamma = (\bigwedge_{l=1}^{m} \bigwedge_{j=1}^{k_l} \Gamma_{j_l})$$

and

$$\Delta = (\bigwedge_{l=1}^{m} \forall \bar{t}_l(\bigwedge_{j=1}^{k_l} \forall \bar{o}_{j_l} \forall \bar{d}_{j_l})).[\bigvee_{l=1}^{m} \bigvee_{j=1}^{k_l} (\gamma_{j_l} \wedge \Theta_{j_l}(t,z,\bar{d}_{j_l},\bar{t}_l,\bar{o}_{j_l}))].$$

Theorem A.1
Let $\Psi = \bigwedge_{l=1}^{m} \bigwedge_{i=1}^{k_l} Tran_{CR}(\phi_i)$ be a formula in $\mathcal{L}(FL)$ satisfying the restriction stated in Remark A.5, which is the result of translating a conjunction $\bigwedge_{l=1}^{m} \bigwedge_{i=1}^{k_l} \phi_i$ of conjunctions of conditional reassignment formulas in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$.

The circumscription of Ψ with $Occlude$ minimized and all other predicates fixed $Circ_{SO}(\Psi; Occlude)$, is equivalent to the following first-order formula

$$\Gamma \wedge \forall t \forall z.(\Delta \equiv Occlude(t,z)),$$

where

$$\Gamma = (\bigwedge_{l=1}^{m} \bigwedge_{j=1}^{k_l} \Gamma_{j_l})$$

and

$$\Delta = (\bigwedge_{l=1}^{m} \forall \bar{t}_l(\bigwedge_{j=1}^{k_l} \forall \bar{o}_{j_l} \forall \bar{d}_{j_l})).[\bigvee_{l=1}^{m} \bigvee_{j=1}^{k_l} (\gamma_{j_l} \wedge \Theta_{j_l}(t,z,\bar{d}_{j_l},\bar{t}_l,\bar{o}_{j_l}))].$$

Proof By Lemma A.3, Ψ can be transformed into the form $\Gamma \wedge \forall t \forall z.(\Delta \rightarrow Occlude(t,z))$. Theorem B.1 by Lifschitz allows us to factor out Γ when circumscribing Ψ with $Occlude$ minimized and all other predicates fixed, since Γ contains no occurrences of $Occlude$. Theorem B.2 by Lifschitz shows us that

$$Circ_{SO}(\{\forall t,z.(\Delta \rightarrow Occlude(t,z))\}; Occlude) \equiv \Gamma \wedge \forall t,z.(\Delta \equiv Occlude(t,z)).$$

The proof of Lemma A.3 can be used to show that Γ and Δ have the values stated in the theorem.

A.3 Causal Constraint Statements in $\mathcal{L}(SD)$

Definition A.32
A *causal constraint* statement in $\mathcal{L}(SD)$ has the following form:

$$\forall t_1 \forall \bar{o}.[Q_{\bar{o}_0}.\gamma(\bar{o}_0, \bar{o}) \rightarrow ([t_1]Q_{\bar{o}_1}.\alpha(\bar{o}_1, \bar{o}) \gg [\tau]Q_{\bar{o}_2}.\beta(\bar{o}_2, \bar{o}))]$$

where $\bar{o}_0, \bar{o}_1, \bar{o}_2, \bar{o}$ are disjoint tuples of variables of sorts $\mathcal{O}$, and $Q_{\bar{o}_1}, Q_{\bar{o}_1}, Q_{\bar{o}_2}$ are finite sequences of quantifiers binding the variables $\bar{o}_0, \bar{o}_1, \bar{o}_2$, respectively, α and β are quantifier free fluent formulas (FlF's), and γ is a quantifier free logic formula where any elementary time-point expression τ' in γ is equal to a function $g_{\tau'}$ of t_1, where $g_{\tau'}(t_1) \leq t_1$. In addition the temporal expression τ is a function g_τ of t_1 where $g_\tau(t_1) \geq t_1$.

We will use the label *cc* to label causal constraint statements in action scenarios.

Remark A.11 *Observe that it is very difficult to provide a strict definition of a causal constraint statement. This definition allows for certain anomalies which certain temporal structures would rule as illegal. For example, in γ, one has to be careful that the interpretation of any temporal expression is greater than 0 if a linear discrete time structure with begin point is being used. In addition, it may be necessary to add additional constraints when defining the translation $Tran_{CC}$ (see below) which translates causal constraint statements to wffs in $\mathcal{L}(FL)$.*

Currently, there are also a number of restrictions placed on the relation between the different time-points mentioned in a causal constraint statement which are most probably over restrictive. For example, it is sometimes useful to mention time-points greater than t_1 in the context γ. The restrictions placed on the context $Q_{\bar{o}_0}.\gamma(\bar{o}_0, \bar{o})$, can be relaxed without difficulty so that a context is simply a logic formula *in $\mathcal{L}(SD)$.*

A.3.1 Reducing Causal Constraint Statements in $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$

Definition A.33 ($Tran_{EFE}$)
Let $\phi = f_k(\omega_1, \ldots, \omega_{i_k}) \doteq \mathcal{X}$ be an elementary fluent formula, where $\mathcal{X} = \{\nu_1, \ldots, \nu_n\}$, $Dom_\nu(f_k)$ is non-boolean, and f_k has arity i_k and sort $Dom_\nu(f_k^1) \times \ldots \times Dom_\nu(f_k^{i_k}) \rightarrow Dom_\nu(f_k)$. $Tran_{EFE}(\phi)$ is defined as follows:

$$Tran_{EFE}(\phi) = \{f_k'(\pi_1, \ldots, \pi_{i_k}, d^k)\},$$

where $f_k'(\pi_1, \ldots, \pi_{i_k}, d^k)$ is a parameterized term in $\mathcal{L}(FL)$, and d^k is a variable of sort D_k of $\mathcal{L}(FL)$, associated with the sort $Dom_\nu(f_k)$.

If $Dom_\nu(f_k)$ is boolean, then

$$Tran_{EFE}(\phi) = \{f_k'(\pi_1, \ldots, \pi_{i_k})\}.$$

If ϕ is a logic formula, we define $Tran_{EFE^*}(\phi)$ as follows:

$$Tran_{EFE^*}(\phi) = \bigcup_{f \in \Delta} Tran_{EFE}(f),$$

where Δ is the set of elementary fluent formula in ϕ.

Definition A.34 ($Tran_X$)
Let $\phi = [\tau]Q_{\bar{o}_2}.\beta(\bar{o}_2, \bar{o})$ be a logical formula (LF), where $\beta(\bar{o}_2, \bar{o})$ is a quantifier free fluent formula.

$$Tran_X(\phi) = Q_X. \bigwedge_{f_i(\pi_1, \ldots, \pi_{i_n}, d^i) \in Tran_{EFE^*}(\phi)} Occlude(\rho, f_i(\pi_1, \ldots, \pi_{i_n}, d^i)),$$

where,

1. $\rho = Tran_{ETE}(\tau)$ is the translation of the ETE τ into $\mathcal{L}(FL)$.
2. Q_X is a sequence of universally quantified variables constructed from $Q_{\bar{o}_2}$ by
   - replacing each existential quantifier in $Q_{\bar{o}_2}$ with a universal quantifier,
   - renaming all variables $\bar{o}_2$ and $\bar{d}$, where $\bar{d}$ are the free variables of sorts D_i in $Tran_{EFE^*}(\phi)$, with new and unique variables not yet used in the formula of which ϕ is a part,
   - and prefixing $\forall \bar{d}$ to $Q_{\bar{o}_2}$.

Definition A.35 ($Tran_{CC}$)
Let $\phi = \forall t_1 \forall \bar{o}.[Q_{\bar{o}_0}.\gamma(\bar{o}_0, \bar{o}) \rightarrow ([t_1]Q_{\bar{o}_1}.\alpha(\bar{o}_1, \bar{o}) \gg [\tau]Q_{\bar{o}_2}.\beta(\bar{o}_2, \bar{o}))]$.

$$
\begin{aligned}
&Tran_{CC}(\phi) = \\
&\quad Tran_{LF}(\forall t_1 \forall \bar{o}.[Q_{\bar{o}_0}.\gamma(\bar{o}_0, \bar{o}) \rightarrow ([t_1]Q_{\bar{o}_1}.\alpha(\bar{o}_1, \bar{o}) \rightarrow [\tau]Q_{\bar{o}_2}.\beta(\bar{o}_2, \bar{o}))]) \wedge \\
&\quad Tran_{LF}(\forall t_1 \forall \bar{o}.[(0 < t_1 \wedge Q_{\bar{o}_0}.\gamma(\bar{o}_0, \bar{o})) \rightarrow [([t_1 - 1]\neg Q_{\bar{o}_1}.\alpha(\bar{o}_1, \bar{o}) \wedge [t_1]Q_{\bar{o}_1}.\alpha(\bar{o}_1, \bar{o})) \\
&\qquad \rightarrow Tran_X([\tau]Q_{\bar{o}_2}.\beta(\bar{o}_2, \bar{o}))]])
\end{aligned}
$$

Remark A.12 *Note the insertion of $0 < t_1$ in the context of the Occlude formula. This is inserted to avoid anomalies at the initial point in the time-line. Another means of avoiding this problem would be to axiomatize change from t_1 to $t_1 + 1$ instead of $t_1 - 1$ to t_1, but this would necessitate change in the temporal terms for the domain constraint. Semantically, there is no distinction between these two approaches. In the report, we will sometimes assume $0 < t_1$ without actually stating it explicitly in the axioms.*

Lemma A.4
Let $\psi = \forall t_1 \forall \bar{o}.[(Q_{\bar{o}_0}.\gamma(\bar{o}_0, \bar{o})) \rightarrow ([t_1]Q_{\bar{o}_1}.\alpha(\bar{o}_1, \bar{o}) \gg [\tau]Q_{\bar{o}_2}.\beta(\bar{o}_2, \bar{o}))]$, be a causal constraint statement in $\mathcal{L}(SD)$. There is an effective algorithm that transforms ψ into a wff α in $\mathcal{L}(FL)$.

Proof Since $Tran_X$ returns a formula in $\mathcal{L}(FL)$, it is easily observable that $Tran_{CC}(\psi)$ is a boolean combination of logic formulas, possibly quantified, together with the reduced argument to $Tran_X$. Application of $Tran_{LF}$ in Lemma A.1 completes the reduction.

The following Lemmas and Propositions will prove to be useful when circumscribing theories containing causal constraints. They provide a basis for first-order reductions of circumscriptive theories.

Lemma A.5
Let $\Gamma = \bigwedge_{i=1}^{k} Tran_{CC}(\phi_i)$ be a formula in $\mathcal{L}(FL)$ which is the result of translating a conjunction $\bigwedge_{i=1}^{k} \phi_i$ of causal constraint statements in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$. The predicate *Occlude* only occurs positively in Γ.

Proof Each $Tran_{CC}(\phi_i)$ has the form

$$
\Gamma_1 \wedge (\forall t_1 \forall \bar{o}.\Theta(\bar{o}) \rightarrow Q_X. \bigwedge_{f_i(\pi_1, \ldots, \pi_{i_n}, d^i) \in Tran_{EFE^*}(\phi)} Occlude(\rho, f_i(\pi_1, \ldots, \pi_{i_n}, d^i)).
$$

Γ_1 contains no occurrences of *Occlude* and

$$
(\forall t_1 \forall \bar{o}.\Theta(\bar{o}) \rightarrow Q_X. \bigwedge_{f_i(\pi_1, \ldots, \pi_{i_n}, d^i) \in Tran_{EFE^*}(\phi)} Occlude(\rho, f_i(\pi_1, \ldots, \pi_{i_n}, d^i))
$$

can be rewritten equivalently as

$$
\forall t_1 \forall \bar{o} Q_X.\Theta(\bar{o}) \rightarrow \bigwedge_{f_i(\pi_1, \ldots, \pi_{i_n}, d^i) \in Tran_{EFE^*}(\phi)} Occlude(\rho, f_i(\pi_1, \ldots, \pi_{i_n}, d^i)).
$$

It is easily observed that the negation normal form contains no negative instances of $Occlude$. Since the translation can be applied to each $Tran_{CC}(\phi_i)$, it follows that Γ contains no positive occurrences of $Occlude$.

Proposition A.6
Let $\phi = Tran_{CC}(\psi)$ be the formula in $\mathcal{L}(FL)$ which is the result of applying $Tran_{CC}$ to the causal constraint statement ψ in $\mathcal{L}(SD)$. Any ϕ can be transformed into an equivalent formula with the following form:

$$\Gamma_1 \wedge \forall t \forall z.\Phi \rightarrow Occlude(t,z).$$

Proof By Lemma (A.5), $Tran_{CC}(\psi)$ has the form

$$\forall t_1 \forall \bar{o} Q_X.\Theta(\bar{o}) \rightarrow \bigwedge_{f_i(\pi_1,\ldots,\pi_{i_n},d^i) \in Tran_{EFE^*}(\phi)} Occlude(\rho, f_i(\pi_1,\ldots,\pi_{i_n},d^i)).$$

The sub-formula,

$$\bigwedge_{f_i(\pi_1,\ldots,\pi_{i_n},d^i) \in Tran_{EFE^*}(\phi)} Occlude(\rho, f_i(\pi_1,\ldots,\pi_{i_n},d^i)),$$

can be transformed to the equivalent form,

$$\forall t \forall z.t = \rho \wedge (\bigvee_{f_i(\pi_1,\ldots,\pi_{i_n},d^i) \in Tran_{EFE^*}(\phi)} z = f_i(\pi_1,\ldots,\pi_{i_n},d^i)) \rightarrow Occlude(t,z).$$

Let

$$\Phi = \forall t_1 \forall \bar{o} Q_X.(\Theta(\bar{o}) \wedge t = \rho \wedge (\bigvee_{f_i(\pi_1,\ldots,\pi_{i_n},d^i) \in Tran_{EFE^*}(\phi)} z = f_i(\pi_1,\ldots,\pi_{i_n},d^i))).$$

It is easily observable that ϕ is has the form

$$\Gamma_1 \wedge \forall t \forall z.\Phi \rightarrow Occlude(t,z).$$

Lemma A.6
Let $\Psi = \bigwedge_{i=1}^{k} Tran_{CC}(\phi_i)$ be a formula in $\mathcal{L}(FL)$ which is the result of translating a conjunction $\bigwedge_{i=1}^{k} \phi_i$ of causal constraint statements in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$. Ψ can be transformed to the logically equivalent form,

$$(\bigwedge_{i=1}^{k} \Gamma_i) \wedge \forall t \forall z.(\bigvee_{i=1}^{k} \Phi_i) \rightarrow Occlude(t,z).$$

Proof Straightforward. It is easily observable that this is the case if each of the $Tran_{CC}(\phi_i)$ is put into the form shown in Proposition A.6.

Remark A.13 *Lemma A.6 provides a very nice logical characterization of the set of causal constraints in an action scenario description. $\bigwedge_{i=1}^{k} \Gamma_i$ is the set of causal constraints without the directional dependencies. $\forall t \forall z.(\bigvee_{i=1}^{k} \Phi_i) \rightarrow Occlude(t,z)$ provides a succinct characterization of the dependency information, where each Φ_i provides the sufficient conditions for "firing" a dependency policy for the ith causal constraint in the scenario.*

Theorem A.2
Let $\Psi = \bigwedge_{i=1}^{k} Tran_{CC}(\phi_i)$ be a formula in $\mathcal{L}(FL)$ which is the result of translating a conjunction $\bigwedge_{i=1}^{k} \phi_i$ of causal constraint statements in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$.

The circumscription of Ψ with $Occlude$ minimized and all other predicates fixed $Circ_{SO}(\Psi; Occlude)$, is equivalent to the following first-order formula

$$\Gamma \wedge \forall t \forall z.(\Delta \equiv Occlude(t,z)),$$

where

$$\Gamma = \bigwedge_{i=1}^{k} \Gamma_i,$$

and

$$\Delta = \bigvee_{i=1}^{k} \Phi_i.$$

Proof By Lemma A.6, Ψ can be transformed into the form

$$(\bigwedge_{i=1}^{k} \Gamma_i) \wedge \forall t \forall z.(\bigvee_{i=1}^{k} \Phi_i) \rightarrow Occlude(t, z).$$

Theorem B.1 by Lifschitz allows us to factor out $(\bigwedge_{i=1}^{k} \Gamma_i)$ when circumscribing Ψ with $Occlude$ minimized and all other predicates fixed, since it contains no occurrences of $Occlude$. Theorem B.2 by Lifschitz shows us that

$$Circ_{SO}(\{\forall t \forall z.(\bigvee_{i=1}^{k} \Phi_i) \rightarrow Occlude(t, z)\}; Occlude) \equiv \Gamma \wedge \forall t, z.(\Delta \equiv Occlude(t, z)).$$

The proof of Lemma A.6 can be used to show that Γ and Δ have the values stated in the theorem.

A.4 Acausal Constraint Statements in $\mathcal{L}(SD)$

Definition A.36 (Acausal Constraint Statement in $\mathcal{L}(SD)$)
An acausal constraint statement in $\mathcal{L}(SD)$ is a logic formula of the form $\forall t.\gamma$, where t is an elementary time-point expression of type temporal variable and any elementary temporal expressions in γ are functions of t.

We will use the label *acc* to label acausal constraint statements in action scenarios.

A.4.1 Reducing Acausal Constraint Statements in $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$

Definition A.37 ($Tran_{AC}$)
Let ϕ be an acausal constraint statement.

$$Tran_{AC}(\phi) = Tran_{LF}(\phi).$$

A.5 Dynamic Fluent Statements in $\mathcal{L}(SD)$

It is often desirable to allow formulas in an action scenario which are not subject to inertia assumptions anywhere on the time line or in restricted parts of the time line. Dynamic and momentary fluents were first discussed in Lifschitz [18].

Definition A.38 (Dynamic Fluent Statements)
A *dynamic fluent statement* is an interval fixed fluent formula (IFFF) of the form $[\tau, \tau']Q_{\bar{o}_2}.\beta(\bar{o}_2)$ where $Q_{\bar{o}_2}$ is a finite sequence of quantifiers binding the variables $\bar{o}_2$, and $\beta(\bar{o}_2)$ is a quantifier free fluent formula.

We will use the label *df* to label dynamic fluent statements in action scenarios.

A.5.1 Reducing Dynamic Fluent Statements in $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$

Definition A.39 ($Tran_{DF}$)
Let $\phi = [\tau, \tau']Q_{\bar{o}_2}.\beta(\bar{o}_2)$ be a dynamic fluent statement.

$$Tran_{DF}(\phi) = \forall t.Tran_{ETE}(\tau) \leq t \leq Tran_{ETE}(\tau') \rightarrow \\ [Tran_{LF}([t]Q_{\bar{o}_2}.\beta(\bar{o}_2)) \wedge Tran_X([t]Q_{\bar{o}_2}.\beta(\bar{o}_2))]. \tag{14}$$

Remark A.14 *In the case where a fluent formula should be dynamic at all time-points, the following interval fixed formula can be used: $[0, \infty]\beta$. In the case where a fluent formula is momentarily dynamic, the following interval fixed formula can be used: $[\tau, \tau]\beta$.*

A.6 Observation Statements in $\mathcal{L}(SD)$

Definition A.40 (Observation Statement in $\mathcal{L}(SD)$)
An observation statement in $\mathcal{L}(SD)$ is one of the following:

- A fixed fluent formula $[\tau]\gamma$, where τ is an elementary time-point expression containing no temporal variables.
- A time-point formula.
- An object formula.

We will use the label *obs* to label observation statements in action scenarios.

Definition A.41 (Fixed Observation Statement in $\mathcal{L}(SD)$)
A fixed observation statement in $\mathcal{L}(SD)$ is the result of translating an observation statement, obs $[\tau]\gamma$, of type fixed fluent formula into the following form,

$$[\tau]Observe(\gamma).$$

In the case where the observation statement is a time-point or object formula, obs γ, it is left unchanged.

Remark A.15 *Note that this definition of observation statements diverges somewhat from Sandewall's in that it is more restricted. Sandewall defines observation statements as logic formulas, although in practice, it appears that it only makes sense to use a subset of logic formulas. We may have to modify this definition in the future, but for current purposes it appears to do the job.*

A.6.1 Reducing Observation Statements in $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$

Definition A.42 ($Tran_{X-OBS}$)
Let $\phi = [\tau]Observe(\gamma)$ be a fixed observation statement, where γ is a fluent formula.

$$Tran_{X-OBS}(\phi) = Q_X. \bigwedge_{f_i(\pi_1,\ldots,\pi_{i_n},d^i) \in Tran_{EFE^*}(\gamma)} Observe(\rho, f_i(\pi_1, \ldots, \pi_{i_n}, d^i)),$$

where,

1. $\rho = Tran_{ETE}(\tau)$ is the translation of the ETE τ into $\mathcal{L}(FL)$.
2. Q_X is a sequence of universally quantified variables constructed by
   - renaming all variables in γ so they are unique,
   - changing all existential quantifiers in γ to universal quantifiers,

- collecting all universal quantifiers generated in the previous step together with those already existing in γ into a sequence of universal quantifiers which we denote by Q_X,
- and prefixing $\forall \bar{d}$ to Q_X.

We will now provide two different translation operators for observation statements. The first, $Tran_{OBS1}$, is used in PMON and PMON(RCs). The second, $Tran_{OBS2}$, is used to translate a fixed observation statement when extending both PMON and PMON(RCs) to deal with external observations.

Definition A.43 ($Tran_{OBS1}$)
Let ϕ be an observation statement.

$$Tran_{OBS1}(\phi) = Tran_{LF}(\phi).$$

Definition A.44 ($Tran_{OBS2}$)
Let ϕ be a fixed observation statement. If $\phi = [\tau]Observe(\gamma)$ then

$$Tran_{OBS2}(\phi) = Tran_{OBS1}([\tau]\gamma) \wedge Tran_{X-OBS}(\phi).$$

If ϕ is a time-point or object formula then

$$Tran_{OBS2}(\phi) = Tran_{OBS1}(\phi).$$

A.7 Action Representation in $\mathcal{L}(SD)$

To represent actions in a scenario description, we introduce a set of *action symbols*. These will usually be denoted by everyday English words such as $Load$, $Fire$, etc., and may contain elementary object or value expressions as arguments.

Definition A.45 (Action Similarity Type)
Let $\nu = \langle \sigma, \mathcal{O}, \mathcal{V}, \mathcal{D} \rangle$ be a vocabulary in $\mathcal{L}(SD)$, where

$$\mathcal{O} = \langle \mathcal{O}_1, \ldots, \mathcal{O}_l \rangle,$$

$$\mathcal{V} = \langle \mathcal{V}_1, \ldots, \mathcal{V}_p \rangle,$$

$$\sigma = \{ f_1^{i_1} : \mathcal{D}_1, f_2^{i_2} : \mathcal{D}_2, \ldots f_n^{i_m} : \mathcal{D}_n \},$$

and

$$\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n \rangle.$$

An *action similarity type* π for $\mathcal{L}(SD)$ is a tuple

$$\{ A_1^{i_1}, A_2^{i_2}, \ldots A_m^{i_m} \},$$

where $i_k \geq 0$ for each k and each $A_k^{i_k}$ is an Action symbol with i_k arguments. An object or value sort $\mathcal{D}_j$, is associated with each of the i_k arguments of $A_k^{i_k}$.

Definition A.46 (Action Occurrence Statements)
An *action occurrence statement* is any expression of the form

- $[\tau, \tau']A_i^0$, or
- $[\tau, \tau']A_i^{i_k}(\omega_1, \ldots, \omega_{i_k})$,

where $A_i^0, A_i^{i_k}$ are action symbols, $\omega_1, \ldots, \omega_{i_k}$, are elementary value or object expressions of type constant or name symbol, and τ and τ' are elementary time-point expressions containing no temporal variables.

Definition A.47 (Action Law Schema)
An *action law schema* is any expression of the form

- $[t_1, t_2]A_i^0 \rightsquigarrow \Delta$, or
- $[t_1, t_2]A_i^{i_k}(x_1, \ldots, x_{i_k}) \rightsquigarrow \Delta$,

where $A_i^0, A_i^{i_k}$ are action symbols, $x_1, \ldots, x_{i_k}$, are elementary value or object expressions of type object variable or value variable, t_1 and t_2 are elementary time-point expressions of type temporal variable, and Δ is a conjunction of conditional reassignment formulas $\delta_1 \wedge \ldots \wedge \delta_j$, where each δ_i has the following form,

$$\forall \bar{t} \forall \bar{o}.([t_1]\phi(\bar{o}) \rightarrow (([t_1, t_2]\theta(\bar{o})) \wedge \psi(\bar{t}, \bar{o}))),$$

where t_1, t_2, and $x_1, \ldots, x_{i_k}$ may only occur free in Δ.

We say that an action occurrence statement *corresponds* to an action law schema if the same action symbol occurs in both expressions.

Definition A.48 (Result of an Action Occurrence)
Let $[\tau, \tau']\mathbf{A}$ be an action occurrence statement corresponding to an action schema $[s, t]\mathbf{A} \rightsquigarrow \Delta$. The *result* of the action occurrence wrt the action schema is the logic formula $\Delta(t_1/\tau, t_2/\tau', x_1/\omega_1, \ldots x_{i_k}/\omega_{i_k})$, where $\Delta(t_1/\tau, t_2/\tau', x_1/\omega_1, \ldots x_{i_k}/\omega_{i_k})$ is obtained from Δ by substituting the ETE's τ and τ for the variables s and t and the EOE's or EVE's $\omega_1, \ldots, \omega_{i_k}$ for the variables $x_1, \ldots, x_{i_k}$.

Example A.1 *For instance, the result of the action occurrence* $[6, \mathsf{t}']Load(\mathsf{g1})$ *wrt the action schema*

$$[t_1, t_2]Load(x_1) \rightsquigarrow ([t_1, t_2]loaded(x_1) := T) \wedge (t_2 = t_1 + 3)$$

is the logic formula,

$$([6, \mathsf{t}']loaded(\mathsf{g1}) := T) \wedge (\mathsf{t}' = 6 + 3).$$

Definition A.49 ($Tran_{OCC}$)
Let $\phi = [\tau, \tau']A_i^{i_k}(\omega_1, \ldots, \omega_{i_k})$ be an action occurrence statement.

$$Tran_{OCC}(\phi) = Occurs(\rho, \rho', \mathsf{A}_i(\pi_1, \ldots . \pi_{i_k})),$$

where $\rho = Tran_{ETE}(\tau)$, $\rho' = Tran_{ETE}(\tau')$, $\pi_j = Tran_{E(OV)E}(\omega_j)$ for each ω_j, $1 \leq j \leq i_k$, and A_i is the function symbol in $\mathcal{L}(FL)$ associated with $A_i^{i_k}$ in $\mathcal{L}(SD)$.

A.8 Action Scenario Descriptions in $\mathcal{L}(SD)$

Definition A.50 (Action Scenario Description)
An *action scenario description* consists of a finite set of observation statements, action occurrence statements, action law schemas, acausal constraint statements, and causal constraint statements. All statements in an action scenario will be prefixed with the labels, "obs", "occ", "acs", "acc", "cc" and "df", for observation statements, action occurrence statements, action law schemas, acausal constraint, causal constraint and dynamic fluent statements, respectively. In addition, it is assumed that for each action occurrence statement in an action scenario, there is exactly one corresponding action law schema.

Example A.2 The following is the Yale shooting scenario (below a and l are feature symbols standing for *alive* and *loaded*, respectively, while *Load* and *Fire* are action symbols).

obs1 [0] $a \wedge \neg l$
occ1 [2,4] $Load$
occ2 [5,6] $Fire$
acs1 $[t_1, t_2]$ $Load \rightsquigarrow [t_1, t_2]$ $l := T$
acs2 $[t_1, t_2]$ $Fire \rightsquigarrow ([t_1]$ $l \rightarrow [t_1, t_2](a := F \wedge l := F))$.

Definition A.51 (Expanded Action Scenario Description)
Let Υ be an action scenario description. An *expanded action scenario description* Υ' for Υ consists of the finite set of labeled (fixed) observation statements in Υ together with the set of schedule statements obtained from Υ by adding, for each action occurrence statement in Υ, its result wrt the corresponding action law schemas in Υ. All schedule statements in an expanded action scenario will be prefixed with the label "scd".

Example A.3 The expanded action scenario description for the action scenario in Example A.2, is,

obs1 [0] $Observe(a \wedge \neg l)$
occ1 [2,4] $Load$
scd1 [2,4] $l := T$
occ2 [5,6] $Fire$
scd2 [5] $l \rightarrow$ [5,6]$(a := F \wedge l := F)$.

Example A.4 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario in Example A.2, is,

obs1 $Holds(0, \mathsf{a}) \wedge Observe(0, \mathsf{a}) \wedge \neg Holds(0, \mathsf{l}) \wedge Observe(0, \mathsf{l})$
occ1 $Occurs(2, 4, \mathsf{Load})$
scd1 $Holds(4, \mathsf{l}) \wedge (\forall t.2 < t \leq 4 \rightarrow Occlude(t, \mathsf{a}))$
occ2 $Occurs(5, 6, \mathsf{Fire})$
scd2 $Holds(5, \mathsf{l}) \rightarrow [(\neg Holds(6, \mathsf{a}) \wedge \neg Holds(6, \mathsf{l}))$
$\wedge Occlude(6, \mathsf{a}) \wedge Occlude(6, \mathsf{l})]$.

Theorem A.3
Let Υ be an action scenario description. There is an effective algorithm that transforms Υ into a finite set Γ of labeled wffs in $\mathcal{L}(FL)$.

Proof

Let Υ' be the expanded action scenario description generated via application of Definition (A.48) to Υ. Υ' only contains observation, occurs, schedule, acausal and causal constraint statements which with the exception of occurs statements, are all logic formulas or conditional reassignment formulas in $\mathcal{L}(SD)$. Each statement can be translated into a wff in $\mathcal{L}(FL)$ using the translation operators $Tran_{LF}$, $Tran_{OBS}$, $Tran_{CR}$, $Tran_{CC}$, $Tran_{DF}$, and the occurs statements can be translated using $Tran_{OCC}$.

A.8.1 Some Useful Notation

For each of the partitions in an expanded scenario description Υ', we use Γ_{OBS}, Γ_{OCC},Γ_{SCD}, Γ_{CC}, Γ_{AC} and Γ_{DF}, to denote the observation, occurrence, schedule, causal constraint, acausal constraint and dynamic fluent formulas in $\mathcal{L}(FL)$ which are the result of translating a scenario description Υ into $\mathcal{L}(FL)$. Each Γ_x may denote either a set or conjunction of formulas.

Note that that the set of *Occur* and *Observe* atoms in $\mathcal{L}(FL)$ are not used in either PMON or PMON(RCs). We will use them in a number of extensions and include them in the translations for later use.

B The PMON Family of Logics

In this appendix, we provide definitions for the PMON family of logics and a number of reduction theorems.

B.1 Circumscription

We will assume familiarity with both second-order and pointwise circumscription [16] and use the notation in Lukaszewicz [19]. Briefly,

Definition B.1 (Second-Order Circumscription)
Let $\bar{P}$ be a tuple of distinct predicate constants, $\bar{S}$ be a tuple of distinct function and/or predicate constants disjoint from $\bar{P}$, and let $\Gamma(\bar{P}, \bar{S})$ be a theory. The *second-order circumscription* of $\bar{P}$ in $\Gamma(\bar{P}, \bar{S})$ with variable $\bar{S}$, written $Circ_{SO}(\Gamma; \bar{P}; \bar{S})$, is the sentence

$$\Gamma(\bar{P}, \bar{S}) \wedge \forall \bar{\Phi}, \bar{\Psi}. \neg[\Gamma(\bar{\Phi}, \bar{\Psi}) \wedge \bar{\Phi} < \bar{P}]$$

where $\bar{\Phi}$ and $\bar{\Psi}$ are tuples of variables similar to $\bar{P}$ and $\bar{S}$, respectively.

Observe that this can be rewritten as,

$$\Gamma(\bar{P}, \bar{S}) \wedge \forall \bar{\Phi}, \bar{\Psi}. [\Gamma(\bar{\Phi}, \bar{\Psi}) \wedge [\bar{\Phi} \leq \bar{P}] \rightarrow [\bar{P} \leq \bar{\Phi}]]$$

where $\bar{U} \leq \bar{W}$ is equivalent to $\bigwedge_{i=1}^{n} \forall \bar{x}. [U_i(\bar{x}) \rightarrow W_i(\bar{x})]$.

Definition B.2 (Pointwise Circumscription)
Let $\Gamma(P, \bar{S})$ be a theory, where P is a predicate constant and $\bar{S}$ is a tuple of distinct function and/or predicate constants not containing P. The *pointwise circumscription of P in $\Gamma(P, \bar{S})$, with variable $\bar{S}$*, denoted by $Circ_{PW}(\Gamma; P; \bar{S})$, is the sentence,

$$\Gamma(P; \bar{S}) \wedge \forall \bar{x} \forall \bar{\Phi}. \neg[P(\bar{x}) \wedge \Gamma(\lambda \bar{y}. (P(\bar{y}) \wedge \bar{y} \neq \bar{x}), \bar{\Phi})].$$

The following two theorems were proven by V. Lifschitz.

Theorem B.1 ([17],p311.)
For any sentence B not containing P,Z,

$$Circ[A(P, Z) \wedge B; P; Z] \equiv Circ[A(P, Z); P; Z] \wedge B.$$

Theorem B.2 ([17],p309.)
If $F(\overline{x})$ does not contain P, then the circumscription

$$Circ[\forall \overline{x} F(\overline{x}) \rightarrow P(\overline{x}); P] \equiv \forall \overline{x} F(\overline{x}) \equiv P(\overline{x}).$$

B.2 The PMON Family of Logics and their Circumscription Policies

B.2.1 The Nochange Axiom

The Nochange axiom is used as a filtering device which filters out spurious change not justified by any of the partitions in an action scenario. The first axiom Γ_{NCG} is used in PMON and PMON(RCs). The second, Γ_{SCD^*}, is used when extending PMON and PMON(RCs) with external observations.

$$\Gamma_{NCG} = \{ \quad \forall t, f. (\neg Occlude(t+1, f) \rightarrow Holds(t, f) \equiv Holds(t+1, f))\}. \tag{15}$$

$$\begin{aligned} \Gamma_{NCG^*} = \{ \quad & \forall t, f. (\neg Occlude(t+1, f) \wedge \neg Occlude_{obs}(t+1, f)) \\ & \rightarrow Holds(t, f) \equiv Holds(t+1, f)\}. \end{aligned} \tag{16}$$

B.2.2 PMON

The original PMON logic is assessed correct for the $\mathcal{K} - IA$ class of action scenarios. We will assume that neither observation statements nor action occurrences are reified in this logic. This implies using $Tran_{OBS1}$ when translating observation statements and translating action occurrence statements to just schedule statements disregarding the occurrence translation.

Before stating the circumscription policy, we assume the following for each legal $\mathcal{K} - IA$ action scenario Υ:

- **A duration condition** – For each action occurrence statement $[\tau_1, \tau_2]A$, we assume additional observation statements

$$obs_i : \tau_1 < \tau_2.$$

- **A sequentiality condition** – For any two action occurrence statements $occ1$ $[\tau_1, \tau_2]A_1$ and $occ2$ $[\tau_3, \tau_4]A_2$, we assume additional observation statements

$$obs_i \ \ (\tau_2 < \tau_3) \vee (\tau_4 < \tau_1).$$

We will use the following circumscription policy:

Definition B.3 (PMON Circumscription)

The *pmon circumscription* of the action scenario description Υ is

$$\Gamma_{FA} \wedge \Gamma_{NCG} \wedge \Gamma_{OBS} \wedge Circ_{SO}(\Gamma_{SCD}(Occlude); \langle Occlude \rangle),$$

where $Circ_{SO}(\Gamma_{SCD}(Occlude); \langle Occlude \rangle)$ is equivalent to

$$\Gamma_{SCD} \wedge \forall \Phi. \neg [\Gamma_{SCD}(\Phi) \wedge \Phi < Occlude].$$

Definition B.4 (PMON Entailment)
A formula α is said to be *PMON-entailed by* the action scenario description Υ if

$$\Gamma_{FA} \wedge \Gamma_{NCG} \wedge \Gamma_{OBS} \wedge Circ_{SO}(\Gamma_{SCD}(Occlude); \langle Occlude \rangle) \models \alpha.$$

Corollary B.1
Let $\Gamma_{SCD}(Occlude)$ be the result of translating a conjunction of conjunctions of conditional reassignment formulas in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$. Then $Circ_{SO}(\Gamma_{SCD}(Occlude); \langle Occlude \rangle)$ is reducible to a first-order formula with the following form,

$$\Gamma \wedge \forall t \forall z. (\Delta \leftrightarrow Occlude(t, z)).$$

Proof Straightforward application of Theorem A.1.

Corollary B.2
Let Γ be the $\mathcal{L}(FL)$ formula denoting the $PMON$ circumscription of the action scenario Υ. Γ is reducible to a logically equivalent first-order formula.

B.2.3 PMON(RCs): Extending PMON with Causal Constraint Statements

The original PMON(RCs) logic is a generalization of PMON which permits the specification of causal and acausal constraints which are used in specifying the indirect effects of actions. It is assessed correct for the $\mathcal{K}-IA$ class of action scenarios when neither casual nor acausal constraints are used. As for PMON we will assume that neither observation statements nor action occurrences are reified in this logic. This implies using $Tran_{OBS1}$ when translating observation statements and translating action occurrence statements to just schedule statements disregarding the occurrence translation. We will use the following circumscription policy:

Definition B.5 (PMON(RCs) Circumscription)

The *PMON(RCs) circumscription* of the action scenario description Υ is

$$\Gamma_{FA} \wedge \Gamma_{NCG} \wedge \Gamma_{OBS} \wedge \Gamma_{AC} \wedge Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude); \langle Occlude \rangle),$$

where $Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude); \langle Occlude \rangle)$ is equivalent to

$$\Gamma_{SCD} \wedge \Gamma_{CC} \wedge \forall \Phi . \neg [\Gamma_{SCD}(\Phi) \wedge \Gamma_{CC}(\Phi) \wedge \Phi < Occlude].$$

Definition B.6 (PMON(RCs) Entailment)
A formula α is said to be *PMON(RCs)-entailed by* the action scenario description Υ if

$$\Gamma_{FA} \wedge \Gamma_{NCG} \wedge \Gamma_{OBS} \wedge \Gamma_{AC} \wedge Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude); \langle Occlude \rangle) \models \alpha.$$

Theorem B.3
Let $\Gamma_{SCD}(Occlude)$ be the result of translating a conjunction of conjunctions of conditional reassignment formulas in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$ and $\Gamma_{CC}(Occlude)$ be the result of translating a conjunction of causal constraint statements in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$. Then $Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude); \langle Occlude \rangle)$ is reducible to a first-order formula with the following form,

$$\Gamma_1 \wedge \Gamma_2 \wedge \forall t \forall z . (\Delta_1 \vee \Delta_2 \leftrightarrow Occlude(t, z)).$$

Proof By Lemma A.3, $\Gamma_{SCD}(Occlude)$ can be transformed into the logically equivalent form,

$$\Gamma_1 \wedge \forall t \forall z . (\Delta_1 \rightarrow Occlude(t, z)).$$

By Lemma A.6, $\Gamma_{CC}(Occlude)$ can be transformed into the logically equivalent form,

$$\Gamma_2 \wedge \forall t \forall z . (\Delta_2 \rightarrow Occlude(t, z)).$$

$\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude)$ is equivalent to

$$\Gamma_1 \wedge \Gamma_2 \wedge \forall t \forall z . (\Delta_1 \vee \Delta_2 \rightarrow Occlude(t, z)).$$

Theorem B.1 by Lifschitz allows us to factor out $\Gamma_1 \wedge \Gamma_2$ when circumscribing $\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude)$ with $Occlude$ minimized and all other predicates fixed, since it contains no occurrences of $Occlude$. Theorem B.2 by Lifschitz shows us that

$$Circ_{SO}(\{\forall t \forall z . (\Delta_1 \vee \Delta_2 \rightarrow Occlude(t, z))\}; Occlude) \equiv \forall t, z . (\Delta_1 \vee \Delta_2 \equiv Occlude(t, z)).$$

Corollary B.3
Let Γ be the $\mathcal{L}(FL)$ formula denoting the $PMON(RCs)$ circumscription of the action scenario Υ. Γ is reducible to a logically equivalent first-order formula.

B.2.4 PMON$^+$: Extending PMON with Dynamic Fluent Statements

In order to extend PMON or PMON(RCs) with dynamic fluent statements, we simply have to minimize the dynamic fluent formulas Γ_{DF} together with Γ_{SCD} and Γ_{CC}. Since there are only positive occurrences of $Occlude$ in Γ_{DF}, the first-order reducibility results also apply for this extension.

Definition B.7 (PMON$^+$ Circumscription)

The *PMON$^+$ circumscription* of the action scenario description Υ is

$$\Gamma_{FA} \wedge \Gamma_{NCG} \wedge \Gamma_{OBS} \wedge \Gamma_{AC} \wedge Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude) \wedge \Gamma_{DF}(Occlude); \langle Occlude \rangle),$$

where $Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude) \wedge \Gamma_{DF}(Occlude); \langle Occlude \rangle)$ is equivalent to

$$\Gamma_{SCD} \wedge \Gamma_{CC} \wedge \forall \Phi. \neg [\Gamma_{SCD}(\Phi) \wedge \Gamma_{CC}(\Phi) \wedge \Gamma_{DF}(\Phi) \wedge \Phi < Occlude].$$

Definition B.8 (PMON$^+$ Entailment)
A formula α is said to be *PMON$^+$-entailed by* the action scenario description Υ if

$$\Gamma_{FA} \wedge \Gamma_{NCG} \wedge \Gamma_{OBS} \wedge \Gamma_{AC} \wedge Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude) \wedge \Gamma_{DF}(Occlude); \langle Occlude \rangle) \models \alpha.$$

Theorem B.4
Let $\Gamma_{SCD}(Occlude)$ be the result of translating a conjunction of conjunctions of conditional reassignment formulas in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$ and $\Gamma_{CC}(Occlude)$ be the result of translating a conjunction of causal constraint statements in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$ and $\Gamma_{DF}(Occlude)$ be the result of translating a conjunction of dynamic fluent statements in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$. Then $Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude) \wedge \Gamma_{DF}(Occlude); \langle Occlude \rangle)$ is reducible to a first-order formula with the following form,

$$\Gamma_1 \wedge \Gamma_2 \wedge \Gamma_3 \wedge \forall t \forall z.(\Delta_1 \vee \Delta_2 \vee \Delta_3 \leftrightarrow Occlude(t,z)).$$

Proof By Lemma A.3, $\Gamma_{SCD}(Occlude)$ can be transformed into the logically equivalent form,

$$\Gamma_1 \wedge \forall t \forall z.(\Delta_1 \rightarrow Occlude(t,z)).$$

By Lemma A.6, $\Gamma_{CC}(Occlude)$ can be transformed into the logically equivalent form,

$$\Gamma_2 \wedge \forall t \forall z.(\Delta_2 \rightarrow Occlude(t,z)).$$

By a method similar to that used in Lemma A.6, $\Gamma_{DF}(Occlude)$ can be transformed into the logically equivalent form,

$$\Gamma_3 \wedge \forall t \forall z.(\Delta_3 \rightarrow Occlude(t,z)).$$

$\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude) \wedge \Gamma_{DF}(Occlude)$ is equivalent to

$$\Gamma_1 \wedge \Gamma_2 \wedge \Gamma_3 \wedge \forall t \forall z.(\Delta_1 \vee \Delta_2 \vee \Delta_3 \rightarrow Occlude(t,z)).$$

Theorem B.1 by Lifschitz allows us to factor out $\Gamma_1 \wedge \Gamma_2 \wedge \Gamma_3$ when circumscribing $\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude) \wedge \Gamma_{DF}(Occlude)$ with $Occlude$ minimized and all other predicates fixed, since it contains no occurrences of $Occlude$. Theorem B.2 by Lifschitz shows us that

$$Circ_{SO}(\{\forall t \forall z.(\Delta_1 \vee \Delta_2 \vee \Delta_3 \rightarrow Occlude(t,z))\}; Occlude) \equiv \forall t, z.(\Delta_1 \vee \Delta_2 \vee \Delta_3 \equiv Occlude(t,z)).$$

Corollary B.4
Let Γ be the $\mathcal{L}(FL)$ formula denoting the $PMON^+$ circumscription of the action scenario Υ. Γ is reducible to a logically equivalent first-order formula.

C Benchmark Examples

This collection of scenario descriptions provides a test suite for informal evaluation of logics of action and change. Many of the examples have appeared previously and are due to other authors. Some are new and have not previously been published. The examples include and subsume those previously collected in Sandewall [27], which were used as representative examples of chronicle completion for the $\mathcal{K} - IA$ class. Most of the new examples deal with problems which arise due to ramification. In some of the examples, we also provide the definition of the *Occlude* predicate derived via the circumscription reduction. All examples have been verified to provide the *correct* conclusions using PMON and PMON(RCs).

C.1 Examples within the $\mathcal{K} - IA$ class

These examples primarily test problems associated with strict inertia, nondeterminism, partial specification of initial states, timing of actions and duration of actions. Note that in the benchmarks, we omit the sequentiality and duration conditions mentioned in Section B.2.2 for notational economy, assuming that they are always added to the theory in question.

C.1.1 Yale Shooting Scenario

The intended conclusion is that at the end of firing, the turkey is no longer alive. This is a temporal prediction reasoning task.

Example C.1 (Yale Shooting Scenario, Hanks and McDermott [9])

Action Symbols: $Load$, $Fire$.
Feature Symbols: $alive$, $loaded$

obs1 [0] $alive \wedge \neg loaded$
occ1 [2,4] $Load$
occ2 [5,6] $Fire$
acs1 $[t_1, t_2]$ $Load \rightsquigarrow [t_1, t_2]$ $loaded := T$
acs2 $[t_1, t_2]$ $Fire \rightsquigarrow ([t_1]$ $loaded \rightarrow [t_1,t_2](alive := F \wedge loaded := F))$.

Example C.2 The expanded action scenario description is,

obs1 [0] $Observe(alive \wedge \neg loaded)$
occ1 [2,4] $Load$
scd1 [2,4] $loaded := T$
occ2 [5,6] $Fire$
scd2 [5] $loaded \rightarrow$ [5,6]$(alive := F \wedge loaded := F)$.

Example C.3 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

obs1 $Holds(0, \mathsf{alive}) \wedge Observe(0, \mathsf{alive}) \wedge \neg Holds(0, \mathsf{loaded}) \wedge Observe(0, \mathsf{loaded})$
occ1 $Occurs(2, 4, \mathsf{Load})$
scd1 $Holds(4, \mathsf{loaded}) \wedge (\forall t.2 < t \leq 4 \rightarrow Occlude(t, \mathsf{alive}))$
occ2 $Occurs(5, 6, \mathsf{Fire})$
scd2 $Holds(5, \mathsf{loaded}) \rightarrow [(\neg Holds(6, \mathsf{alive}) \wedge \neg Holds(6, \mathsf{loaded}))$
$\wedge Occlude(6, \mathsf{alive}) \wedge Occlude(6, \mathsf{loaded})]$.

Circumscribing Γ_{SCD} with *Occlude* minimized and *Holds* fixed results in the following definition:

$$\forall t, f. Occlude(t, f) \equiv$$
$$[(2 < t \leq 4 \wedge f = \mathsf{loaded}) \vee (Holds(5, \mathsf{loaded}) \wedge t = 6 \wedge (f = \mathsf{alive} \vee f = \mathsf{loaded})]$$

C.1.2 Stanford Murder Mystery Scenario

The intended conclusions are that the gun is loaded at the initial time-point and the turkey is not alive at the end of the firing action. This is a temporal postdiction reasoning task.

Example C.4 (Stanford Murder Mystery, Ginsberg and Baker [1])

Action Symbols: $Fire$.
Feature Symbols: $alive$, $loaded$

obs1 [0] $alive$
occ1 [5,6] $Fire$
acs1 $[t_1, t_2]$ $Fire \rightsquigarrow ([t_1]\ loaded \rightarrow [t_1,t_2](alive := F \wedge loaded := F))$.
obs2 [8] $\neg alive$

Example C.5 The expanded action scenario description is,

obs1 [0] $Observe(alive)$
occ1 [5,6] $Fire$
scd1 [5] $loaded \rightarrow$ [5,6]$(alive := F \wedge loaded := F)$.
obs2 [8] $Observe(\neg alive)$

Example C.6 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

obs1 $Holds(0, \mathsf{alive}) \wedge Observe(0, \mathsf{alive})$
occ1 $Occurs(5, 6, \mathsf{Fire})$
scd1 $Holds(5, \mathsf{loaded}) \rightarrow [(\neg Holds(6, \mathsf{alive}) \wedge \neg Holds(6, \mathsf{loaded}))$
$\wedge Occlude(6, \mathsf{alive}) \wedge Occlude(6, \mathsf{loaded})]$.
obs2 $\neg Holds(8, \mathsf{alive}) \wedge Observe(8, \mathsf{alive})$

Circumscribing Γ_{SCD} with $Occlude$ minimized and $Holds$ fixed results in the following definition:

$$\forall t, f. Occlude(t, f) \equiv$$
$$[(Holds(5, \mathsf{loaded}) \wedge t = 6 \wedge (f = \mathsf{alive} \vee f = \mathsf{loaded})]$$

C.1.3 The Red and Yellow Bus Scenario

This is a standard example of a postdiction problem, where filtering and partitioning of statements solve the problem which arises when using nondeterministic actions. The intended result is that after buying a ticket, the ticket holder should not be on any bus. Kartha shows that Baker's approach has unintended results. Shannahan claims that this is a problem for state based approaches which use the situation calculus and additionally a problem for narrative based approaches. He solves the problem by complicating the narrative with additional disjunctive events, one for each disjunction in an action, together with additional action-effect laws for each of the new events. Our approach provides the intended model without any need for introducing additional events or action-effect laws to deal with non-deterministic actions. There is one preferred model and in that, one is not on the red bus until after the boarding action.

Example C.7 (The Red and Yellow Bus Scenario, Kartha [14])

Action Symbols: $BuyTicket$, $BoardBus$.
Feature Symbols: $hasTicket$, $onYellow$, $onRed$

obs1 $[0]\ \neg hasTicket$
occ1 [1,2] $BuyTicket$
occ2 [3,4] $BoardBus$
obs2 $[5]\ onRed$
acs1 $[t_1, t_2]\ BuyTicket \rightsquigarrow [t_1]\ \neg hasTicket \rightarrow [t_1, t_2]\ hasTicket := T$
acs2 $[t_1, t_2]\ BoardBus \rightsquigarrow [t_1]\ (hasTicket \wedge \neg onRed \wedge \neg onYellow) \rightarrow$
$[t_1, t_2]\ (onYellow := T \vee onRed := T)$
acc1 $\forall t.[t]\neg(onRed \wedge onYellow)$
acc2 $\forall t.[t]onRed \rightarrow hasTicket$
acc3 $\forall t.[t]onYellow \rightarrow hasTicket$

Example C.8 The expanded action scenario description is,

obs1 $[0]\ Observe(\neg hasTicket)$
occ1 [1,2] $BuyTicket$
occ2 [3,4] $BoardBus$
obs2 $[5]\ Observe(onRed)$
scd1 $[1]\ \neg hasTicket \rightarrow$ [1,2] $hasTicket := T$
scd2 $[3]\ (hasTicket \wedge \neg onRed \wedge \neg onYellow) \rightarrow$
[3,4] $(onYellow := T \vee onRed := T)$
acc1 $\forall t.[t]\neg(onRed \wedge onYellow)$
acc2 $\forall t.[t]onRed \rightarrow hasTicket$
acc3 $\forall t.[t]onYellow \rightarrow hasTicket$

Example C.9 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ is

obs1 $\neg Holds(0, \mathsf{hasTicket}) \wedge Observe(0, \mathsf{hasTicket})$
occ1 $Occurs(1, 2, \mathsf{BuyTicket})$
occ2 $Occurs(3, 4, \mathsf{BoardBus})$
obs2 $Holds(5, \mathsf{onRed}) \wedge Observe(5, \mathsf{onRed})$
scd1 $[\neg Holds(1, \mathsf{hasTicket}) \rightarrow Holds(2, \mathsf{hasTicket})] \wedge$
$[\neg Holds(1, \mathsf{hasTicket}) \rightarrow Occlude(2, \mathsf{hasTicket})]$
scd2 $[Holds(3, \mathsf{hasTicket}) \wedge \neg Holds(3, \mathsf{onRed}) \wedge \neg Holds(3, \mathsf{onYellow}) \rightarrow$
$(Holds(4, \mathsf{onYellow}) \vee Holds(4, \mathsf{onRed}))] \wedge$
$[Holds(3, \mathsf{hasTicket}) \wedge \neg Holds(3, \mathsf{onRed}) \wedge \neg Holds(3, \mathsf{onYellow}) \rightarrow$
$(Occlude(4, \mathsf{onYellow}) \wedge Occlude(4, \mathsf{onRed}))]$
acc1 $\forall t.\neg(Holds(t, \mathsf{onRed}) \wedge Holds(t, \mathsf{onYellow}))$
acc2 $\forall t.Holds(t, \mathsf{onRed}) \rightarrow Holds(t, \mathsf{hasTicket})$
acc3 $\forall t.Holds(t, \mathsf{onYellow}) \rightarrow Hold(t, \mathsf{hasTicket})$

Circumscribing Γ_{SCD} with $Occlude$ minimized and $Holds$ fixed results in the following definition:

$$\forall t, f.Occlude(t, f) \equiv [(\neg Holds(1, \mathsf{hasTicket}) \wedge t = 2 \wedge f = \mathsf{hasTicket}) \vee$$
$$(Holds(3, \mathsf{hasTicket}) \wedge \neg Holds(3, \mathsf{onRed}) \wedge \neg Holds(3, \mathsf{onYellow}) \wedge t = 4 \wedge f = \mathsf{onYellow}) \vee$$
$$(Holds(3, \mathsf{hasTicket}) \wedge \neg Holds(3, \mathsf{onRed}) \wedge \neg Holds(3, \mathsf{onYellow}) \wedge t = 4 \wedge f = \mathsf{onRed})]$$

C.1.4 Hiding Turkey Scenario

The intended conclusion is that at the end of firing, the turkey is deaf and no longer alive, or not deaf and still alive. This is a temporal prediction reasoning task where the initial state is incompletely specified in a way that effects future outcome.

Example C.10 (Hiding Turkey Scenario, Sandewall [27])

Action Symbols: $Load$, $Fire$.
Feature Symbols: $alive$, $loaded$, $deaf$, $hiding$

obs1 [0] $alive \wedge \neg loaded \wedge \neg hiding$
occ1 [2,4] $Load$
occ2 [5,6] $Fire$
acs1 $[t_1, t_2]$ $Load \rightsquigarrow$
$([t_1, t_2]\ loaded := T) \wedge$
$([t_1]\ \neg deaf \rightarrow [t_1,t_2] hiding := T)$
acs2 $[t_1, t_2]$ $Fire \rightsquigarrow$
$([t_1]\ loaded \rightarrow [t_1,t_2] loaded := F)$
$([t_1]\ loaded \wedge \neg hiding \rightarrow [t_1,t_2] alive := F)$.

Example C.11 The expanded action scenario description is,

obs1 [0] $Observe(alive \wedge \neg loaded \wedge \neg hiding)$
occ1 [2,4] $Load$
scd1 $([2,4]\ loaded := T) \wedge$
$([2]\ \neg deaf \rightarrow [2,4] hiding := T)) \wedge$
occ2 [5,6] $Fire$
scd2 $([5]\ loaded \rightarrow [5,6] loaded := F)) \wedge$
$([5]\ loaded \wedge \neg hiding \rightarrow [5,6] alive := F)$.

Example C.12 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

obs1 $Holds(0, \mathsf{alive}) \wedge Observe(0, \mathsf{alive}) \wedge \neg Holds(0, \mathsf{loaded}) \wedge Observe(0, \mathsf{loaded})$
occ1 $Occurs(2, 4, \mathsf{Load})$
scd1 $[Holds(4, \mathsf{loaded}) \wedge (\forall t.2 < t \leq 4 \rightarrow Occlude(t, \mathsf{loaded}))] \wedge$
$[\neg Holds(2, \mathsf{deaf}) \rightarrow (Holds(4, \mathsf{hiding}) \wedge (\forall t.2 < t \leq 4 \rightarrow Occlude(t, \mathsf{hiding})))]$
occ2 $Occurs(5, 6, \mathsf{Fire})$
scd2 $[Holds(5, \mathsf{loaded}) \rightarrow (\neg Holds(6, \mathsf{loaded}) \wedge Occlude(6, \mathsf{loaded}))]$
$[Holds(5, \mathsf{loaded}) \wedge \neg Holds(5, \mathsf{hiding}) \rightarrow (\neg Holds(6, \mathsf{alive}) \wedge Occlude(6, \mathsf{alive}))]$

There are two classes of intended models. In the first, the turkey is initially not deaf and

$$[0, \infty] alive, [0, \infty] \neg deaf,$$

$$[0, 2] \neg loaded, [4, 5] loaded, [6, \infty] \neg loaded,$$

$$[0, 2] \neg hiding, [4, \infty] hiding.$$

In the second class, the turkey is initially deaf and

$$[0, 5] alive, [6, \infty] \neg alive$$

$$[0, \infty] deaf, [0, \infty] \neg hiding,$$

$$[0, 2] \neg loaded, [4, 5] loaded, [6, \infty] \neg loaded.$$

C.1.5 Ferry boat Connection Scenario

This is an example which involves reasoning about a scenario where the the time of occurrence of an action is not completely specified. In this case, if a motorcycle arrives at a ferry boat landing before time 1000 then it will embark and arrive in Jutland at time 1100. If it arrives at the landing after time 1000 then it will remain at the landing. The intended conclusions from the preferred models are that the motorcycle remains at the ferry boat landing or arrives in Jutland at time 1100.[4]

[4] Note that this scenario appears to allow concurrency, but this is not the case because the sequentiality conditions are assumed.

Example C.13 (Ferry boat Connection Scenario, Sandewall [27])

Action Symbols: $Embark$(`vehicle`), $Disembark$(`vehicle`), $Arrive$(`vehicle`).
Feature Symbols: loc : `vehicle` $\rightarrow$ `location`.
Object Symbols: $onboat$: `location`, $atlanding$: `location`, $Jutland$: `location`, $Fyen$: `location`, mc : `vehicle`.

obs1 $990 \leq \mathsf{t_1} \leq 1010$
obs2 $[0]\ loc(mc) \hat{=} Fyen$
occ1 $[\mathsf{t_1} - 1, \mathsf{t_1}]\ Arrive(mc)$
occ2 [1000,1001] $Embark(mc)$
occ3 [1050,1100] $Disembark(mc)$
acs1 $[t_1, t_2]\ Arrive(v) \rightsquigarrow ([t_1, t_2]\ loc(v) := atlanding)$
acs2 $[t_1, t_2]\ Embark(v) \rightsquigarrow ([t_1]\ loc(v) \hat{=} atlanding \rightarrow [t_1,t_2] loc(v) := onboard)$
acs3 $[t_1, t_2]\ Disembark(v) \rightsquigarrow ([t_1]\ loc(v) \hat{=} onboard \rightarrow [t_1,t_2] loc(v) := Jutland$
$\wedge \forall t.t_1 < t \leq t_2 - 1 \rightarrow [t] loc(v) \hat{=} onboard)$

C.1.6 Furniture Assembly Scenario

This is an example of a scenario which includes actions with conditional duration. Initially some furniture is purchased at a store and has to be assembled. If one starts assembling the furniture and the instructions are included then the *Assemble* action will take 20 minutes. If the instructions are not included, the action will take 60 minutes.

Example C.14 (Furniture Assembly Scenario, Sandewall [27])

Action Symbols: $Assemble$
Feature Symbols: $assembled$, $instructions$

obs1 $[0]\ \neg assembled$
occ1 $[100, \mathsf{t_1}]\ Assemble$
acs1 $[t_1, t_2]\ Assemble \rightsquigarrow$
$(([t_1]\ (\neg assembled \wedge instructions) \rightarrow [t_1,t_2] assembled := T \wedge t_2 = t_1 + 20) \wedge$
$([t_1]\ (\neg assembled \wedge \neg instructions) \rightarrow [t_1,t_2] assembled := T \wedge t_2 = t_1 + 60) \wedge$
$([t_1]\ assembled \rightarrow [t_1,t_2] t_2 = t_1 + 1))$

C.2 Examples outside the $\mathcal{K} - IA$ class

These examples deal with problems associated with indirect effects of actions.[5]

C.2.1 Jump in a Lake Scenario

This is an example concerning the persistence of derived effects of actions. In some cases, such as the first example, we expect the indirect effects of actions to persist after the action terminates. In others, such as the second example, we do not. In still other examples, we do expect indirect effects to persist, but only for awhile. The next three examples demonstrate each of these cases.

Example C.15 (Jump in Lake Scenario, Crawford [2])

[5] Note that for any translation of a causal constraint in $\mathcal{L}(FL)$, we aways assume that $t_1 > 0$, although we may sometimes omit this as we do duration and sequentiality conditions.

The intended conclusion in this example is that a person is still wet after jumping into a lake and then getting out.

Action Symbols: $JumpIn$, $GetOut$.
Feature Symbols: wet, $inlake$,

obs1 [0] $\neg inlake \wedge \neg wet$
occ1 [2,3] $JumpIn$
occ2 [5,6] $GetOut$
acs1 $[t_1, t_2]\ JumpIn \rightsquigarrow [t_1, t_2]\ inlake := T$
acs2 $[t_1, t_2]\ GetOut \rightsquigarrow [t_1, t_2]\ inlake := F$
cc1 $\forall t.[t]\ inlake \gg [t]\ wet$

Example C.16 The expanded action scenario description is,

obs1 [0] $Observe(\neg inlake \wedge \neg wet)$
occ1 [2,3] $JumpIn$
scd1 [3] $inlake := T$
occ2 [5,6] $GetOut$
scd2 [6] $inlake := F$
cc1 $\forall t.[t]\ inlake \gg [t]\ wet$

Example C.17 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

obs1 $\neg Holds(0, \mathsf{inlake}) \wedge Observe(0, \mathsf{inlake}) \wedge \neg Holds(0, \mathsf{wet}) \wedge Observe(0, \mathsf{wet})$
occ1 $Occurs(2, 3, \mathsf{JumpIn})$
scd1 $Holds(3, \mathsf{inlake}) \wedge Occlude(3, \mathsf{inlake})$
occ2 $Occurs(5, 6, \mathsf{GetOut})$
scd2 $\neg Holds(6, \mathsf{inlake}) \wedge Occlude(6, \mathsf{inlake})$
occ1 $(\forall t_1.Holds(t_1, \mathsf{inlake}) \rightarrow Holds(t_1, \mathsf{wet})) \wedge$
$\forall t_1.(\neg Holds(t_1 - 1, \mathsf{inlake}) \wedge Holds(t_1, \mathsf{inlake})) \rightarrow Occlude(t_1, \mathsf{wet})$

Example C.18 (Jump in Lake with Hat Scenario, Giunchiglia, et. al. [7])

The intended conclusion in this example is that if a person jumps into a lake with a hat on then the person is still wet after jumping into the lake and then getting out, but the person may not have a hat on any longer.

Action Symbols: $JumpIn$, $GetOut$.
Feature Symbols: wet, $inlake$, hat

obs1 [0] $\neg inlake \wedge \neg wet \wedge hat$
occ1 [2,3] $JumpIn$
occ2 [5,6] $GetOut$
acs1 $[t_1, t_2]\ JumpIn \rightsquigarrow [t_1, t_2]\ inlake := T$
acs2 $[t_1, t_2]\ GetOut \rightsquigarrow [t_1, t_2]\ inlake := F$
cc1 $\forall t.[t]\ inlake \gg [t]\ wet \wedge (hat \vee \neg hat)$

Example C.19 (Extended Jump in Lake Scenario, Doherty)

A more realistic representation of the problem would be that a person is still wet after getting out of the lake, but only for a few minutes because the sun will dry the person. The following delayed effect permits this conclusion and the intended conclusions from the previous two examples.

Action Symbols: $JumpIn$, $GetOut$.
Feature Symbols: wet, $inlake$, hat

obs1 $[0]\ \neg inlake \wedge \neg wet \wedge hat$
occ1 [2,3] $JumpIn$
occ2 [5,6] $GetOut$
acs1 $[t_1, t_2]\ JumpIn \rightsquigarrow [t_1, t_2]\ inlake := T$
acs2 $[t_1, t_2]\ GetOut \rightsquigarrow [t_1, t_2]\ inlake := F$
cc1 $\forall t.[t]\ inlake \gg [t]\ wet \wedge (hat \vee \neg hat)$
cc2 $\forall t.[t]\ \neg inlake \gg [t+5]\ \neg wet$

One problem with this approach to delayed effects is the possibility of intervening events which would qualify being dry at a later time-point. One possibility would be to use a context to constrain the applicability of the causal constraint. Here is an alternative:

cc2 $\forall t.(\forall t_1.t < t_1 < t + 5 \rightarrow \neg Holds(t_1, inlake)) \rightarrow ([t]\ \neg inlake \gg [t+5]\ \neg wet)$

Roughly, this causal constraint states that "If I get out of the lake and stay out of the lake for 5 minutes, then I will not be wet."

One can view this problem in two ways. In the first, causal constraints behave very much like action effect rules and one might try and deal with *causal qualification* in a manner similar to action qualification. One other way to view the matter is that a delayed causal effect should be interpreted as an action which may occur concurrently with others. Consequently, one would have to deal with *keep* conditions and *cancellation* conditions among concurrent actions.

C.2.2 Extended Electric Circuit Scenario

This scenario is intended to show that categorization-based approaches for encoding dependencies among fluents when modeling indirect effects are most probably not adequate because the category a fluent is classified as being in may change dynamically. For example, in the following scenario the fluent $sw2$ is primary and *light* secondary in the causal constraint $cc1$, while $sw2$ is secondary in the causal constraint $cc5$. In addition, the causal constraints provide a good test to assure that casual chains are transitive. The intended effect of turning on $sw1$ is that the *relay* goes on, $sw2$ is popped to the off position and the *light* never goes on. The unintended model would be that where $sw3$ magically goes to the off position rather than the *relay* going on. In PMON(RCs) there is one preferred model which coincides with the intended model.

Although one could agree with Thielscher's analysis as regards current approaches which use categorization, it would appear to be a straightforward exercise to add an extra state argument to a $Frame$ or *Release* predicate in order to encode context dependent or dynamic categorization. How one would choose to place fluents in different categories is still problematic. It seems that this is the wrong level of abstraction to encode dependencies in a reasonable manner. However, such a level of specification could easily be the result of a compilation of sorts from a higher level description of causal rules or relations. Note that the current approach which we use is essentially a categorization-based approach, yet provides the intended models. In our case, there are two categories: *Occluded* and non-*Occluded*.

Example C.20 (Extended Electric Circuit Scenario, Thielscher [29])

Action Symbols: $SwitchOn$(`switch`).
Feature Symbols: on(`switch`), *light*, *relay*
Object Symbols: $sw1$: `switch`, $sw2$: `switch`, $sw3$: `switch`

obs1 $[0]\ \neg on(sw1) \wedge on(sw2) \wedge on(sw3)$
obs2 $[0]\ \neg light \wedge \neg relay$
occ1 [2,3] $SwitchOn(sw1)$
acs1 $[t_1, t_2]\ SwitchOn(sw) \rightsquigarrow [t_1, t_2]\ on(sw) := T$
cc1 $\forall t.[t]\ (on(sw1) \wedge on(sw2)) \gg [t]\ light$
cc2 $\forall t.[t]\ \neg(on(sw1) \wedge on(sw2)) \gg [t]\ \neg light$
cc3 $\forall t.[t]\ (on(sw1) \wedge on(sw3)) \gg [t]\ relay$
cc4 $\forall t.[t]\ \neg(on(sw1) \wedge on(sw3)) \gg [t]\ \neg relay$
cc5 $\forall t.[t]\ relay \gg [t]\ \neg on(sw2)$

Example C.21 The expanded action scenario description is,

obs1 $[0]\ Observe(\neg on(sw1) \wedge on(sw2) \wedge on(sw3))$
obs2 $[0]\ Observe(\neg light \wedge \neg relay)$
occ1 [2,3] $SwitchOn(sw1)$
scd1 $[3]\ on(sw1) := T$
cc1 $\forall t.[t]\ (on(sw1) \wedge on(sw2)) \gg [t]\ light$
cc2 $\forall t.[t]\ \neg(on(sw1) \wedge on(sw2)) \gg [t]\ \neg light$
cc3 $\forall t.[t]\ (on(sw1) \wedge on(sw3)) \gg [t]\ relay$
cc4 $\forall t.[t]\ \neg(on(sw1) \wedge on(sw3)) \gg [t]\ \neg relay$
cc5 $\forall t.[t]\ relay \gg [t]\ \neg on(sw2)$

Example C.22 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

obs1 $\neg Holds(0, \mathsf{on(sw1)}) \wedge Observe(0, \mathsf{on(sw1)}) \wedge Holds(0, \mathsf{on(sw2)}) \wedge Observe(0, \mathsf{on(sw2)}) \wedge$
$Holds(0, \mathsf{on(sw3)}) \wedge Observe(0, \mathsf{on(sw3)})$
obs2 $\neg Holds(0, \mathsf{light}) \wedge Observe(0, \mathsf{light}) \wedge \neg Holds(0, \mathsf{relay}) \wedge Observe(0, \mathsf{relay})$
occ1 $Occurs(2, 3, \mathsf{SwitchOn(sw1)})$
scd1 $Holds(3, \mathsf{on(sw1)}) \wedge Occlude(3, \mathsf{on(sw1)})$
cc1 $(\forall t_1.(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)})) \rightarrow Holds(t_1, \mathsf{light})) \wedge$
$\forall t_1.[\neg(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw2)})) \wedge (Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)}))]$
$\rightarrow Occlude(t_1, \mathsf{light})$
cc2 $(\forall t_1.\neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)})) \rightarrow \neg Holds(t_1, \mathsf{light})) \wedge$
$\forall t_1.[(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw2)})) \wedge \neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)}))]$
$\rightarrow Occlude(t_1, \mathsf{light})$
cc3 $(\forall t_1.(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)})) \rightarrow Holds(t_1, \mathsf{relay})) \wedge$
$\forall t_1.[\neg(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw3)})) \wedge (Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)}))]$
$\rightarrow Occlude(t_1, \mathsf{relay})$
cc4 $(\forall t_1.\neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)})) \rightarrow \neg Holds(t_1, \mathsf{relay})) \wedge$
$\forall t_1.[(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw3)})) \wedge \neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)}))]$
$\rightarrow Occlude(t_1, \mathsf{relay})$
cc5 $(\forall t_1.(Holds(t_1, \mathsf{relay}) \rightarrow \neg Holds(t_1, \mathsf{on(sw2)})) \wedge$
$\forall t_1.[(\neg Holds(t_1 - 1, \mathsf{relay}) \wedge Holds(t_1 - 1, \mathsf{relay})) \rightarrow Occlude(t_1, \mathsf{on(sw2)})]$

Note that $cc1 \wedge cc2$ is equivalent to

$(\forall t_1.(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)})) \leftrightarrow Holds(t_1, \mathsf{light})) \wedge$
$\forall t_1.[(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw2)})) \oplus (Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)}))]$
$\rightarrow Occlude(t_1, \mathsf{light})$

and that $cc3 \wedge cc4$ is equivalent to

$(\forall t_1.(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)})) \leftrightarrow Holds(t_1, \mathsf{relay})) \wedge$
$\forall t_1.[(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw3)})) \oplus (Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)}))]$
$\rightarrow Occlude(t_1, \mathsf{relay})$

C.2.3 Extended Electric Circuit Example with Device

Thielscher ([29], pp 22-23) uses this example to put forth the argument that approaches which "minimize change" might not be adequate for modeling indirect effects of actions because, here one wants to minimize *uncaused change* which is not the same as minimizing *change*. His formalism appears to result in two successor states[6], where one differs from the initial state in strictly more values than the other does. The claim is that a minimal change approach would not allow the *larger* model. Thielscher states that,

> When the first switch $sw1$, is toggled ..., then we would expect two possible successor states due to the non-deterministic behavior of the circuit: In any case, we end up with the relay activated and both the second switch and the light bulb off. Yet, the complete outcome depends on whether or not the activation of the relay and its affecting the second switch is faster than the intermediate activation of the light bulb and, triggered by this, the activation of the photo-device. Since *detect* remains true once activated, it might happen that the fluent additionally becomes true.

We would claim on the contrary, that there is a confusion in Thielscher's approach between the procedural non-determinism used in computing a fixpoint (successor state) for the indirect effects of an action , and any perceived nondeterminism in this example. Clearly, there could be nondeterminism as Thieschler claims, but this is simply not axiomatized. One would have to introduce additional timing constraints in order to consider the delayed effects of the circuit. There are a number of ways to do this. We consider one approach in the next section.

Example C.23 (Extended Electric Circuit with Device Scenario, Thielscher [29])

Action Symbols: $SwitchOn(\texttt{switch})$.
Feature Symbols: $on(\texttt{switch})$, $light$, $relay$, $detect$
Object Symbols: $sw1 : \texttt{switch}$, $sw2 : \texttt{switch}$, $sw3 : \texttt{switch}$

obs1	$[0]\ \neg on(sw1) \wedge on(sw2) \wedge on(sw3)$
obs2	$[0]\ \neg light \wedge \neg relay \wedge \neg detect$
occ1	[2,3] $SwitchOn(sw1)$
acs1	$[t_1, t_2]\ SwitchOn(sw) \rightsquigarrow [t_1, t_2]\ on(sw) := T$
cc1	$\forall t.[t]\ (on(sw1) \wedge on(sw2)) \gg [t]\ light$
cc2	$\forall t.[t]\ \neg(on(sw1) \wedge on(sw2)) \gg [t]\ \neg light$
cc3	$\forall t.[t]\ (on(sw1) \wedge on(sw3)) \gg [t]\ relay$
cc4	$\forall t.[t]\ \neg(on(sw1) \wedge on(sw3)) \gg [t]\ \neg relay$
cc5	$\forall t.[t]\ relay \gg [t]\ \neg on(sw2)$
cc6	$\forall t.[t]\ light \gg [t]\ detect$

Example C.24 The expanded action scenario description is,

obs1	$[0]\ Observe(\neg on(sw1) \wedge on(sw2) \wedge on(sw3))$
obs2	$[0]\ Observe(\neg light \wedge \neg relay \wedge \neg detect)$
occ1	[2,3] $SwitchOn(sw1)$
scd1	$[3]\ on(sw1) := T$
cc1	$\forall t.[t]\ (on(sw1) \wedge on(sw2)) \gg [t]\ light$
cc2	$\forall t.[t]\ \neg(on(sw1) \wedge on(sw2)) \gg [t]\ \neg light$
cc3	$\forall t.[t]\ (on(sw1) \wedge on(sw3)) \gg [t]\ relay$
cc4	$\forall t.[t]\ \neg(on(sw1) \wedge on(sw3)) \gg [t]\ \neg relay$
cc5	$\forall t.[t]\ relay \gg [t]\ \neg on(sw2)$
cc6	$\forall t.[t]\ light \gg [t]\ detect$

[6] It is a bit difficult to figure out from the wording in the quote above and the technical report, whether there are one or two successor states and whether the result differs from that in the previous example. Either way, we would still claim that the problem is under axiomatized and even more unintuitive if there is only one succesor state.

Example C.25 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

obs1 $\neg Holds(0, \mathsf{on(sw1)}) \wedge Observe(0, \mathsf{on(sw1)}) \wedge Holds(0, \mathsf{on(sw2)}) \wedge Observe(0, \mathsf{on(sw2)}) \wedge$
$Holds(0, \mathsf{on(sw3)}) \wedge Observe(0, \mathsf{on(sw3)})$

obs2 $\neg Holds(0, \mathsf{light}) \wedge Observe(0, \mathsf{light}) \wedge \neg Holds(0, \mathsf{relay}) \wedge Observe(0, \mathsf{relay})$
$\neg Holds(0, \mathsf{detect}) \wedge Observe(0, \mathsf{detect})$

occ1 $Occurs(2, 3, \mathsf{SwitchOn(sw1)})$

scd1 $Holds(3, \mathsf{on(sw1)}) \wedge Occlude(3, \mathsf{on(sw1)})$

cc1 $(\forall t_1.(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)})) \rightarrow Holds(t_1, \mathsf{light})) \wedge$
$\forall t_1.[\neg(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw2)})) \wedge (Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)}))]$
$\rightarrow Occlude(t_1, \mathsf{light})$

cc2 $(\forall t_1.\neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)})) \rightarrow \neg Holds(t_1, \mathsf{light})) \wedge$
$\forall t_1.[(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw2)})) \wedge \neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)}))]$
$\rightarrow Occlude(t_1, \mathsf{light})$

cc3 $(\forall t_1.(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)})) \rightarrow Holds(t_1, \mathsf{relay})) \wedge$
$\forall t_1.[\neg(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw3)})) \wedge (Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)}))]$
$\rightarrow Occlude(t_1, \mathsf{relay})$

cc4 $(\forall t_1.\neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)})) \rightarrow \neg Holds(t_1, \mathsf{relay})) \wedge$
$\forall t_1.[(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw3)})) \wedge \neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)}))]$
$\rightarrow Occlude(t_1, \mathsf{relay})$

cc5 $(\forall t_1.(Holds(t_1, \mathsf{relay}) \rightarrow \neg Holds(t_1, \mathsf{on(sw2)})) \wedge$
$\forall t_1.[(\neg Holds(t_1 - 1, \mathsf{relay}) \wedge Holds(t_1, \mathsf{relay})) \rightarrow Occlude(t_1, \mathsf{on(sw2)})]$

cc6 $(\forall t_1.(Holds(t_1, \mathsf{light}) \rightarrow Holds(t_1, \mathsf{detect})) \wedge$
$\forall t_1.[(\neg Holds(t_1 - 1, \mathsf{light}) \wedge Holds(t_1, \mathsf{light})) \rightarrow Occlude(t_1, \mathsf{detect})]$

C.2.4 The Delayed Circuit Scenario

This a good example where the use of delayed effects can be used to model *causal lag*. If the value of the temporal constant $\mathsf{t_1}$ is less than $\mathsf{t_2}$ then this implies that when $sw1$ and $sw2$ are on that the *light* will be on for the period $(\mathsf{t_2} - \mathsf{t_1}) + \mathsf{t_1}$. At this point the *relay* kicks in and turns $sw2$ off. When this happens the *light* will go off $\mathsf{t_1}$ time-points later. This means that the *detector* will go on and stay on even after the *light* goes off. If the value of the temporal constant $\mathsf{t_1}$ is greater than $\mathsf{t_2}$ then this implies that the *light* will never go on because the *relay* kicks in before and turns $sw2$ off. If the value of the temporal constant $\mathsf{t_1}$ is equal to $\mathsf{t_2}$ then the results will be the same.

Without placing additional constraints on $\mathsf{t_1}$ and $\mathsf{t_2}$, the preferred models would provide both types of conclusion, so, the nondeterminism is made explicit in the axioms and the minimization policy in PMON(RCs) does not interfere with the results.

Action Symbols: $SwitchOn(\mathtt{switch})$.
Feature Symbols: $on(\mathtt{switch})$, *light*, *relay*, *detect*
Object Symbols: $sw1 : \mathtt{switch}$, $sw2 : \mathtt{switch}$, $sw3 : \mathtt{switch}$

Example C.26

obs1 $[0]\ \neg on(sw1) \wedge on(sw2) \wedge on(sw3)$
obs2 $[0]\ \neg light \wedge \neg relay \wedge \neg detect$
occ1 [2,3] $SwitchOn(sw1)$
acs1 $[t_1, t_2]\ SwitchOn(sw) \rightsquigarrow [t_1, t_2]\ on(sw) := T$
cc1 $\forall t.\mathsf{t_1} < \mathsf{t_2} \rightarrow ([t]\ (on(sw1) \wedge on(sw2)) \gg [t + \mathsf{t_1}]\ light)$
cc2 $\forall t.\mathsf{t_1} < \mathsf{t_2} \rightarrow ([t]\ \neg(on(sw1) \wedge on(sw2)) \gg [t + \mathsf{t_1}]\ \neg light)$
cc3 $\forall t.[t]\ (on(sw1) \wedge on(sw3)) \gg [t + \mathsf{t_2}]\ relay$
cc4 $\forall t.[t]\ \neg(on(sw1) \wedge on(sw3)) \gg [t + \mathsf{t_2}]\ \neg relay$
cc5 $\forall t.[t]\ relay \gg [t]\ \neg on(sw2)$
cc6 $\forall t.[t]\ light \gg [t]\ detect$

Example C.27 The expanded action scenario description is,

obs1 $[0]\ Observe(\neg on(sw1) \wedge on(sw2) \wedge on(sw3))$
obs2 $[0]\ Observe(\neg light \wedge \neg relay \wedge \neg detect)$
occ1 [2,3] $SwitchOn(sw1)$
scd1 $[3]\ on(sw1) := T$
cc1 $\forall t.\mathsf{t_1} < \mathsf{t_2} \rightarrow ([t]\ (on(sw1) \wedge on(sw2)) \gg [t + \mathsf{t_1}]\ light)$
cc2 $\forall t.\mathsf{t_1} < \mathsf{t_2} \rightarrow ([t]\neg(on(sw1) \wedge on(sw2)) \gg [t + \mathsf{t_1}]\ \neg light)$
cc3 $\forall t.[t]\ (on(sw1) \wedge on(sw3)) \gg [t]\ relay$
cc4 $\forall t.[t]\ \neg(on(sw1) \wedge on(sw3)) \gg [t]\ \neg relay$
cc5 $\forall t.[t]\ relay \gg [t]\ \neg on(sw2)$
cc6 $\forall t.[t]\ light \gg [t]\ detect$

Example C.28 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

obs1 $\neg Holds(0, \mathsf{on(sw1)}) \wedge Observe(0, \mathsf{on(sw1)}) \wedge Holds(0, \mathsf{on(sw2)}) \wedge Observe(0, \mathsf{on(sw2)}) \wedge$
$Holds(0, \mathsf{on(sw3)}) \wedge Observe(0, \mathsf{on(sw3)})$
obs2 $\neg Holds(0, \mathsf{light}) \wedge Observe(0, \mathsf{light}) \wedge \neg Holds(0, \mathsf{relay}) \wedge Observe(0, \mathsf{relay})$
$\neg Holds(0, \mathsf{detect}) \wedge Observe(0, \mathsf{detect})$
occ1 $Occurs(2, 3, \mathsf{SwitchOn(sw1)})$
scd1 $Holds(3, \mathsf{on(sw1)}) \wedge Occlude(3, \mathsf{on(sw1)})$
cc1 $(\forall t_1.(\mathsf{t_1} < \mathsf{t_2} \wedge Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)})) \rightarrow Holds(t_1 + \mathsf{t_1}, \mathsf{light})) \wedge$
$\forall t_1.[\mathsf{t_1} < \mathsf{t_2} \wedge \neg(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw2)})) \wedge$
$(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)}))] \rightarrow Occlude(t_1 + \mathsf{t_1}, \mathsf{light})$
cc2 $(\forall t_1.\mathsf{t_1} < \mathsf{t_2} \wedge \neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)})) \rightarrow \neg Holds(t_1 + \mathsf{t_1}, \mathsf{light})) \wedge$
$\forall t_1.[(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw2)})) \wedge \neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw2)}))]$
$\rightarrow Occlude(t_1 + \mathsf{t_1}, \mathsf{light})$
cc3 $(\forall t_1.(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)})) \rightarrow Holds(t_1 + \mathsf{t_2}, \mathsf{relay})) \wedge$
$\forall t_1.[\neg(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw3)})) \wedge (Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)}))]$
$\rightarrow Occlude(t_1 + \mathsf{t_2}, \mathsf{relay})$
cc4 $(\forall t_1.\neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)})) \rightarrow \neg Holds(t_1 + \mathsf{t_2}, \mathsf{relay})) \wedge$
$\forall t_1.[(Holds(t_1 - 1, \mathsf{on(sw1)}) \wedge Holds(t_1 - 1, \mathsf{on(sw3)})) \wedge \neg(Holds(t_1, \mathsf{on(sw1)}) \wedge Holds(t_1, \mathsf{on(sw3)}))]$
$\rightarrow Occlude(t_1 + \mathsf{t_2}, \mathsf{relay})$
cc5 $(\forall t_1.(Holds(t_1, \mathsf{relay}) \rightarrow \neg Holds(t_1, \mathsf{on(sw2)})) \wedge$
$\forall t_1.[(\neg Holds(t_1 - 1, \mathsf{relay}) \wedge Holds(t_1, \mathsf{relay})) \rightarrow Occlude(t_1, \mathsf{on(sw2)})]$
cc6 $(\forall t_1.(Holds(t_1, \mathsf{light}) \rightarrow Holds(t_1, \mathsf{detect})) \wedge$
$\forall t_1.[(\neg Holds(t_1 - 1, \mathsf{light}) \wedge Holds(t_1, \mathsf{light})) \rightarrow Occlude(t_1, \mathsf{detect})]$

C.2.5 The Trap Door Scenario

This example is an elaboration on an example due to McCain and Turner where one wants to distinguish between what are called ramification and qualification constraints. Here, the problem is more subtle, a precondition to a causal constraint has some fluents which should play the role

of indirect effects, where they are meant to change from false to true or vice-versa, and others simply have to be true or false.[7]

The idea is as follows; in the first scenario, the turkey is alive and is not at a trapdoor, and the trapdoor is closed. If the turkey is enticed to the trapdoor and then it is opened, the turkey will no longer be alive.

In the second scenario, the turkey is alive and is not at a trapdoor, but the trapdoor is open. The claim is that the trapdoor being opened is an implicit qualification to the action *Entice* and that the action should fail. The action is under-specified and the causal constraint brings this point to light. In this particular case, failure of an action would mean an inconsistent scenario because we do not deal with qualification problems. In a similar manner, Thielscher's formalism would not be able to compute a successor state.

Thielscher claims that this example implies that any approach based on "minimization of change" will inadequately deal with this distinction. Even though one might claim that our approach is in this class (actually, we minimize potential change), we simply do not have these problems. This is more a matter of what granularity a formalism has in restricting global inertia policies. Like Thielscher, we can distinguish between fluents in preconditions to causal constraints that play the role of being indirect effects of actions, or simply must be true or false. This is done by using the context of our causal constraint. In this case, rather than using the causal constraint:

$$\forall x, t.([t]at(x, trapdoor) \wedge opened(trapdoor) \gg \neg alive(x)),$$

we use,

$$\forall x, t.[t]at(x, trapdoor) \rightarrow ([t]opened(trapdoor) \gg \neg alive(x)).$$

Example C.29 (The Trap Door Scenario, Thielscher [29])

Action Symbols: $Open(\texttt{door})$, $Entice(\texttt{thing}, \texttt{door})$.
Feature Symbols: $alive(\texttt{thing})$, $opened(\texttt{door})$. $at(\texttt{thing}, \texttt{door})$
Object Symbols: $turkey : \texttt{thing}$, $trapdoor : \texttt{door}$

obs1	[0] $alive(turkey) \wedge \neg at(turkey, trapdoor) \wedge \neg opened(trapdoor)$
occ1	[1,2] $Entice(turkey, trapdoor)$
occ2	[3,4] $Open(trapdoor)$
acs1	$[t_1, t_2]$ $Open(door) \rightsquigarrow [t_1, t_2]$ $opened(door) := T$
acs2	$[t_1, t_2]$ $Entice(x, door) \rightsquigarrow [t_1]$ $\neg at(x, door) \rightarrow [t_1, t_2]$ $at(x, door) := T$
cc	$\forall x, t.[t]at(x, trapdoor) \rightarrow ([t]opened(trapdoor) \gg \neg alive(x))$

Example C.30 The expanded action scenario description is,

obs1	[0] $Observe(alive(turkey) \wedge \neg at(turkey, trapdoor) \wedge \neg opened(trapdoor))$
occ1	[1,2] $Entice(turkey, trapdoor)$
scd2	[1] $\neg at(turkey, trapdoor) \rightarrow$ [1,2] $at(turkey, trapdoor) := T$
occ2	[3,4] $Open(trapdoor)$
scd2	[3,4] $opened(trapdoor) := T$
cc	$\forall x, t.[t]at(x, trapdoor) \rightarrow ([t]opened(trapdoor) \gg \neg alive(x))$

Example C.31 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

[7] Its somewhat difficult to make sense of the whole discussion regarding implicit qualification and one begins to wonder whether it is really an issue at all. Since what one means by implicit qualification depends on what one means by qualification, we will place this issue on the shelf until PMON is extended for qualification. We simply note that the current non-qualified version appears to behave in a manner consistent with the current discussion in the literature.

obs1 $Holds(0, \mathsf{alive(turkey)}) \wedge Observe(0, \mathsf{alive(turkey)} \wedge$
$\neg Holds(0, \mathsf{at(turkey, trapdoor)}) \wedge Observe(0, \mathsf{at(turkey, trapdoor)})$
$\neg Holds(0, \mathsf{opened(trapdoor)}) \wedge Observe(0, \mathsf{opened(trapdoor)})$
occ1 $Occurs(1, 2, \mathsf{Entice(turkey, trapdoor)})$
scd1 $\neg Holds(1, \mathsf{at(turkey, trapdoor)}) \rightarrow Holds(2, \mathsf{at(turkey, trapdoor)}) \wedge$
$\neg Holds(1, \mathsf{at(turkey, trapdoor)}) \rightarrow Occlude(2, \mathsf{at(turkey, trapdoor)})$
occ2 $Occurs(3, 4, \mathsf{Open(trapdoor)})$
scd2 $Holds(4, \mathsf{opened(trapdoor)}) \wedge Occlude(4, \mathsf{opened(trapdoor)})$
cc $(\forall x, t_1.Holds(t_1, at(x, trapdoor) \rightarrow ((Holds(t_1, \mathsf{opened(trapdoor)}) \rightarrow \neg Holds(t_1, \mathsf{alive(x)}))) \wedge$
$\forall t_1.Holds(t_1, at(x, trapdoor) \rightarrow [(\neg Holds(t_1 - 1, \mathsf{opened(trapdoor)}) \wedge Holds(t_1 - 1, \mathsf{opened(trapdoor)}))$
$\rightarrow Occlude(t_1, \mathsf{alive(x)})]$

Example C.32 (The Trap Door Scenario and Derived Qualification)

obs1 [0] $alive(turkey) \wedge \neg at(turkey, trapdoor) \wedge opened(trapdoor)$
occ1 [1,2] $Entice(turkey, trapdoor)$
acs1 $[t_1, t_2]\ Open(door) \rightsquigarrow [t_1, t_2]\ opened(door) := T$
acs2 $[t_1, t_2]\ Entice(x, door) \rightsquigarrow [t_1]\ \neg at(x, door) \rightarrow [t_1, t_2]\ at(x, door) := T$
cc $\forall x, t.[t]at(x, trapdoor) \rightarrow ([t]opened(trapdoor) \gg \neg alive(x))$

Here, because the trapdoor is initially open, the precondition to the causal constraint (that $opened(trapdoor)$ changes from false to true) is not satisfied. Consequently, the fluent $alive(turkey)$ is not allowed to change value, yet the domain constraint states that it must. This results in an inconsistent scenario. Note that if we had used the first causal constraint alternative,

$$\forall x, t.([t]at(x, trapdoor) \wedge opened(trapdoor) \gg \neg alive(x)),$$

that the scenario would not be inconsistent.

C.2.6 Extended Baby Protection Scenario

This example provides a number of tests. Firstly, one uses both propositional and non-propositional value domains. The use of the latter implies new value ambiguity as one can see from the action schema $acs4$. New value ambiguity introduces nondeterminism. The question is whether nondeterministic effects interact properly with the indirect effects of actions. This example also shows the importance of using context with causal constraints. The combination of interactions in this example provide a good test for how *fine-grained* ones formalism is as regards specifying change and indirect change on an object by object basis.

There are two classes of preferred models in this example due to the ambiguity that arises when the *gun* is removed from the *closet*. If it ends up on the *table* then it is safe, otherwise it is not. One can preferentially entail the following: From [0 2), both the *gun* and *toy* are $unsafe$. From [2,4), the *gun* is safe and the *toy* is not. From [4,5), the *toy* remains unsafe while the *gun* becomes $unsafe$. From [6,8), the *toy* remains $unsafe$, but in this case, the *gun* may or may not be safe, depending on whether it is on the *floor* or on the *table*. From [8,∞] the *toy* is $safe$ and we do not know whether the *gun* is $safe$ or not. Note also that closing and opening the closet door will only affect the status of objects in the closet and not those at other locations.

Example C.33 (Extended Baby Protection Scenario, Giunchiglia et. al. [7])
This example is a modification of one originally due to Myers and Smith [20].

Action Symbols: $Close(\mathtt{door})$, $Open(\mathtt{door})$, $Put(\mathtt{thing}, \mathtt{location})$, $Remove(\mathtt{thing}, \mathtt{location})$
Feature Symbols: $closed(\mathtt{door})$, $safe(\mathtt{thing})$, $loc : \mathtt{thing} \rightarrow \mathtt{location}$
Object Symbols: $door1 : \mathtt{door}$, $gun : \mathtt{thing}$, $toy : \mathtt{thing}$, $table : \mathtt{location}$, $closet : \mathtt{location}$, $floor : \mathtt{location}$

obs1 [0] $loc(toy)\hat{=}floor \wedge loc(gun1)\hat{=}closet \wedge \neg closed(door1)$
occ1 [1,2] $Close(door1)$
occ2 [3,4] $Open(door1)$
occ3 [5,6] $Remove(gun, closet)$
occ4 [7,8] $Put(toy, table)$
acs1 $[t_1, t_2]\ Close(d) \rightsquigarrow [t_1,t_2]\ closed(d) := T$
acs2 $[t_1, t_2]\ Open(d) \rightsquigarrow [t_1,t_2]\ closed(d) := F$
acs3 $[t_1, t_2]\ Put(x, l) \rightsquigarrow [t_1,t_2]\ loc(x) := l$
acs4 $[t_1, t_2]\ Remove(x, l) \rightsquigarrow [t_1]loc(x)\hat{=}l \rightarrow [t_1,t_2]\ \neg(loc(x) := l)$
cc1 $\forall x, t.[t]loc(x)\hat{=}table \gg [t]safe(x)$
cc2 $\forall x, t.[t]loc(x)\hat{=}floor \gg [t]\neg safe(x)$
cc3 $\forall x, t.loc(x, closet) \rightarrow ([t]closed(door1) \gg [t]safe(x))$
cc4 $\forall x, t.loc(x, closet) \rightarrow ([t]\neg closed(door1) \gg [t]\neg safe(x))$

Example C.34 The expanded action scenario description is,

obs1 [0] $Observe(loc(toy)\hat{=}floor \wedge loc(gun1)\hat{=}closet \wedge \neg closed(door1))$
occ1 [1,2] $Close(door1)$
occ2 [3,4] $Open(door1)$
occ3 [5,6] $Remove(gun, closet)$
occ4 [7,8] $Put(toy, table)$
scd1 [1,2] $closed(door1) := T$
scd2 [1,2] $closed(door1) := F$
scd3 [5]$loc(gun)\hat{=}closet \rightarrow$ [5,6] $\neg(loc(gun) := closet)$
scd4 [7,8] $(loc(toy) := table)$
cc1 $\forall x, t.[t]loc(x)\hat{=}table \gg [t]safe(x)$
cc2 $\forall x, t.[t]loc(x)\hat{=}floor \gg [t]\neg safe(x)$
cc3 $\forall x, t.loc(x, closet) \rightarrow ([t]closed(door1) \gg [t]safe(x))$
cc4 $\forall x, t.loc(x, closet) \rightarrow ([t]\neg closed(door1) \gg [t]\neg safe(x))$

Example C.35 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

obs1 $Holds(0, \mathsf{loc(toy, floor)}) \wedge Holds(0, \mathsf{loc(gun, closet)}) \wedge \neg Holds(0, \mathsf{closed(door1)})$
$\wedge Observe(0, \mathsf{loc(toy, floor)}) \wedge Observe(0, \mathsf{loc(gun, closet)}) \wedge Observe(0, \mathsf{closed(door1)})$
occ1 $Occurs(1, 2, \mathsf{Close(door1)})$
occ2 $Occurs(3, 4, \mathsf{Open(door1)})$
occ3 $Occurs(5, 6, \mathsf{Remove(gun, closet)})$
occ4 $Occurs(7, 8, \mathsf{Put(toy, table)})$
scd1 $Holds(2, \mathsf{closed(door1)}) \wedge Occlude(2, \mathsf{closed(door1)})$
scd2 $\neg Holds(4, \mathsf{closed(door1)}) \wedge Occlude(4, \mathsf{closed(door1)})$
scd3 $[Holds(5, \mathsf{loc(gun, closet)}) \rightarrow \neg Holds(6, \mathsf{loc(gun, closet)})] \wedge$
$[Holds(5, \mathsf{loc(gun, closet)}) \rightarrow \forall l.Occlude(6, \mathsf{loc(gun, l)})]$
scd4 $Holds(8, \mathsf{loc(toy, table)}) \wedge \forall l.Occlude(8, \mathsf{loc(toy, l)})$
cc1 $[\forall x, t_1.Holds(t_1, \mathsf{loc}(x, \mathsf{table})) \rightarrow Holds(t_1, \mathsf{safe}(x))] \wedge$
$\forall x, t_1.(\neg Holds(t_1 - 1, \mathsf{loc}(x, \mathsf{table})) \wedge Holds(t_1, \mathsf{loc}(x, \mathsf{table}))) \rightarrow Occlude(t, \mathsf{safe}(x))$
cc2 $[\forall x, t_1.Holds(t_1, \mathsf{loc}(x, \mathsf{floor})) \rightarrow \neg(Holds(t_1, \mathsf{safe}(x)))] \wedge$
$[\forall x, t_1.(\neg Holds(t_1 - 1, \mathsf{loc}(x, \mathsf{floor})) \wedge Holds(t_1, \mathsf{loc}(x, \mathsf{floor}))) \rightarrow Occlude(t, \mathsf{safe}(x))]$
cc3 $[\forall x, t_1.Holds(t_1, \mathsf{loc}(x, \mathsf{closet})) \wedge Holds(t_1, \mathsf{closed(door1)}) \rightarrow Holds(t_1, \mathsf{safe}(x))] \wedge$
$\forall x, t_1.(Holds(t_1, \mathsf{loc}(x, \mathsf{closet})) \wedge \neg Holds(t_1 - 1, \mathsf{closed(door1)}) \wedge Holds(t_1, \mathsf{closed(door1)}))$
$\rightarrow Occlude(t_1, \mathsf{safe}(x))]$
cc4 $[\forall x, t_1.Holds(t_1, \mathsf{loc}(x, \mathsf{closet})) \wedge \neg Holds(t_1, \mathsf{closed(door1)}) \rightarrow \neg Holds(t_1, \mathsf{safe}(x))] \wedge$
$\forall x, t_1.(Holds(t_1, \mathsf{loc}(x, \mathsf{closet})) \wedge Holds(t_1 - 1, \mathsf{closed(door1)}) \wedge \neg Holds(t_1, \mathsf{closed(door1)}))$
$\rightarrow Occlude(t_1, \mathsf{safe}(x))]$

Acknowledgements

I'm grateful to KPLAB members, Lars Karlsson, Joakim Gustafsson, and Jonas Kvarnström, for useful comments regarding the various drafts of this paper. Part of the work dealing with ramification and PMON(RCs) is joint work done with Joakim Gustafsson.

References

[1] A. Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5–23, 1991.

[2] J. Crawford. Three issues in action. Unpublished note for the 5th Int. Workshop on Nonmonotonic Reasoning, 1994.

[3] P. Doherty. Notes on PMON circumscription. Technical Report LITH-IDA-94-43, Department of Computer and Information Science, Linköping University, Linköping, Sweden, December 1994.

[4] P. Doherty. Reasoning about action and change using occlusion. In *Proceedings of the 11th European Conference on Artificial Intelligence, Aug. 8-12, Amsterdam*, pages 401–405, 1994.

[5] P. Doherty and W. Lukaszewicz. Circumscribing features and fluents. In D. Gabbay and H. J. Ohlbach, editors, *Proceedings of the 1st International Conference on Temporal Logic*, volume 827 of *Lecture Notes in Artificial Intelligence*, pages 82–100. Springer, 1994.

[6] P. Doherty and W. Lukaszewicz. Circumscribing features and fluents: A fluent logic for reasoning about action and change. In *Proceedings of the 8th International Symposium on Methodologies for Intelligent Systems*, 1994.

[7] E Giunchiglia and V. Lifschitz. Dependent fluents. In *Proc. of the 14th Int'l Conf. on Artificial Intelligence*, 1995.

[8] J. Gustafsson and P. Doherty. Embracing occlusion in specifying the indirect effects of actions. In *5th International Conference on Principles of Knowledge Representation and Reasoning*, 1996.

[9] S. Hanks and D. McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33, 1987.

[10] L. Karlsson. Specification and synthesis of plans using the features and fluents framework. Technical Report LiTH-IDA-R-94-28, Dept. of Computer and Info. Science, Linköping University, 1994.

[11] Lars Karlsson. Specification and synthesis of plans using the features and fluents framework. Licentiate thesis, Department of Computer and Information Science, Linköping University, 1995. Available on WWW: `http://www.ida.liu.se/labs/rkllab/people/larka/`.

[12] Lars Karlsson. Causal links planning and the systematic approach to action and change. In *Proceedings of the AAAI 96 Workshop on Reasoning about actions, planning and control: bridging the gap*, Portland, Oregon, August 1996. AAAI Press.

[13] Lars Karlsson. Planning, truth criteria and the systematic approach to action and change. In *Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems*, Lecture Notes for Artificial Intelligence. Springer Verlag, 1996. To appear.

[14] G. N. Kartha. Two counterexamples related to baker's approach to the frame problem. *Artificial Intelligence*, pages 379–391, 1994.

[15] M. Koubarakis. Complexity results for first-order theories of temporal constraints. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 379–390, 1994. (KR-94).

[16] V. Lifschitz. Pointwise circumscription. In M. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 179–193. Morgan Kaufmann, 1988.

[17] V. Lifschitz. Circumscription. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3 of *Handbook of Artificial Intelligence and Logic Programming*. Oxford University Press, 1994.

[18] V. Lifschitz and A. Rabinov. Things that change by themselves. In *Proc. of the Int'l Joint Conference on Artificial Intelligence (IJCAI-89)*, 1989.

[19] W. Lukaszewicz. *Non-Monotonic Reasoning – Formalization of Commonsense Reasoning*. Ellis Horwood Series in Artificial Intelligence. Ellis Horwood, 1990.

[20] K. L. Myers and D. E. Smith. The persistence of derived information. In *Proc. AAAI-93*, pages 496–500, 1988.

[21] E. Sandewall. Non-monotonic entailment for reasoning about time and action: Part i: Sequential actions. Technical Report LITH-IDA-R-88-27, Department of Computer and Information Science, Linköping University, 1988.

[22] E. Sandewall. Non-monotonic entailment for reasoning about time and action: Part iii: Decision procedure. Technical Report LITH-IDA-R-88-29, Department of Computer and Information Science, Linköping University, 1988.

[23] E. Sandewall. Filter preferential entailment for the logic of action and change. In *Proc. Int'l Joint Conf. on Artificial Intelligence, (IJCAI-89)*, 1989.

[24] E. Sandewall. Features and fluents. Technical Report LITH-IDA-R-91-29, Department of Computer and Information Science, Linköping University, 1991.

[25] E. Sandewall. Features and fluents: A systematic approach to the representation of knowledge about dynamical systems. Technical Report LITH-IDA-R-92-30, Department of Computer and Information Science, Linköping University, September 1992. Second Review Version.

[26] E. Sandewall. The range of applicability of nonmonotonic logics for the inertia problem. In *Proc. Int'l Joint Conf. on Artificial Intelligence, (IJCAI-93)*, 1993.

[27] E. Sandewall. *Features and Fluents: A Systematic Approach to the Representation of Knowledge about Dynamical Systems*. Oxford University Press, 1994.

[28] E. Sandewall and Y. Shoham. Nonmonotonic temporal reasoning. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Epistemic and Temporal Reasoning*, volume 4 of *Handbook of Artificial Intelligence and Logic Programming*. Oxford University Press, 1994.

[29] M. Thielscher. Ramification and causality. Technical Report TR-96-003, International Computer Science Institute, 1996.