Linköping Electronic Articles in
Computer and Information Science
Vol. 3(1998): nr 015

TAL: Temporal Action Logics Language Specification and Tutorial

Patrick Doherty
Joakim Gustafsson
Lars Karlsson
Jonas Kvarnström

Department of Computer and Information Science
Linköping University
Linköping, Sweden

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/1998/015/

Published on October 1, 1998 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

Typeset by the author using LaTeX
Formatted using étendu *style*

Recommended citation:
<Author>. <Title>. Linköping Electronic Articles in Computer and Information Science, Vol. 3(1998): nr 015. http://www.ep.liu.se/ea/cis/1998/015/. *October 1, 1998.*

This URL will also contain a link to the author's home page.

Abstract

The purpose of this article is to provide a uniform, lightweight language specification and tutorial for a class of temporal logics for reasoning about action and change that has been developed by our group during the period 1994-1998. The class of logics are collected under the name TAL, an acronym for *Temporal Action Logics.* TAL has its origins and inspiration in the work with Features and Fluents (FF) by Sandewall, but has diverged from the methodology and approach through the years. We first discuss distinctions and compatibility with FF, move on to the lightweight language specification, and then present a tutorial in terms of an excursion through the different parts of a relatively complex narrative defined using TAL. We conclude with an annotated list of published work from our group. The article tries to strike a reasonable balance between detail and readability, making a number of simplifications regarding narrative syntax and translation to a base logical language. Full details are available in numerous technical reports and articles which are listed in the final section of this article.

1 Introduction

The purpose of this article is to provide a lightweight language specification and brief tutorial for a class of narrative-based temporal action logics with acronym TAL standing for *Temporal Action Logics.* Work with TAL ensued in the early 90's with the first publications in 1994. Work with the TAL class of logics originated with, and was inspired by, research on the meta-theory of action and change initiated by Sandewall and described in the monograph [28]. A concurrent article by Sandewall [29] summarizing previous work on the meta-theory of reasoning about action and change and extensions and new results since then may be read as a companion to this article. Sandewall calls the generalized framework *Action Systems Metatheory* (ASM).

TAL currently has the following in common with the ASM framework:

- The distinction between the surface and base languages where the surface language should be viewed as a macro-language which provides a convenient high-level notation for describing narratives.
- The basis for the definitions of preferential entailment used in the TAL class of logics is primarily that of PMON, one of the definitions of preferential entailment proposed in Sandewall [28], which makes use of the occlusion predicate.
- The use of filtered preferential entailment techniques, also originating in Sandewall [27, 28].
- The solution to the frame problem, incomplete specification of states and non-determinism is basically the same as in PMON [28].

The TAL framework differs from the ASM framework in the following respects:

- Our starting point is the use of two formal languages, a surface language $\mathcal{L}(\mathrm{ND})$ for representing narratives, a base language $\mathcal{L}(\mathrm{FL})$ which is an order-sorted 1st- or 2nd-order classical language with the standard semantics (see Section 4), and a translation function which translates narratives described in $\mathcal{L}(\mathrm{ND})$ into a theory in $\mathcal{L}(\mathrm{FL})$. The dissimilarity with Features and Fluents [28] is primarily due to the use of different base languages, the addition of new statement classes in narratives, and new and powerful extensions to the surface language. Sandewall has recently begun using a sorted 1st-order logic in ASM and has adopted a number of operators originating in $\mathcal{L}(\mathrm{ND})$.
- A logic in the TAL class has <u>both</u> a proof and model theory based on classical 1st or 2nd-order logic.[1] Generally, the 2nd-order nature of the logic is related to the use of a circumscription axiom which reflects the nonmonotonic nature of the formalism, or the use of 2nd-order axioms used in the definition of temporal structures. The proof theory generated provides the basis for efficient implementations of the TAL logics or well-behaved fragments.
- A main part of the methodology used in defining or extending existing logics in TAL is to optimize away, when possible, the 2nd-order nature

[1] In some cases, the proof theory may be incomplete.

of narratives by using reduction algorithms or techniques such as quantifier elimination [9, 10]. This can result in the use of logically equivalent 1st-order theories which are amenable to classical theorem-proving techniques and optimizations.

- The use of assessment techniques via definitions of an underlying semantics, which is a cornerstone of the ASM approach, is de-emphasized, although not ruled out as unimportant.
- Work with TAL has resulted in broadening the expressibility of narratives, providing preliminary solutions to not only the frame problem, but also the ramification and qualification problems in an integrated manner for an expressive class of narrative descriptions. In addition progress has been made in the modeling of both concurrent actions and their interactions, and the use of delayed effects of actions.

Although the ASM and TAL approaches are not at all incompatible, they should be viewed as emphasizing different aspects associated with the larger problem of modeling action and change and using somewhat different methodologies and formal techniques. As in the past, we expect mutual migration of results between ASM and TAL in the future.

In broad outline, Sections 3–6 describe the core language definitions and underlying logic used in TAL, while Sections 2, 7 and the appendices provide a brief tutorial as to how to use the languages and logic for representing and reasoning about narratives in TAL.

The detailed outline is as follows. In Section 2, we outline the methodology and reasoning techniques used with TAL. In Section 3, the surface language $\mathcal{L}$(ND) is described. In Section 4, the base language $\mathcal{L}$(FL) is described. In Section 5, the translation from the surface language $\mathcal{L}$(ND) to the base language $\mathcal{L}$(FL) is described. In Section 6, the circumscription policy associated with the base language $\mathcal{L}$(FL) is described. In Section 7, we present a narrative example in TAL described using the surface language $\mathcal{L}$(ND). Each of the components in a narrative is described in individual subsections with comments intended to provide intuitions about both the syntax and syntactic restrictions for statement types, and their informal semantics. In Section 8, we briefly describe annotated references to previous work. In Appendix 1, the narrative considered in parts in Section 3 is listed in full. In Appendix 2, the translation into $\mathcal{L}$(FL) of the narrative in Appendix 1 is listed in full. In Appendix 3, a visualization of those facts true in the preferred models for the narrative are displayed as a timeline which has been generated using the VITAL research tool.

2 Reasoning about Narratives in TAL

The TAL methodology uses two languages for representing and reasoning about narratives. The surface language $\mathcal{L}$(ND) (Narrative Description Language) provides a convenient high-level notation for describing narratives. A narrative consists of a collection of labeled statements which partition the narrative into common sets of statements. Currently, a narrative consists of two parts, the *narrative background specification* and the *narrative specification.* Each part contains different component specifications, the sum of which describe both the static and dynamic aspects of a narrative.

The *narrative background specification* (NBS) consists of the following components:

- Narrative Type Specification – The NTS describes type descriptions for features, feature value domains, and type descriptions for actions.
- Action Type Specification – The ATS provides generic definitions of actions.
- Domain Constraint Specification – The DCS provides statements intended to represent static knowledge about logical dependencies between fluents generally true in every state.
- Dependency Constraint Specification – The DeCS provides statements intended to represent knowledge about directional dependencies between fluents true in states or across states.

The *narrative specification* (NS) consists of the following components:

- Observation statements – Observation statements are intended to represent values of fluents at specific timepoints.
- Action occurrence statements – Action occurrence statements are intended to represent a particular occurrence of an action during a particular duration. Both the timing between and the duration of actions can be incompletely specified.
- Schedule statements – A class of intermediary statements called *schedule statements* are generated from action type specifications during the translation process. Two alternative syntactic structures are used which depend on whether the logic in question treats actions as 1st-class citizens or not.

Neither the language $\mathcal{L}(\mathrm{ND})$ nor the statement labels have any formal semantics, although the labels are used in the specification of the circumscription policy applied to the base theory.

Given a narrative Υ, described using $\mathcal{L}(\mathrm{ND})$, a translation function is used to translate each statement in Υ into a 1st-order wff in $\mathcal{L}(\mathrm{FL})$, an order-sorted classical 1st- or 2nd-order logic. The 1st-order theory associated with Υ is augmented with a number of additional 1st-order (and sometimes 2nd-order) formulas representing unique-names and domain closure axioms, unique-value axioms, axioms for the temporal base structure, etc., all of a technical nature. In addition one or more circumscription axioms are added to provide solutions to the frame, ramification and qualification problems. Filtered circumscription is often used which circumscribes only some of the partitions of a narrative. In the logics we currently use, the 2nd-order circumscriptive theory expressing Υ in the base language $\mathcal{L}(\mathrm{FL})$ is reducible to a logically equivalent 1st-order theory using quantifier elimination techniques from Doherty et al. [9, 10] or variations of predicate completion techniques from Doherty [3, 5].

Specialized inference algorithms or standard proof theoretical techniques (or both) are then used to reason about the narrative. An on-line research tool called VITAL [24] is available on the WWW. The tool automates the process of translating and reducing narrative descriptions in $\mathcal{L}(\mathrm{ND})$ to 1st-order theories in $\mathcal{L}(\mathrm{FL})$ and includes a query mechanism and visualization of preferred models in the form of annotated time-lines.

3 The Surface Language $\mathcal{L}$(ND)

The surface language $\mathcal{L}$(ND) (Narrative Description Language) is a high level description language used to represent narrative descriptions.[2] The language has no formal semantics, but does have a formal syntax. It provides a convenient set of constructs and macro-operators which contribute to simplifying the representation of narrative descriptions and enhancing their readability. All formal reasoning is done after translating a narrative description described in $\mathcal{L}$(ND) into a theory in $\mathcal{L}$(FL).

3.1 Temporal Expressions

Assume that the base language $\mathcal{L}$(FL) uses the temporal base structure $\langle \mathcal{T}, c_1, \ldots, o_1, \ldots o_k, r_1, \ldots r_l \rangle$ (see Section 4). Temporal expressions are defined inductively using the variables, constants and operators associated with the temporal base structure. In the following, we use τ and τ' as meta-variables for temporal expressions.

3.2 Feature Expressions

The narrative type specification provides a description of feature names, feature value domains and feature value sorts and sub-sorts. Elements in the value domains may be used as constants in feature expressions and it is assumed that additional non-designated constants and variables of the proper sorts are available for use in narrative descriptions.

Feature expressions are defined inductively in a manner similar to the definition of typed function terms in an order-sorted classical logic. In the following, we use f and f' as meta-variables for feature expressions.

3.3 Value Expressions

Value expressions are used as arguments in composite feature expressions and on the right hand side of the *isvalue* macro-operator "$\hat{=}$". The following value expressions are used:

- Variables and designated and non-designated constants associated with the feature value domains described in the narrative type specification.
- $value(\tau, f)$ – denotes the value of the feature expression f at time τ.

In the following, we use v and v' as meta-variables for value expressions.

3.4 Atomic Expressions

The following atomic expressions are used in the construction of narrative statements:

[2] In previous papers we used $\mathcal{L}$(SD) to denote this language.

- $[\tau]f \hat{=} v$ – A *temporal isvalue expression* states that the feature f has the value v at time τ.
- $X([\tau]f \hat{=} v)$ or $X([\tau]f)$ – An *occlusion expression* states that the feature f is *occluded* at time τ.
- $f = f'$ – A *feature equality expression* states that two feature expressions refer to the same feature.
- $v = v'$ – A *value equality expression* states that two value expressions refer to the same value.
- $\tau \otimes \tau'$ – A *temporal equality or relational expression* (where $\otimes$ is "=" or one of the relation symbols defined in the temporal base structure) states that two temporal expressions refer to the same timepoint or have a particular relational correspondence.
- $[\tau, \tau']A(\overline{\omega})$ – An *action occurrence expression* states that action A, with arguments $\overline{\omega}$, occurs during the interval $[\tau, \tau']$.

3.5 Narrative Statements

Narrative statements are defined inductively in a manner similar to the definition of well-formed formulas in order-sorted classical logic using the standard connectives, quantifiers and notational conventions. The exception is that atomic expressions are used instead of atomic formulas as the basis for the inductive definition.

Narrative statements are always labeled in a narrative description. Normally, a set of syntactic restrictions are associated with each labeled class of statements (See Section 7.6). The labeling is also used to partition the corresponding wffs in $\mathcal{L}$(FL) into appropriate classes required for the specification of a filtered circumscription policy (See Section 6).

In the following, we use α, β and γ as meta-variables for narrative statements or sub-statements. We will also continue to use the term "narrative statement" for those statements which include the operators and abbreviations discussed in the next section.

3.6 Additional Macro-Operators and Abbreviations

The surface language $\mathcal{L}$(ND) has a rich collection of macro-operators and abbreviations. In this section, we briefly describe several of the more widely used abbreviations and refer the reader to Doherty et al. [5, 21, 14] for the details.

3.6.1 Temporal Intervals and Composite Isvalue Expressions

- Temporal isvalue expressions are generalized to allow boolean operators inside the temporal scope and the temporal scope is generalized to allow closed, open, or semi-open temporal intervals.
  - For example, $[\tau](f \hat{=} v \otimes f' \hat{=} v')$ is an abbreviation for the expression $[\tau]f \hat{=} v \otimes [\tau]f' \hat{=} v'$, where $\otimes \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

  - For example, $[\tau, \tau'](f \hat{=} v \otimes f' \hat{=} v')$ is an abbreviation for the expression $\forall t.\ \tau \leq t \leq \tau' \rightarrow [t](f \hat{=} v \otimes f' \hat{=} v')$.

- Features with boolean value sorts may abbreviate the use of "$\hat{=}$".
  - For example, $[\tau](f \otimes \neg f')$ is an abbreviation for the expression $[\tau](f \hat{=} \mathsf{true} \otimes f' \hat{=} \mathsf{false})$.

Note that the abbreviations defined above do not necessarily apply in the scopes of the X, R and I operators defined next.

3.6.2 Special Primitive and Macro-Operators

The following operators are used in the consequents of action type and dependency constraint statements:

- The occlusion operator X is a primitive operator used to specify at what timepoints fluents are exempt from the persistence or inertia assumption associated with the frame problems. For example,
  - $X((\tau, \tau']\alpha)$ – is an abbreviation for $\forall t.\ \tau < t \leq \tau' \rightarrow X([t]\alpha)$
  - $X([\tau]\alpha)$ – is translated into a conjunction $Occlude(\tau, f_1) \wedge \ldots \wedge Occlude(\tau, f_n)$ in $\mathcal{L}(\mathrm{FL})$, where each f_i occurs on the left hand side of an isvalue expression $f_i \hat{=} v_i$ in α.
- The reassignment operator R is a macro-operator used to specify explicit reassignment of a feature value to a fluent at a particular timepoint. For example,
  - $R((\tau, \tau']\alpha)$ – is an abbreviation for $X((\tau, \tau']\alpha) \wedge [\tau']\alpha$.
  - $R([\tau]\alpha)$ – is an abbreviation for $X([\tau]\alpha) \wedge [\tau]\alpha$.
- The durational reassignment operator I is a macro-operator normally used in conjunction with durational fluents and reassigns the default feature value of the durational fluent temporarily to a new value during a specified duration. For example,
  - $I((\tau, \tau']\alpha)$ – is an abbreviation for $X((\tau, \tau']\alpha) \wedge (\tau, \tau']\alpha$.
  - $I([\tau]\alpha)$ – is an abbreviation for $X([\tau]\alpha) \wedge [\tau]\alpha$.

The following macro-operators are generally found in the antecedents of dependency constraints and are useful for expressing trigger conditions for such constraints:

- $C_T([\tau]\alpha)$ – is an abbreviation for $([\tau - 1]\neg\alpha \wedge [\tau]\alpha)$.
- $C_F([\tau]\alpha)$ – is an abbreviation for $([\tau - 1]\alpha \wedge [\tau]\neg\alpha)$.
- $C([\tau]\alpha)$ – is an abbreviation for $(C_T([\tau]\alpha) \vee C_F([\tau]\alpha))$.

The following macro-operators determine the type of feature and the persistence policy employed:

- $Per(\tau, f)$ – is an abbreviation for $\forall v.\ (C([\tau]f \hat{=} v) \rightarrow X([\tau]f))$.
- $Dur(\tau, f, v)$ – is an abbreviation for $[\tau]\neg f \hat{=} v \rightarrow X([\tau]f)$.

The distinction between feature types originates in Karlsson and Gustafsson [21], and has proven to be a very useful concept. Originally, the macro-operators were translated into specific predicates *Per* and *Dur* in $\mathcal{L}$(FL). Sandewall [29] observes that this is not always required and we use his translation above. For more details regarding alternative translations, see Section 6.1.

The following macro-operators are also sometimes used:

- $[\tau]\alpha \gg [\tau']\beta$ – is an abbreviation for $[C_T([\tau]\alpha) \rightarrow X([\tau']\beta)] \wedge [[\tau]\alpha \rightarrow [\tau']\beta]$
- $[\tau, \tau']f := v$ – is an abbreviation for $R(([\tau, \tau']f \hat{=} v)$.

Note that both the syntactic restrictions placed on narrative statements and the definitions for abbreviations and macro-operators may vary somewhat relative to the choice of temporal base structure used in a particular logic. For example, the abbreviation for $C_T([\tau]\alpha)$ would be defined as

$$([\tau - 1]\neg\alpha \wedge [\tau]\alpha)$$

for integer time, whereas it would be defined as

$$(\forall t.\ \tau = t + 1 \rightarrow [t]\neg\alpha) \wedge [\tau]\alpha$$

for natural time (0 and the positive integers). In the latter case, one must deal with the boundary condition associated with the initial point 0 on the timeline.

4 The Base Logic and Language $\mathcal{L}$(FL)

$\mathcal{L}$(FL) (Fluent Logic Language) is an order-sorted 1st-order language. The base logic associated with $\mathcal{L}$(FL) is an order-sorted 1st-order predicate logic with equality using a standard semantics. An order-sorted logic is simply a generalization of a many-sorted logic where sorts may be sub-sorts of others. On occasion, we use 2nd-order logic in a restricted manner, primarily for formulating circumscription axioms. The translation of a narrative description from $\mathcal{L}$(ND) to $\mathcal{L}$(FL) always results in a finite set of 1st-order wffs. A set of foundational axioms is associated with each narrative theory and may contain 2nd-order formulas related to the axiomatization of temporal base structures such as the natural numbers or integers. Due to the use of 2nd-order logics, we often use $\mathcal{L}$(FL) to refer to both the logic and language whether it is 1st- or 2nd-order.

For any TAL logic, the following predicates, or subsets of these predicates, are included in the language signature:

- $\mathit{Holds}(t, f, v)$, where t, f, and v are of sort time, fluent, and fluentval, respectively.
- $\mathit{Occlude}(t, f)$, where t and f are of sort time and fluent, respectively.
- $\mathit{Occurs}(t, t', a)$, where t, t', and a are of sort time, time, and action, respectively.
- $\mathit{Observe}(t, f, v)$, where t, f, and v are of sort time, fluent, and fluentval, respectively.

- $Dur(t, f, v)$, where t, f, and v are of sort time, fluent, and fluentval, respectively.
- $Per(t, f)$, where t and f are of sort time and fluent, respectively.

For any TAL logic, the following function is included in the language signature:

- $val(t, f)$, where t and f are of sort time and fluent, respectively. The type description for val is $(\mathsf{time}, \mathsf{fluent}) \rightarrow \mathsf{fluentval}$.

For each particular class of narratives considered, the following functions and sorts are used:

- For each domain d_i declared in a narrative type specification, a new sort s_i is included in the language signature together with a designated function symbol of arity 0 (i.e. a constant symbol) for each object in the domain with type description $() \rightarrow s_i$. Additional non-designated constant symbols of sort s_i are included in the signature when appropriate. All sorts s_i are sub-sorts of fluentval.
- For each feature f_i declared in a narrative type specification, a function symbol of the proper arity and its associated type description are included in the language signature.
- It is assumed that each logic specifies a temporal base structure $\langle \mathcal{T}, c_1, \dots, o_1, \dots o_k, r_1, \dots r_l \rangle$ consisting of a temporal domain $\mathcal{T}$, 0 or more constants $c_1 \dots$, a finite set of operators $o_1, \dots o_k$, and a finite set of relations $r_1, \dots r_l$. Given a temporal base structure, a new sort and additional constant and function symbols of that sort, denoting the objects in the structure, are added to the language signature.

The number and type of arguments to predicates and functions in the signature may vary from logic to logic. As a general policy, we prefer the use of macro-operators and abbreviations in $\mathcal{L}(\mathrm{ND})$ to the use of additional predicates in $\mathcal{L}(\mathrm{FL})$. For example, we already mentioned the use of macros for *Dur* and *Per* as an alternative to predicates.

5 Translating from $\mathcal{L}(\mathrm{ND})$ to $\mathcal{L}(\mathrm{FL})$

Translating narrative statements in $\mathcal{L}(\mathrm{ND})$ to formulas in $\mathcal{L}(\mathrm{FL})$ is a straightforward, but tedious process. In this section, we will consider the basic idea and refer the reader to previous references described in Section 8 which provide the necessary technical definitions and algorithms. In the following, we assume the existence of a translation function *Trans*() which takes an expression in $\mathcal{L}(\mathrm{ND})$ and returns one in $\mathcal{L}(\mathrm{FL})$. The following translations are the most important:

- $Trans([\tau] f \hat{=} v) = Holds(\tau, f, v)$.
- $Trans([\tau] \neg \alpha) = \neg Trans([\tau]\alpha)$.
- $Trans([\tau]\alpha \otimes \beta) = Trans([\tau]\alpha) \otimes Trans([\tau]\beta)$,where $\otimes \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
- $Trans([\tau]\mathcal{Q}x[\alpha]) = \mathcal{Q}x.Trans([\tau]\alpha)$, where $\mathcal{Q} \in \{\forall, \exists\}$.

- $\mathit{Trans}(X([\tau]f \hat{=} v)) = \mathit{Occlude}(\tau, f)$.
- $\mathit{Trans}(X([\tau]\neg\alpha)) = \mathit{Trans}(X([\tau]\alpha))$
- $\mathit{Trans}(X([\tau]\alpha \otimes \beta)) = \mathit{Trans}(X([\tau]\alpha)) \wedge \mathit{Trans}(X([\tau]\beta))$, where $\otimes \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
- $\mathit{Trans}(X([\tau]\mathcal{Q}x[\alpha])) = \forall x.\ \mathit{Trans}(X([\tau]\alpha))$, where $\mathcal{Q} \in \{\forall, \exists\}$.
- $\mathit{Trans}(\mathit{value}(\tau, f)) = \mathit{val}(\tau, f)$.
- $\mathit{Trans}([\tau, \tau']A(\overline{w})) = \mathit{Occurs}(\tau, \tau', A(\overline{w}))$, where $[\tau, \tau']A(\overline{w})$ is an action occurrence statement in $\mathcal{L}(\mathrm{ND})$.

6 The Circumscription Policy

In this section, we assume familiarity with circumscription [26], and refer the reader to Łukaszewicz [25] for details. In the following, we use

- Υ – to denote the collection of narrative statements contained in a narrative in $\mathcal{L}(\mathrm{ND})$ and Υ_{per}, Υ_{obs}, Υ_{occ}, Υ_{scd}, Υ_{domc}, Υ_{depc}, to denote the set of persistence, observation, action occurrence, schedule, domain constraint, and dependency constraint statements in Υ, respectively.
- Γ – to denote the collection of narrative formulas in $\mathcal{L}(\mathrm{FL})$ corresponding to the translation of the collection of narrative statements in Υ and Γ_{obs}, Γ_{occ}, Γ_{scd}, Γ_{domc}, Γ_{depc}, to denote the corresponding sets of observation, action occurrence, schedule, domain constraint, and dependency constraint formulas in Γ, respectively.
- Γ_{fnd} – to denote the set of foundational axioms in $\mathcal{L}(\mathrm{FL})$ which contain unique names axioms, unique values axioms, possibly domain closure axioms, etc. Γ_{fnd} is always conjoined with the narrative theory Γ.
- Γ_{time} – to denote the set of axioms representing the particular temporal base structure being used. Γ_{time} is always conjoined with the narrative theory Γ.
- Γ_{per} – to denote the set of persistence axioms characterizing the behavior of persistent and durational fluents. Γ_{per} is always conjoined with the narrative theory Γ.

Given a narrative Υ in $\mathcal{L}(\mathrm{ND})$, the corresponding theory Δ in $\mathcal{L}(\mathrm{FL})$ is

$$\Gamma \wedge \Gamma_{fnd} \wedge \Gamma_{time} \wedge \Gamma_{per},$$

where

$$\Gamma = \Gamma_{obs} \wedge \Gamma_{occ} \wedge \Gamma_{domc} \wedge \Gamma_{depc} \wedge \Gamma_{scd}.$$

Given the corresponding theory in $\mathcal{L}(\mathrm{FL})$, we apply a filtered circumscription policy which results in the following circumscribed theory Δ',

$$\Gamma' \wedge \Gamma_{fnd} \wedge \Gamma_{time} \wedge \Gamma_{per},$$

where

$$\Gamma' = \Gamma_{obs} \wedge \Gamma_{domc} \wedge Circ(\Gamma_{occ}; Occurs) \wedge Circ(\Gamma_{depc} \wedge \Gamma_{scd}; Occlude).$$

For this particular policy, we assume actions are 1st-class citizens in the object language.[3] The policy circumscribes the predicate *Occlude* relative to the set of schedule and dependency constraint formulas and leaves all other predicates fixed. It is easily shown that the 2nd-order formula $Circ(\Gamma_{depc} \wedge \Gamma_{scd}; Occlude)$ is reducible to a logically equivalent 1st-order formula, since the predicate *Occlude* only occurs positively in $\Gamma_{depc} \wedge \Gamma_{scd}$. In addition, *Occurs* only occurs positively in Γ_{occ}. It is also easily shown that $Circ(\Gamma_{occ}; Occurs)$ is also reducible to a logically equivalent 1st-order formula. In fact, simple applications of predicate completion generally suffice.

Consequently, provided the temporal axiomatization Γ_{time} is first-order, the theory Δ' is too, which makes it amenable to existing theorem-proving techniques and optimizations for 1st-order logic. In the case that Γ_{time} is 2nd-order, one can still apply specialized theorem proving techniques such as constraint logic programming techniques which provide sound, but incomplete proof techniques.

6.1 Intuitions

The intuition behind the choice of circumscription policy is relatively straightforward. Γ_{scd} provides a specification of the sufficient conditions for when the fluents which comprise the effects of actions should be exempted from a background persistence or inertia assumption and what their new values should be. In a similar manner, Γ_{depc} provides a specification of the sufficient conditions for when fluents can potentially change value and what their new values should be. Note that, due to nondeterminism, the actual value a fluent takes may be constrained by other parts of the narrative and not just the direct action effects and dependency constraints.

Given the sufficient conditions for occlusion (exemption from a persistence assumption), the circumscription of *Occlude* over $\Gamma_{scd} \wedge \Gamma_{depc}$ constrains the sufficient conditions to be the necessary conditions. Given that we now have a definition for potential change and constraints for determining the values of fluents permitted to change value, the filtering condition Γ_{per} serves to filter away all spurious change (i.e. change not specified via *Occlude*). The filtering condition used is based on the type of feature, that is, whether it is a persistent or durational feature.[4] Recall the definition of macro-operators *Per* and *Dur* in Section 3.6.2.

- $Per(\tau, f)$ – is an abbreviation for $\forall v.\ (C([\tau]f \hat{=} v) \rightarrow X([\tau]f))$.
- $Dur(\tau, f, v)$ – is an abbreviation for $[\tau]\neg f \hat{=} v \rightarrow X([\tau]f)$.

[3] In the case where actions are not 1st-class citizens, the *Occurs* predicate is not used and neither Γ_{occ} nor $Circ(\Gamma_{occ}; Occurs)$ is required in Δ and Δ', respectively (See Section 7.5.1).

[4] In older versions of TAL, we have used one filtering axiom call the *nochange* axiom where all features behaved as persistent fluents.

Normally, for each feature $\mathsf{f}(\overline{\omega})$ in a narrative, we associate a corresponding set of persistence statements in Υ_{per} with one of the two following forms:[5]

- $\mathcal{Q}(t,\overline{\omega}).\ \alpha \rightarrow Per(\tau, \mathsf{f}(\overline{\omega}))$ or
- $\mathcal{Q}(t,\overline{\omega}).\ \alpha \rightarrow Dur(t, \mathsf{f}(\overline{\omega}), v)$,

where $\mathcal{Q}(t,\overline{\omega}).\ \alpha$ is an expression which conditionalizes at what timepoints and with what arguments the feature $\mathsf{f}(\overline{\omega})$ will behave as a persistent or a durational fluent.

In the simple case, a feature is either durational with a single default value at all timepoints and arguments or persistent at all timepoints and arguments, but not both. For example, if f is declared as a persistent feature then the following persistence statement is used:

$$\forall \tau, \overline{\omega}.\ \mathit{True} \rightarrow Per(\tau, \mathsf{f}(\overline{\omega})),$$

whose translation in $\mathcal{L}$(FL) after simplification is[6]

$$\forall t, \overline{w}, v.\ (\mathit{Holds}(t-1, \mathsf{f}(\overline{w}), v) \oplus \mathit{Holds}(t, \mathsf{f}(\overline{w}), v)) \rightarrow \mathit{Occlude}(t, \mathsf{f}(\overline{w})).$$

If $\mathsf{f}(\omega)$ is declared as a durational feature with default value v then the following persistence statement is used:

$$\forall \tau, \overline{\omega}.\ \mathit{True} \rightarrow Dur(\tau, \mathsf{f}(\overline{\omega}), \mathsf{v}),$$

whose translation in $\mathcal{L}$(FL) is

$$\forall t, \overline{w}.\ \neg \mathit{Holds}(t, \mathsf{f}(\overline{w}), \mathsf{v}) \rightarrow \mathit{Occlude}(t, \mathsf{f}(\overline{w})).$$

For each persistence statement in $\mathcal{L}$(ND), the corresponding formula is added to Γ_{per}. These formulas constrain potential change in a theory to those timepoints where fluents are explicitly occluded based on specific persistence policies per fluent. Note that one may even avoid declaring a feature as either durational or persistent. In that case, it behaves as a *dynamic fluent* which is never subject to any persistence or inertia assumption.

The distinction between persistent and durational fluents originates in Karlsson and Gustafsson [21] and is used in Doherty and Kvarnström [7] to deal with qualifications of actions. In those papers, the macros were translated into formulas using a *Per* and *Dur* predicate. In some cases this approach may still be useful and provides additional expressivity. Sandewall [29] observed that one could do without the explicit predicates and use macros as we have described here. The only distinction is that we conditionalize the specification of feature types where a feature can behave as a durational, persistent or dynamic fluent in the same narrative. Durational fluents provide a limited form of default reasoning in TAL. Dynamic fluents were first discussed in Doherty [5]. Persistent fluents have generally been the rule for fluent behavior in mainstream research.

[5] In more complicated cases, we allow other forms for persistence statements when representing persistence policies for fluents.

[6] The symbol "$\oplus$" denotes exclusive-or.

7 A Narrative Example in TAL

In this section, we will represent a relatively complex narrative in the surface language $\mathcal{L}$(ND). This particular narrative is intended to demonstrate both the basic structure of a narrative and a subset of the representational power the language provides for constructing narrative descriptions. We will begin with an informal natural language description of a narrative whose proper formal representation would have to assume solutions to the frame, qualification and ramification problems, the use of both boolean and non-boolean fluents, concurrent actions, and domain and dependency constraints. Although the language provides additional expressivity such as context dependent and non-deterministic effects of actions, the use of temporal variables and timing constraints, and interacting concurrent actions, this particular narrative does not use that expressivity.

The following narrative, which we call the *Russian Hijacking Scenario*, appears in Doherty and Kvarnström [7].[7] The formal counterpart to the narrative can be translated into a circumscribed theory in the language $\mathcal{L}$(FL), reduced to its first-order equivalent, and queried for inferences which follow from the narrative theory using the on-line tool VITAL [24]. We encourage the reader to try our reasoning tool.

The story begins as follows:

A Russian businessman, Vladimir, travels a lot and is concerned about both his hair and safety. Consequently, when traveling, he places both a comb and a gun in his pocket. A Bulgarian businessman, Dimiter, is less concerned about his hair, but when traveling by air, has a tendency to drink large amounts of vodka before boarding a flight to subdue his fear of flying. A Swedish businessman, Erik, travels a lot, likes combing his hair, but is generally law abiding. Now, one ramification of putting an object in your pocket is that it will follow with you as you travel from location to location. Generally, when boarding a plane, the only preconditions are that you are at the gate and you have a ticket. One *possible* qualification to the boarding action is if you arrive at the gate in a sufficiently inebriated condition, as will be the case for Dimiter. A ramification that may in some cases play a dual role as a qualification to the boarding action is if you try to board a plane with a gun in your pocket, which may be the case for Vladimir. Now, Vladimir, Erik and Dimiter, start from home, stop by the office, go to the airport, and try to board flight SAS609 to Stockholm. Both Erik and Vladimir put combs in their pockets at home, Vladimir picks up a gun at the office, while Dimiter is already drunk at home.

Given the narrative, we would like to answer questions such as "Who will successfully board the plane?" "What are their final locations?" "What is in their pockets after attempting to board the plane and after the plane has arrived at its destination?".

In the following sections, we will consider the components of a formal narrative in the surface language $\mathcal{L}$(ND). The full narrative description can be found in Appendix 1.

[7]The narrative is an elaboration and concretization of a sketch suggested by Vladimir Lifschitz in recent on-line discussions in the Electronic Transactions on Artificial Intelligence (ETAI/ENAI).

7.1 Narrative Type Specification

We first introduce a specification of feature value domains and type descriptions. The type specification in $\mathcal{L}$(ND) [8] is directly translatable into the order-sorted logic language $\mathcal{L}$(FL).

TYPE SPECIFICATION

```
domain thing = {gun, comb1, comb2, comb3,
                vladimir, dimiter, erik, sas609}
domain location = {home1, home2, home3, office,
                   airport, run609, run609b, air}
domain pocket = {pocket1, pocket2, pocket3}
domain runway = location [run609, run609b]
domain airplane = thing [sas609]
domain person = thing [vladimir - erik]
domain pthing = thing [gun, comb1 - comb3]

feature loc(thing): location
feature inpocket(person,pthing): boolean
feature drunk(person): boolean
feature onplane(airplane,person): boolean
feature poss_board(person,airplane): boolean

action put(person, pthing, pocket)
action travel(person, location, location)
action fly(airplane, runway, runway)
action board(person, airplane)
```

This particular narrative has 7 user-defined sorts (and a pre-defined boolean sort), the last 4 of which are sub-sorts constructed from the first 3 sorts. For example, the sub-sort pthing,

```
domain pthing = thing [gun, comb1 - comb3]
```

is a sub-sort of thing and contains gun, comb1, comb2, and comb3.

There are 5 features, 4 of which are boolean and 1 of which has the non-boolean sort location. The feature specifications describe both the feature type and the types of the feature arguments. For example, the feature specification

```
feature loc(thing): location
```

specifies that the non-boolean fluent associated with the feature loc returns a value of sort location.

There are 4 action specifications of sort action. For example, the action specification

```
action put(person, pthing, pocket)
```

has 3 arguments and is the action type for the action of a person putting a thing in his or her pocket. Whether the action types have respective constant or function terms in the order-sorted vocabulary of $\mathcal{L}$(FL) and

[8] The type specification syntax used below is taken directly from the VITAL research tool syntax. The actual $\mathcal{L}$(ND) syntax may vary somewhat.

action is introduced as a sort, will be dependent on whether actions are viewed as *first-class* objects in the particular version of the TAL logic used.

7.1.1 Feature Types

In some of our work, we have noticed that it is useful to classify features according to the type of persistence assumption associated with a feature's fluent behavior. Currently, features are classified as being *dynamic*, *persistent* or *durational*. A dynamic fluent is not subject to any persistence constraints whatsoever. Both persistent and durational fluents are subject to persistence constraints, but differ somewhat in values assigned. When a persistent fluent changes value, the assumption is that it retains the new value until there is reason to change it again. On the other hand, a durational fluent has a specific default value. When a durational fluent changes value, the assumption is that it will keep that exceptional value for a specified interval and when that interval ends, re-acquire its default value. Generally, durational fluents are used together with the I operator to specify the exceptional value interval.

Each feature has a persistence policy associated with it, represented in the form of a feature specification and a set of 0 or more persistence statements characterizing when and what persistence behavior a feature has. A persistence statement has the label per in $\mathcal{L}(\mathrm{ND})$. For each feature specified, a set of *persistence statements* is provided and placed in Υ_{per} which is then translated to a set of formulas in Γ_{per} in $\mathcal{L}(\mathrm{FL})$.

Durational Feature Specification
For example, the feature poss_board is specified as a durational feature with default value true, where the feature specification is

```
feature poss_board(person,airplane): boolean
```

and the persistence policy is specified using one persistence statement

$$\mathsf{per} \qquad \forall t, \overline{w}.\ \mathit{True} \rightarrow \mathit{Dur}(t, \mathsf{poss\_board}(\overline{w}), \mathsf{true}).$$

The persistence policy expresses that the feature poss_board has a persistent default value of true, and functions as a durational fluent for all timepoints and arguments specified by the conditionalizer expression $\mathcal{Q}(t, \overline{\omega}).\ \alpha$ which is "$\forall t, \overline{\omega}. \mathit{True}$". Generally, a default conditionalizer similar to the above is used, but one may want to limit a fluent's durational behavior to selected parts of the timeline, or for selected arguments and different values. In this case, one would add additional persistence statements to Υ_{per}.

Persistent Feature Specification
For example, the feature loc is specified as a persistent feature, where the feature specification is

```
feature loc(thing): location
```

and the persistence policy is specified using one persistence statement

$$\mathsf{per} \qquad \forall t, \overline{w}.\ \mathit{True} \rightarrow \mathit{Per}(t, \mathsf{loc}(\overline{w})).$$

The translation of the macro-operators *Per* and *Dur* and additional intuitions are provided in Sections 3.6.2 and 6.1.

Dynamic Feature Specification
In the default case, a persistence policy for a dynamic feature consists of a feature specification and an empty set of persistence statements.

Before considering the specific narrative describing action occurrences and observations, we will first specify the background knowledge associated with the particular world domain to which the narrative belongs. This knowledge consists of the *action types*, the *domain constraints*, and the *dependency constraints*.

7.2 Action Type Specification

The *action type* specifications provide generic definitions of actions. The following 4 action types are used in the narrative:

acs1 $[t_1, t_2]\ \mathsf{fly}(airplane, runway_1, runway_2) \rightsquigarrow$
$[t_1]\ \mathsf{loc}(airplane) \hat{=} runway_1 \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(airplane) \hat{=} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(airplane) \hat{=} runway_2)$

acs2 $[t_1, t_2]\ \mathsf{put}(person, pthing, pocket) \rightsquigarrow$
$[t_1]\ \mathsf{loc}(person) \hat{=} value(t_1, \mathsf{loc}(pthing)) \rightarrow$
$R((t_1, t_2]\ \mathsf{inpocket}(person, pthing))$

acs3 $[t_1, t_2]\ \mathsf{travel}(person, loc_1, loc_2) \rightsquigarrow [t_1]\ \mathsf{loc}(person) \hat{=} loc_1 \rightarrow$
$R([t_2]\ \mathsf{loc}(person) \hat{=} loc_2)$

acs4 $[t_1, t_2]\ \mathsf{board}(person, airplane) \rightsquigarrow$
$[t_1]\ \mathsf{poss\_board}(person, airplane) \wedge \mathsf{loc}(person) \hat{=} \mathsf{airport} \rightarrow$
$R([t_2]\ \mathsf{loc}(person) \hat{=} value(t_2, \mathsf{loc}(airplane)) \wedge$
$\mathsf{onplane}(airplane, person))$

An action type specification consists of an action type name specification which includes the number and sort of arguments to the action, and the action type definition which consists of the preconditions, postconditions, and any durational constraints which may limit the behavior of an instance of the action type during its particular duration. Let's take a closer look at the action type specification acs1:

acs1 $[t_1, t_2]\ \mathsf{fly}(airplane, runway_1, runway_2) \rightsquigarrow$
$[t_1]\ \mathsf{loc}(airplane) \hat{=} runway_1 \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(airplane) \hat{=} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(airplane) \hat{=} runway_2)$

An action type specification can be viewed as a schema which will be instantiated and translated into logical formulas in $\mathcal{L}(\mathrm{FL})$. The meta-variables, t_1, t_2, *airplane*, $runway_1$, and $runway_2$ of sort time, airplane, and runway, will be instantiated with temporal or fluent terms of the proper sort via action occurrence statements in the narrative.

The *action type name* statement of acs1 is

acs1 $[t_1, t_2]\ \mathsf{fly}(airplane, runway_1, runway_2)$

In a narrative, action occurrences are simply instantiations of an action type name statement and the duration in which it occurs.

The action type definition of acs1 is
$[t_1]\ \mathsf{loc}(airplane) \hat{=} runway_1 \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(airplane) \hat{=} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(airplane) \hat{=} runway_2)$

where

$[t_1]\ \mathsf{loc}(airplane) \hat{=} runway_1$

is the precondition to the action and

$I((t_1, t_2)\ \mathsf{loc}(airplane) \hat{=} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(airplane) \hat{=} runway_2)$

is the postcondition to the action. Note the use of both the I and the R macro operator in the postcondition which states that the location of the airplane is in the air throughout the whole duration (t_1, t_2), of the action excluding the endpoints t_1 and t_2, while the location of the airplane at the end of the action t_2 is at runway2.

7.3 Domain Constraint Specification

Domain constraints are intended to represent static knowledge about the world true in every state. Any constraints placed on features or sets of features are solely due to the logical form of the domain constraint statements. No use of assignment or occlusion macros are permitted (e.g. $:=$, $\gg$, X, R, I). Causal or directional dependencies between features are specified using *dependency constraints.* The domain constraints used in the current narrative are:

//A pthing cannot be in two pockets at the same time.
dom1 $\forall t, pthing_1, person_1, person_2\ [\neg(person_1 = person_2) \wedge$
$[t]\ \mathsf{inpocket}(person_1, pthing_1) \rightarrow [t]\ \neg\mathsf{inpocket}(person_2, pthing_1)]$

//A person cannot be on board two airplanes at the same time.
dom2 $\forall t, person_1, airplane1, airplane2\ [\neg(airplane1 = airplane2) \wedge$
$[t]\ \mathsf{onplane}(airplane1, person_1) \rightarrow [t]\ \neg\mathsf{onplane}(airplane2, person_1)]$

Any variables or terms used in domain constraints are assumed to be of the proper sort and are preserved in the translation to $\mathcal{L}(\mathrm{FL})$.

7.4 Dependency Constraint Specification

Dependency constraints are intended to represent knowledge about directional dependencies between features that are more constrained than the logical dependencies represented via domain constraints. The main syntactic difference between domain and dependency constraints is that the use of reassignment and occlusion operators is allowed (e.g. $:=$, $\gg$, R, I, X). This, of course makes sense since occlusion together with a filtered circumscription policy is used to enforce directionality of change between features. Dependency constraints are often used to represent causal dependencies between fluents but can also be used in a more general manner.

The most common structure for a dependency constraint statement is via the use of the macro $\gg$ where, assuming that α and β are legal statements and τ and τ' are temporal terms,

$$[\tau]\alpha \gg [\tau']\beta$$

denotes the dependency constraint which states that if α changes value from false to true from $\tau - 1$ to τ then β will become true at τ'. Recall that $[\tau]\alpha \gg [\tau']\beta$ has an equivalent expansion as

$$([\tau - 1]\neg\alpha \wedge [\tau]\alpha \rightarrow X([\tau']\beta)) \quad \wedge \quad [\tau]\alpha \rightarrow [\tau']\beta.$$

The intuition behind the representation is that an explicit change in value(s) of features in the trigger of a dependency constraint permit the features in the consequent β to change value (the 1st conjunct). What those values are is dependent on the material implication implicit in the constraint (the 2nd conjunct) and perhaps other constraints in the narrative. This expansion reflects a number of intuitions one generally has about causal dependencies such as that the material implication is entailed by the *causal rule.*

Various restrictions are sometimes placed on α, β, and τ and τ' which reflect the temporal structure being used or allowable sequences of quantifiers. For example, when using dependency constraints to represent causal dependencies, the additional constraint that $\tau \leq \tau'$ might be enforced.

In later versions of TAL, we have often used combinations of the C_T, C_F, C, R, I, and X operators instead of $\gg$ which provide for larger variation and subtlety in representing *trigger* conditions and effects for dependency constraints. For example,

$$[\tau]\alpha \gg [\tau']\beta$$

may also be represented as

$$(C_T([\tau]\alpha) \rightarrow X([\tau']\beta)) \quad \wedge \quad [\tau]\alpha \rightarrow [\tau']\beta,$$

or when one wants to weaken the trigger condition and the logical impact of a dependency, the following might be used instead,

$$[\tau]\alpha \wedge C_T([\tau]\beta) \rightarrow R([\tau']\gamma).$$

As another example, in some cases we may want to maintain an exception (e.g. false) to the default value (e.g. true) of a durational feature (e.g. $\mathsf{poss\_board}$). This could be represented as

$$[\tau]\alpha \rightarrow I([\tau']\mathsf{poss\_board} \hat{=} \mathsf{false}).$$

Note that unlike the previous example, the trigger condition for this dependency constraint does not require that α change value from false to true from $\tau - 1$ to τ as was the case with the use of $\gg$.

As a general rule, some instance of the C operator is used in the antecedent of a dependency constraint with the use of the I or R operator in the consequent. This reflects the intuition that *explicit change* causes *potential change.* The general exception to this case is when the consequent of a dependency constraint contains one or several durational features. In this case, one may do without the explicit trigger, but generally use the I macro in the consequent. The important point to note is that the axiom writer has a large degree of freedom as regards specification of trigger conditions and effects.

The following dependency constraints are used in the current narrative:

//A person who has a gun cannot board any airplane.
dep1 $\forall t, person\ [[t]\ \mathsf{inpocket}(person, \mathsf{gun}) \rightarrow$
$I([t]\ \forall airplane[\neg \mathsf{poss\_board}(person, airplane)])]$

//A person who is drunk may not be able to board an airplane.
dep2 $\forall t, person\ [[t]\ \mathsf{drunk}(person) \rightarrow$
$X([t]\ \forall airplane[\neg \mathsf{poss\_board}(person, airplane)])]$

//When an airplane moves, persons on board the airplane move.

dep3 $\forall t, airplane, person, loc_3$ [
$[t]$ $\mathsf{onplane}(airplane, person) \wedge C_T([t]\ \mathsf{loc}(airplane) \hat{=} loc_3) \rightarrow$
$R([t]\ \mathsf{loc}(person) \hat{=} value(t, \mathsf{loc}(airplane)))$]

//When persons move, things in their pockets also move.

dep4 $\forall t, person, pthing, loc_3$ [
$[t]$ $\mathsf{inpocket}(person, pthing) \wedge C_T([t]\ \mathsf{loc}(person) \hat{=} loc_3) \rightarrow$
$R([t]\ \mathsf{loc}(pthing) \hat{=} value(t, \mathsf{loc}(person)))$]

In the current narrative, the dependency constraints dep1 and dep2 describe qualifications to instances of action type acs4. Note that via the use of the operators, I and X, subtle variations in the specification of action qualification are made possible. In the case of dep1 for instance, the boarding action would be definitely qualified and have no effects if the person boarding had a gun in his pocket at the starting point of the action occurrence. On the other hand, in the case of dep2, the degree of qualification is much weaker, a drunk person may or may not be able to board. In fact, a later explicit observation that the person was on board after the occurrence of a boarding action in which the person was drunk would permit one to infer that the boarding action was successful after all.

The dependency constraints dep3 and dep4 basically describe the ramifications that ensue when certain objects are in a person's pocket who happens to be in a plane. Provided a person is on a plane, when it changes location, all objects in the persons pocket will also change location. Note the composite trigger for each of the dependencies. Due to the use of the C_T macro-operator, the location of the plane and the person has to explicitly change value from $t-1$ to t, whereas the features onplane and inpocket provide the context in which the dependencies *trigger* due to change in feature value.

7.5 The Narrative Specification

The narrative specification consists of observation statements, labeled using the obs label, and action occurrence statements, labeled using the occ label.

Observation statements are generally intended to describe the values of features at particular timepoints and generally consist of boolean combinations of atoms using the "$\hat{=}$" macro. Quantifiers may be used, but none of the operators I, R, or X are allowed. Although not present in this narrative, observation statements can be used to describe additional temporal constraints of a global nature such as those used to constrain any temporal terms describing the intervals in which actions occur. In this narrative, the temporal terms used are numeric and are implicitly constrained to be ordered by the background axiomatization of the linear discrete temporal structure assumed.

Note also that observations may occur at any point in the narrative and not only in the initial timepoint as in the example.

Action occurrence statements describe the occurrence and duration of actions in a narrative. Action occurrence statements, together with additional temporal constraints in observation statements allow for the description of partially-ordered sets of action occurrences with incompletely specified durations.[9]

obs1 [0] $\mathsf{loc(vladimir)} \hat{=} \mathsf{home1} \wedge \mathsf{loc(gun)} \hat{=} \mathsf{office} \wedge$
$\mathsf{loc(comb1)} \hat{=} \mathsf{home1} \wedge \neg\mathsf{drunk(vladimir)}$
obs2 [0] $\mathsf{loc(erik)} \hat{=} \mathsf{home2} \wedge \mathsf{loc(comb2)} \hat{=} \mathsf{home2} \wedge \neg\mathsf{drunk(erik)}$
obs3 [0] $\mathsf{loc(dimiter)} \hat{=} \mathsf{home3} \wedge \mathsf{loc(comb3)} \hat{=} \mathsf{home3} \wedge \mathsf{drunk(dimiter)}$
obs4 [0] $\mathsf{loc(sas609)} \hat{=} \mathsf{run609}$
occ1 [1, 2] put(vladimir, comb1, pocket1)
occ2 [1, 2] put(erik, comb2, pocket2)
occ3 [2, 4] travel(dimiter, home3, office)
occ4 [3, 5] travel(vladimir, home1, office)
occ5 [4, 6] travel(erik, home2, office)
occ6 [6, 7] put(vladimir, gun, pocket1)
occ7 [5, 7] travel(dimiter, office, airport)
occ8 [7, 9] travel(erik, office, airport)
occ9 [8, 10] travel(vladimir, office, airport)
occ10 [9, 10] board(dimiter, sas609)
occ11 [10, 11] board(vladimir, sas609)
occ12 [11, 12] board(erik, sas609)
occ13 [13, 16] fly(sas609, run609, run609b)

7.5.1 Translating Action Occurrence Statements

In some uses of TAL, we introduce actions as 1st-class citizens in the object language, providing a sort action and a predicate *Occurs* with type time × time × action. In other uses, we can do without this additional machinery. Due to this choice, there are two ways to translate an action occurrence statement. In both cases, a set Γ_{scd} in $\mathcal{L}(\mathrm{FL})$ is generated, although the formulas are different. In the former case, a set Γ_{occ} in $\mathcal{L}(\mathrm{FL})$ is also generated. To reflect this there is an intermediate preprocessing phase during translation which generates a set of labeled *schedule statements* in $\mathcal{L}(\mathrm{ND})$.

<u>Case 1: Actions are not 1st-Class Citizens</u>

Before translating narrative statements from $\mathcal{L}(\mathrm{ND})$ to $\mathcal{L}(\mathrm{FL})$, an intermediate transformation is applied, where for each action occurrence statement and its corresponding action type, a *schedule* statement is generated. For instance, given the action occurrence statement

occ13 [13, 16] fly(sas609, run609, run609b),

and the corresponding action type specification acs1

acs1 $[t_1, t_2]\ \mathsf{fly}(airplane, runway_1, runway_2) \leadsto$
$[t_1]\ \mathsf{loc}(airplane) \hat{=} runway_1 \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(airplane) \hat{=} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(airplane) \hat{=} runway_2)$,

the following schedule statement is generated:

scd13 $[13]\ \mathsf{loc(sas609)} \hat{=} \mathsf{run609} \rightarrow$
$I((13, 16)\ \mathsf{loc(sas609)} \hat{=} \mathsf{air}) \wedge R([16]\ \mathsf{loc(sas609)} \hat{=} \mathsf{run609b})$,

[9]Currently, action occurrence statements are simply action occurrence expressions and we leave open the possibility of conditionalizing action occurrences for future work.

and replaces the occurrence statement occ13 in the narrative description. The translation of scd13 is then added to Γ_{scd}.

Case 2: Actions are 1st-Class Citizens

In this case, the occurrence statement occ13 will be translated into an *Occurs* formula using the translation rule in Section 5 and placed in Γ_{occ}. In addition, the occurrence statement is used to generate a slightly modified schedule statement:

scd13 $\forall t_1, t_2, airplane, runway_1, runway_2.$
$[t_1, t_2]\ \mathsf{fly}(airplane, runway_1, runway_2) \rightarrow$
$[t_1]\ \mathsf{loc}(airplane) \hat{=} runway_1 \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(airplane) \hat{=} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(airplane) \hat{=} runway_2)$

The translation of this version of scd13 is then added to Γ_{scd}.

7.6 Restrictions on Narrative Syntax

This section is intended to provide a set of guidelines for expressing narrative statements, some voluntary, others obligatory. Technical details can be found in the following references [5, 21, 14].

7.6.1 Observation Statements

Observation statements can not contain any operator that translates into a wff containing the predicate *Occlude*. This would rule out the use of the operators I, R, or X. Observation statements are intended to be used for two purposes.

- Expressing temporal constraints not easily included in the definitions of action types. For instance, one might want to restrict a narrative's action occurrences to be totally ordered. The total ordering would be expressed as observations.
- Expressing direct observations about the possible or actual values of fluents at specific timepoints.

7.6.2 Action Occurrence Statements

Currently, action occurrence statements are restricted to be action occurrence expressions (see Section 3.4). In the future, we can foresee the need to model conditionalized actions. In this case, the syntax can be extended in a straightforward manner.

7.6.3 Action Type Statements

To simplify the discussion, assume that action type statements have the form $\alpha \rightarrow \beta$.[10]

[10] More complicated forms are permitted for both action type statements and dependency constraint statements and the restrictions would be modified accordingly. Details may be found in our other papers.

In order to ensure that predicate completion techniques can be applied to the occlusion predicate in the theory in $\mathcal{L}$(FL) that results from translating the narrative description in $\mathcal{L}$(ND) the following obligatory restrictions are placed on action type statements:

- The operators I, R, or X can not be used in the antecedent α of an action type statement.
- The use of disjunction (either explicit or implicit) in the consequent β of the action type statement can only be used in the scope of the I, R, or X operator.
- The use of an existentially quantified variable, quantified outside the scope of an I, R, or X operator, when used in the scope of the operator, may only be used in a feature term in the r.h.s. of the value operator (i.e. $\hat{=}$). Existentially quantified variables both quantified and used outside of the scope of the I, R, or X operator have no restrictions.
- Universally quantified variables have no restrictions.

Generally, the effects of actions should always be in the scope of either an I, R or X operator with no implicit or explicit application of negation to any of these operators. It is sometimes useful to express part of the effect outside of the scope. Assuming that the duration of an action is $[t, t']$, the preconditions to an action are restricted to holding at timepoint t.

7.6.4 Domain Constraint Statements

Domain constraint statements are intended to express logical constraints between fluents true in every state. Domain constraint statements can not contain any operator that translates into a wff containing the predicate *Occlude*. This would rule out the use of the operators I, R, or X. Other than that, there really aren't any restrictions.

7.6.5 Dependency Constraint Statements

To simplify the discussion, assume that dependency constraint statements have the form $\alpha \rightarrow \beta$.

In order to ensure that predicate completion techniques can be applied to the occlusion predicate in the theory in $\mathcal{L}$(FL) that results from translating the narrative description in $\mathcal{L}$(ND) the following obligatory restrictions are placed on dependency constraint statements:

- The operators I, R, or X can not be used in the antecedent α of a dependency constraint.
- The use of disjunction (either implicit or explicit) in the consequent β of the dependency constraint can only be used in the scope of the I, R, or X operator.
- The use of an existentially quantified variable, quantified outside the scope of an I, R, or X operator, when used in the scope of the operator, may only be used in a feature term in the r.h.s. of the value

operator (i.e. $\hat{=}$). Existentially quantified variables both quantified and used outside of the scope of the I, R, or X operator have no restrictions.

- Universally quantified variables have no restrictions.

Generally, the effects of dependency constraints should always be in the scope of either an I, R or X operator with no implicit or explicit application of negation to any of these operators. It is sometimes useful to express part of the effect outside of the scope. A large degree of freedom is offered in regard to triggering conditions for dependency constraints and we leave the degree of freedom up to the discretion of the axiom writer.

7.6.6 Temporal Constraint Statements

Temporal constraints can be found in a number of different statements in a narrative. The only restrictions we place on the use of temporal constraints is that the temporal terms and operators used directly reflect usage in the temporal base structure in $\mathcal{L}(\mathrm{FL})$ to which they are translated.

8 Previous Work

In this section, we provide an annotated list of references pertaining to previous work done in the context of TAL. Many of the technical details, proofs, specific logics and proposals for solving the frame, ramification, qualification, and action modeling problems are contained in this body of work.

- Sandewall – [1988-1998] The inspiration and origins of work with TAL begin with the body of work initiated by Sandewall and collected in the Features and Fluents (F&F) monograph [28]. For details concerning this work and later developments, we refer the reader to a concurrent article by Sandewall in this special series of papers in ETAI [29] containing a more exhaustive list of references pertaining to F&F and ASM.
- Doherty [4] – [1994] Doherty focuses on one of the 8 definitions of preferential entailment considered in F&F called PMON. In this paper, he provides a syntactic characterization of PMON in a many-sorted logic using pointwise circumscription. He shows that a particular class of narratives expressed using circumscribed theories can be reduced to logically equivalent 1st-order theories.
- Doherty [3] – [1994] An extended technical report subsuming the work in Doherty [4], provides additional characterizations of PMON in terms of standard predicate circumscription, predicate completion, and an algorithm used to avoid circumscription altogether which generates a definition of the *Occlude* predicate via syntactic transformations on a narrative description. The report also contains an appendix of all the benchmark narratives considered in Sandewall [28].
- Doherty and Łukaszewicz [12] – [1994] In this paper, we provide syntactic characterizations of 7 out of the 9 definitions of preferential entailment considered in F&F, using circumscription.

- Doherty and Peppas [11] – [1995] In this paper, we incorporate the use of primary and secondary fluents to model a particular class of indirect effects. What is perhaps more interesting, is the attempt to set up a framework for comparing logics such as PMON which use linear discrete time structures with the situation calculus which uses branching time.
- Karlsson [16] – [1995] In Karlsson's Licentiate Thesis, he considers how to formally characterize different modal truth criteria used in planning algorithms such as TWEAK and NONLIN. The formalization is done using PMON as a basis. Additional publications by Karlsson [17, 19] have extended this work.
- Doherty [5] – [1996] – (PMON+) This technical report contains a detailed and somewhat pedantic account of both surface and base languages for the first version of TAL. The purpose of the report was to use it as a basis for implementation of specific TAL logics. The report also contains an appendix of many benchmark narratives specified in TAL and continually discussed in the literature.
- Doherty, Łukaszewicz and Szałas [9, 10] – [1995,1997] In these papers, we develop a quantifier elimination algorithm which generates logically equivalent 1st-order formulas for a certain class of 2nd-order formulas. The intent with the work was to study the possibility of reducing other logics for action and change characterized in terms of circumscriptive theories, thus making them amenable to classical theorem proving techniques.
- Gustafsson and Doherty [15] – [1996] In this paper, we extend the original solution to the frame problem used in TAL with the use of causal constraints which takes us some of the way toward dealing with the specification of indirect effects of actions. Causal constraints are now subsumed by later work using dependency constraints. This paper also considers an example involving delayed effects of actions.
- Doherty, Łukaszewicz and Szałas [13] – [1996] In this paper, we consider the relation between the automatic generation of a definition for the *Occlude* predicate via circumscription reduction with the manual generation of Explanation Closure axioms considered in Schubert [30].
- Doherty and Gustafsson [6] – [1997] This report contains a rejected submission to AAAI97 which considers the problem of delayed effects of actions, a problem we believe is much more difficult than generally assumed. It is a predecessor to Karlsson et al. [23] in which the relation between delays and concurrency mentioned in [15] is developed in more detail.
- Karlsson [19] – [1997] In this paper, Karlsson investigates a number of weaknesses in situation calculus and provides an alternative semantics grounded in intuitions derived from work with TAL.
- Bjäreland and Karlsson [2] – [1997] This paper investigates the use of regression operators as a means of doing inference in TAL related formalisms. A detailed presentation of this approach and other approaches using tractable temporal logics can be found in Bjäreland [1].
- Karlsson, Gustafsson and Doherty [23] – [1998] This paper examines the use of delayed effects of actions and various problems of interference which arise.

- Doherty and Kvarnström [7] – [1997] In this paper, we consider a straightforward solution to simple forms of qualification. The key to the solution is the use of what we call *durational fluents*, first proposed in Karlsson and Gustafsson [21].
- Karlsson and Gustafsson [22] – [1998] In this submitted journal article, Karlsson and Gustafsson tackle the problem of modeling concurrent actions and the variety of interactions that may ensue between actions executing concurrently. An earlier version of the article is presented in [21].
- Gustafsson [14] – [1998] In his Licentiate Thesis, Gustafsson provides a detailed study of extensions to TAL involving dependency constraints, concurrency, and delayed effects of actions. The thesis provides a more detailed investigation of some of the work published in previous articles.
- Karlsson [20] – [1998] In this paper, Karlsson studies the possibility of introducing narratives as 1st-class objects in the object language of a logic whose semantics is related to that of TAL. This provides interesting possibilities in the area of planning where one can reason about the construction of narratives (i.e. plans) in the object language.
- Doherty, Łukaszewicz and E. Madalińska-Bugai [8] – [1998] In this paper, we study the relation between TAL and belief update. We show that dependency constraints can be used to advantage when dealing with integrity constraints in the context of belief update.
- Kvarnström and Doherty – [1997-98] We have been working on the design and implementation of an on-line research tool which implements a theorem prover for a restricted fragment of the logic in TAL considered in this article. The project is called VITAL which stands for *Visualization and Implementation of Temporal Action Logics*. The tool is implemented in Java, is delivered as a Marimba Castanet channel, and can be accessed via the WWW. See `http://anton.ida.liu.se/vital/` for details.

Many of our papers and software implementations can be found at the following sites:

- `http://www.ida.liu.se/labs/kplab/`
- `http://www.ida.liu.se/labs/kplab/projects/dls/`
- `http://anton.ida.liu.se/vital/`
- `http://www.ida.liu.se/publications/`
- `http://www.ida.liu.se/ext/pur/enter/`

An extended version of this paper and a new version of the surface and base languages for TAL can soon be found at the following site:

- `http://www.ida.liu.se/ext/cgi-bin/epa/cis/`

Appendix 1: Narrative in $\mathcal{L}$(ND)

DOMAIN SPECIFICATION

```
domain thing = {gun, comb1, comb2, comb3,
                vladimir, dimiter, erik, sas609}
domain location = {home1, home2, home3, office,
                   airport, run609, run609b, air}
domain runway = location [run609, run609b]
domain airplane = thing [sas609]
domain person = thing [vladimir - erik]
domain pthing = thing [gun, comb1 - comb3]
domain pocket = {pocket1, pocket2, pocket3}
feature loc(thing): location showname
feature inpocket(person,pthing): boolean
feature poss_board(person,airplane): boolean
feature drunk(person): boolean
feature onplane(airplane,person): boolean
action put(person, pthing, pocket)
action travel(person, location, location)
action fly(airplane, runway, runway)
action board(person, airplane)
```

PERSISTENCE STATEMENTS

per1 $\forall t, thing\, [\mathtt{true} \rightarrow Per(t, \mathsf{loc}(thing))]$

per2 $\forall t, person, pthing\, [\mathtt{true} \rightarrow Per(t, \mathsf{inpocket}(person, pthing))]$

per3 $\forall t, person, airplane\, [$
$\mathtt{true} \rightarrow Dur(t, \mathsf{poss\_board}(person, airplane), \mathsf{true})]$

per4 $\forall t, person\, [\mathtt{true} \rightarrow Per(t, \mathsf{drunk}(person))]$

per5 $\forall t, airplane, person\, [\mathtt{true} \rightarrow Per(t, \mathsf{onplane}(airplane, person))]$

THE NARRATIVE: OBSERVATIONS, ACTION OCCURRENCES AND TIMING

obs1 $[0]\ \mathsf{loc}(\mathsf{vladimir}) \hat{=} \mathsf{home1} \wedge \mathsf{loc}(\mathsf{gun}) \hat{=} \mathsf{office} \wedge$
$\mathsf{loc}(\mathsf{comb1}) \hat{=} \mathsf{home1} \wedge \neg \mathsf{drunk}(\mathsf{vladimir})$

obs2 $[0]\ \mathsf{loc}(\mathsf{erik}) \hat{=} \mathsf{home2} \wedge \mathsf{loc}(\mathsf{comb2}) \hat{=} \mathsf{home2} \wedge \neg \mathsf{drunk}(\mathsf{erik})$

obs3 $[0]\ \mathsf{loc}(\mathsf{dimiter}) \hat{=} \mathsf{home3} \wedge \mathsf{loc}(\mathsf{comb3}) \hat{=} \mathsf{home3} \wedge \mathsf{drunk}(\mathsf{dimiter})$

obs4 $[0]\ \mathsf{loc}(\mathsf{sas609}) \hat{=} \mathsf{run609}$

occ1 $[1, 2]\ \mathsf{put}(\mathsf{vladimir}, \mathsf{comb1}, \mathsf{pocket1})$

occ2 $[1, 2]\ \mathsf{put}(\mathsf{erik}, \mathsf{comb2}, \mathsf{pocket2})$

occ3 $[2, 4]\ \mathsf{travel}(\mathsf{dimiter}, \mathsf{home3}, \mathsf{office})$

occ4 $[3, 5]\ \mathsf{travel}(\mathsf{vladimir}, \mathsf{home1}, \mathsf{office})$

occ5 $[4, 6]\ \mathsf{travel}(\mathsf{erik}, \mathsf{home2}, \mathsf{office})$

occ6 $[6, 7]\ \mathsf{put}(\mathsf{vladimir}, \mathsf{gun}, \mathsf{pocket1})$

occ7 $[5, 7]\ \mathsf{travel}(\mathsf{dimiter}, \mathsf{office}, \mathsf{airport})$

occ8 $[7, 9]\ \mathsf{travel}(\mathsf{erik}, \mathsf{office}, \mathsf{airport})$

occ9 $[8, 10]\ \mathsf{travel}(\mathsf{vladimir}, \mathsf{office}, \mathsf{airport})$

occ10 $[9, 10]\ \mathsf{board}(\mathsf{dimiter}, \mathsf{sas609})$

occ11 $[10, 11]\ \mathsf{board}(\mathsf{vladimir}, \mathsf{sas609})$

occ12 $[11, 12]\ \mathsf{board}(\mathsf{erik}, \mathsf{sas609})$

occ13 $[13, 16]\ \mathsf{fly}(\mathsf{sas609}, \mathsf{run609}, \mathsf{run609b})$

ACTION TYPES

acs1 $[t_1, t_2]\ \mathsf{fly}(airplane, runway_1, runway_2) \rightsquigarrow$
$[t_1]\ \mathsf{loc}(airplane) \hat{=} runway_1 \rightarrow$

$I((t_1, t_2)\ \mathsf{loc}(airplane) \hat{=} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(airplane) \hat{=} runway_2)$

acs2 $[t_1, t_2]\ \mathsf{put}(person, pthing, pocket) \leadsto$
$[t_1]\ \mathsf{loc}(person) \hat{=} value(t_1, \mathsf{loc}(pthing)) \rightarrow$
$R((t_1, t_2]\ \mathsf{inpocket}(person, pthing))$

acs3 $[t_1, t_2]\ \mathsf{travel}(person, loc_1, loc_2) \leadsto [t_1]\ \mathsf{loc}(person) \hat{=} loc_1 \rightarrow$
$R([t_2]\ \mathsf{loc}(person) \hat{=} loc_2)$

acs4 $[t_1, t_2]\ \mathsf{board}(person, airplane) \leadsto$
$[t_1]\ \mathsf{poss\_board}(person, airplane) \wedge \mathsf{loc}(person) \hat{=} \mathsf{airport} \rightarrow$
$R([t_2]\ \mathsf{loc}(person) \hat{=} value(t_2, \mathsf{loc}(airplane)) \wedge$
$\mathsf{onplane}(airplane, person))$

DOMAIN CONSTRAINTS

//A pthing cannot be in two pockets at the same time.
dom1 $\forall t, pthing_1, person_1, person_2\ [\neg(person_1 = person_2) \wedge$
$[t]\ \mathsf{inpocket}(person_1, pthing_1) \rightarrow [t]\ \neg\mathsf{inpocket}(person_2, pthing_1)]$

//A person cannot be on board two airplanes at the same time.
dom2 $\forall t, person_1, airplane1, airplane2\ [\neg(airplane1 = airplane2) \wedge$
$[t]\ \mathsf{onplane}(airplane1, person_1) \rightarrow [t]\ \neg\mathsf{onplane}(airplane2, person_1)]$

DEPENDENCY CONSTRAINTS

//A person who has a gun cannot board any airplane.
dep1 $\forall t, person\ [[t]\ \mathsf{inpocket}(person, \mathsf{gun}) \rightarrow$
$I([t]\ \forall airplane[\neg\mathsf{poss\_board}(person, airplane)])]$

//A person who is drunk may not be able to board an airplane.
dep2 $\forall t, person\ [[t]\ \mathsf{drunk}(person) \rightarrow$
$X([t]\ \forall airplane[\neg\mathsf{poss\_board}(person, airplane)])]$

//When an airplane moves, persons on board the airplane move.
dep3 $\forall t, airplane, person, loc_3\ [$
$[t]\ \mathsf{onplane}(airplane, person) \wedge C_T([t]\ \mathsf{loc}(airplane) \hat{=} loc_3) \rightarrow$
$R([t]\ \mathsf{loc}(person) \hat{=} value(t, \mathsf{loc}(airplane)))]$

//When persons move, things in their pockets also move.
dep4 $\forall t, person, pthing, loc_3\ [$
$[t]\ \mathsf{inpocket}(person, pthing) \wedge C_T([t]\ \mathsf{loc}(person) \hat{=} loc_3) \rightarrow$
$R([t]\ \mathsf{loc}(pthing) \hat{=} value(t, \mathsf{loc}(person)))]$

INTERMEDIATE SCHEDULE STATEMENTS

The following intermediate schedule statements are generated from the action type specifications.

scd1 $\forall t_1, t_2, airplane, runway_1, runway_2.$
$[t_1, t_2]\ \mathsf{fly}(airplane, runway_1, runway_2) \rightarrow$
$[t_1]\ \mathsf{loc}(airplane) \hat{=} runway_1 \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(airplane) \hat{=} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(airplane) \hat{=} runway_2)$

scd2 $\forall t_1, t_2, person, pthing, pocket.\ [t_1, t_2]\ \mathsf{put}(person, pthing, pocket) \rightarrow$
$[t_1]\ \mathsf{loc}(person) \hat{=} value(t_1, \mathsf{loc}(pthing)) \rightarrow$
$R((t_1, t_2]\ \mathsf{inpocket}(person, pthing))$

scd3 $\forall t_1, t_2, person, loc_1, loc_2.\ [t_1, t_2]\ \mathsf{travel}(person, loc_1, loc_2) \rightarrow$
$[t_1]\ \mathsf{loc}(person) \hat{=} loc_1 \rightarrow R([t_2]\ \mathsf{loc}(person) \hat{=} loc_2)$

scd4 $\forall t_1, t_2, person, airplane.\ [t_1, t_2]\ \mathsf{board}(person, airplane) \rightarrow$
$[t_1]\ \mathsf{poss\_board}(person, airplane) \wedge \mathsf{loc}(person) \hat{=} \mathsf{airport} \rightarrow$
$R([t_2]\ \mathsf{loc}(person) \hat{=} value(t_2, \mathsf{loc}(airplane)) \wedge$
$\mathsf{onplane}(airplane, person))$

Appendix 2: Narrative in $\mathcal{L}$(FL)

PERSISTENCE STATEMENTS

per1 $\forall t, thing, v\ [Holds(t-1, \mathsf{loc}(thing), v) \oplus$
$Holds(t, \mathsf{loc}(thing), v) \rightarrow Occlude(t, \mathsf{loc}(thing))]$

per2 $\forall t, person, thing, v\ [Holds(t-1, \mathsf{inpocket}(person, thing), v) \oplus$
$Holds(t, \mathsf{inpocket}(person, thing), v) \rightarrow$
$Occlude(t, \mathsf{inpocket}(person, thing))]$

per3 $\forall t, person, airplane\ [\neg Holds(t, \mathsf{poss\_board}(person, airplane), \mathsf{true}) \rightarrow$
$Occlude(t, \mathsf{poss\_board}(person, airplane))]$

per4 $\forall t, person, v\ [Holds(t-1, \mathsf{drunk}(person), v) \oplus$
$Holds(t, \mathsf{drunk}(person), v) \rightarrow Occlude(t, \mathsf{drunk}(person))]$

per5 $\forall t, airplane, person, v\ [Holds(t-1, \mathsf{onplane}(airplane, person), v) \oplus$
$Holds(t, \mathsf{onplane}(airplane, person), v) \rightarrow$
$Occlude(t, \mathsf{onplane}(airplane, person))]$

THE NARRATIVE: OBSERVATIONS, ACTION OCCURRENCES AND TIMING

obs1 $Holds(0, \mathsf{loc}(\mathsf{vladimir}), \mathsf{home1}) \wedge Holds(0, \mathsf{loc}(\mathsf{gun}), \mathsf{office}) \wedge$
$Holds(0, \mathsf{loc}(\mathsf{comb1}), \mathsf{home1}) \wedge \neg Holds(0, \mathsf{drunk}(\mathsf{vladimir}), \mathsf{true})$

obs2 $Holds(0, \mathsf{loc}(\mathsf{erik}), \mathsf{home2}) \wedge Holds(0, \mathsf{loc}(\mathsf{comb2}), \mathsf{home2}) \wedge$
$\neg Holds(0, \mathsf{drunk}(\mathsf{erik}), \mathsf{true})$

obs3 $Holds(0, \mathsf{loc}(\mathsf{dimiter}), \mathsf{home3}) \wedge Holds(0, \mathsf{loc}(\mathsf{comb3}), \mathsf{home3}) \wedge$
$Holds(0, \mathsf{drunk}(\mathsf{dimiter}), \mathsf{true})$

obs4 $Holds(0, \mathsf{loc}(\mathsf{sas609}), \mathsf{run609})$

occ1 $Occurs(1, 2, \mathsf{put}(\mathsf{vladimir}, \mathsf{comb1}, \mathsf{pocket1}))$

occ2 $Occurs(1, 2, \mathsf{put}(\mathsf{erik}, \mathsf{comb2}, \mathsf{pocket2}))$

occ3 $Occurs(2, 4, \mathsf{travel}(\mathsf{dimiter}, \mathsf{home3}, \mathsf{office}))$

occ4 $Occurs(3, 5, \mathsf{travel}(\mathsf{vladimir}, \mathsf{home1}, \mathsf{office}))$

occ5 $Occurs(4, 6, \mathsf{travel}(\mathsf{erik}, \mathsf{home2}, \mathsf{office}))$

occ6 $Occurs(6, 7, \mathsf{put}(\mathsf{vladimir}, \mathsf{gun}, \mathsf{pocket1}))$

occ7 $Occurs(5, 7, \mathsf{travel}(\mathsf{dimiter}, \mathsf{office}, \mathsf{airport}))$

occ8 $Occurs(7, 9, \mathsf{travel}(\mathsf{erik}, \mathsf{office}, \mathsf{airport}))$

occ9 $Occurs(8, 10, \mathsf{travel}(\mathsf{vladimir}, \mathsf{office}, \mathsf{airport}))$

occ10 $Occurs(9, 10, \mathsf{board}(\mathsf{dimiter}, \mathsf{sas609}))$

occ11 $Occurs(10, 11, \mathsf{board}(\mathsf{vladimir}, \mathsf{sas609}))$

occ12 $Occurs(11, 12, \mathsf{board}(\mathsf{erik}, \mathsf{sas609}))$

occ13 $Occurs(13, 16, \mathsf{fly}(\mathsf{sas609}, \mathsf{run609}, \mathsf{run609b}))$

SCHEDULE STATEMENTS

scd1 $\forall t_1, t_2, airplane, runway_1, runway_2\ [$
$Occurs(t_1, t_2, \mathsf{fly}(airplane, runway_1, runway_2)) \rightarrow$
$(Holds(t_1, \mathsf{loc}(airplane), runway_1) \rightarrow \forall t\, [t_1 < t \wedge t < t_2 \rightarrow$
$Holds(t, \mathsf{loc}(airplane), \mathsf{air}) \wedge Occlude(t, \mathsf{loc}(airplane))] \wedge$
$Holds(t_2, \mathsf{loc}(airplane), runway_2) \wedge Occlude(t_2, \mathsf{loc}(airplane)))]$

scd2 $\forall t_1, t_2, person, pthing, pocket\ [$
$Occurs(t_1, t_2, \mathsf{put}(person, pthing, pocket)) \rightarrow$
$(Holds(t_1, \mathsf{loc}(person), val(t_1, \mathsf{loc}(pthing))) \rightarrow$
$Holds(t_2, \mathsf{inpocket}(person, pthing), \mathsf{true}) \wedge$
$Occlude(t_2, \mathsf{inpocket}(person, pthing)))]$

scd3 $\forall t_1, t_2, person, loc_1, loc_2\ [Occurs(t_1, t_2, \mathsf{travel}(person, loc_1, loc_2)) \rightarrow$
$(Holds(t_1, \mathsf{loc}(person), loc_1) \rightarrow$
$Holds(t_2, \mathsf{loc}(person), loc_2) \wedge Occlude(t_2, \mathsf{loc}(person)))]$

scd4 $\forall t_1, t_2, person, airplane\ [Occurs(t_1, t_2, \mathsf{board}(person, airplane)) \rightarrow$
$Holds(t_1, \mathsf{poss\_board}(person, airplane), \mathsf{true}) \wedge$
$Holds(t_1, \mathsf{loc}(person), airport) \rightarrow$
$Holds(t_2, \mathsf{loc}(person), val(t_2, loc(airplane))) \wedge$
$Holds(t_2, \mathsf{onplane}(airplane, person), \mathsf{true}) \wedge$
$Occlude(t_2, \mathsf{loc}(person)) \wedge Occlude(t_2, \mathsf{onplane}(airplane, person))]$

DOMAIN CONSTRAINTS

dom1 $\forall t, pthing_1, person_1, person_2\ [\neg(person_1 = person_2) \wedge$
$Holds(t, \mathsf{inpocket}(person_1, pthing_1), \mathsf{true}) \rightarrow$
$\neg Holds(t, \mathsf{inpocket}(person_2, pthing_1), \mathsf{true})]$

dom2 $\forall t, person_1, airplane_1, airplane_2\ [\neg(airplane_1 = airplane_2) \wedge$
$Holds(t, \mathsf{onplane}(airplane_1, person_1), \mathsf{true}) \rightarrow$
$\neg Holds(t, \mathsf{onplane}(airplane_2, person_1))]$

DEPENDENCY CONSTRAINTS

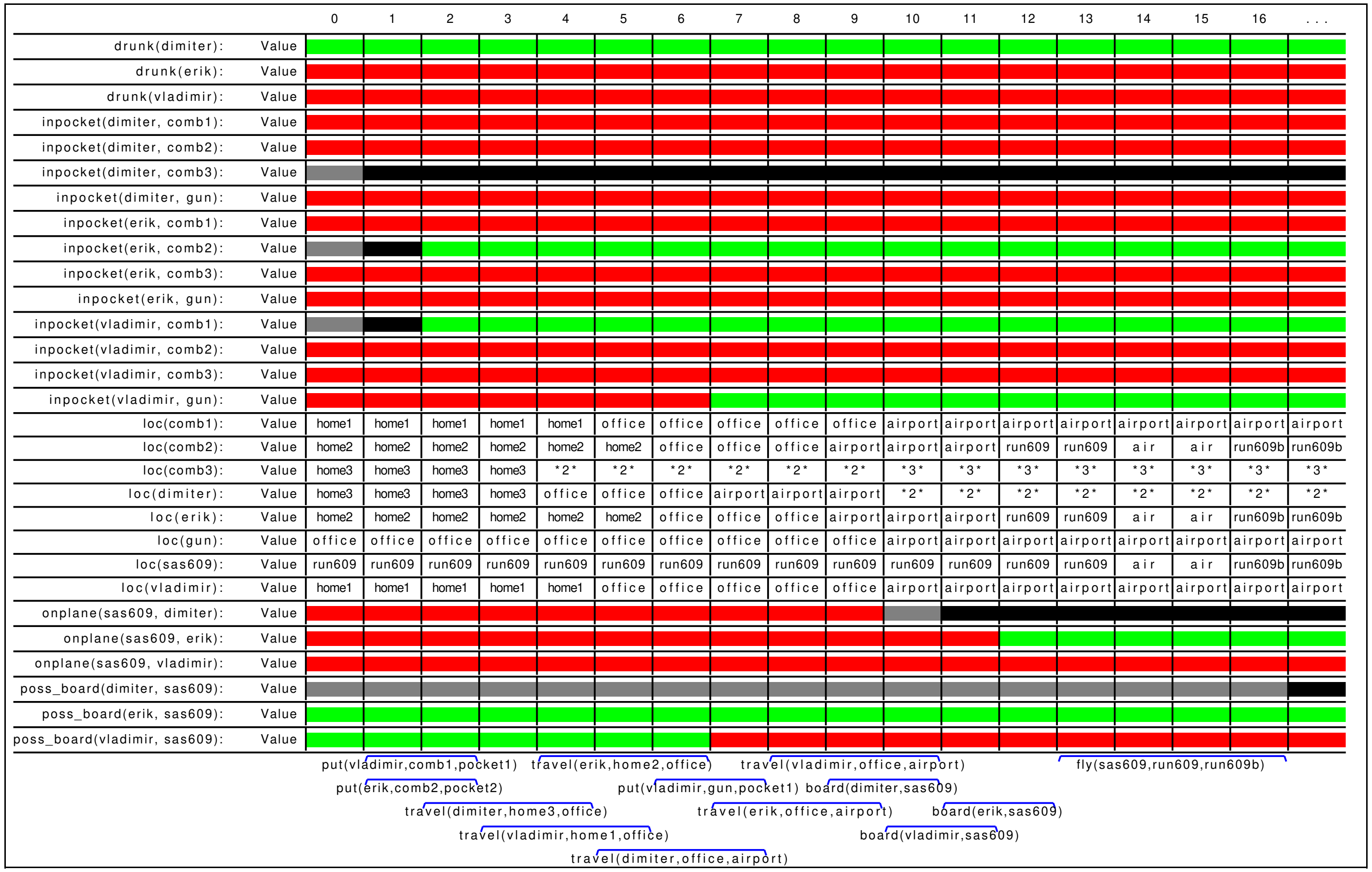
dep1 $\forall t, person\ [Holds(t, \mathsf{inpocket}(person, \mathsf{gun}), \mathsf{true}) \rightarrow$
$\forall airplane\ [\neg Holds(t, \mathsf{poss\_board}(person, airplane), \mathsf{true})] \wedge$
$\forall airplane\ [Occlude(t, \mathsf{poss\_board}(person, airplane))]]$

dep2 $\forall t, person\ [Holds(t, \mathsf{drunk}(person), \mathsf{true}) \rightarrow$
$\forall airplane\ [Occlude(t, \mathsf{poss\_board}(person, airplane))]]$

dep3 $\forall t, airplane, person, loc_3\ [Holds(t, \mathsf{onplane}(airplane, person), \mathsf{true}) \wedge$
$Holds(t, \mathsf{loc}(airplane), loc_3) \wedge$
$\forall u\ [t = u + 1 \rightarrow \neg Holds(u, \mathsf{loc}(airplane), loc_3)] \rightarrow$
$Holds(t, \mathsf{loc}(person), val(t, \mathsf{loc}(airplane))) \wedge Occlude(t, \mathsf{loc}(person))]$

dep4 $\forall t, person, pthing, loc_3\ [Holds(t, \mathsf{inpocket}(person, pthing), \mathsf{true}) \wedge$
$Holds(t, \mathsf{loc}(person), loc_3) \wedge$
$\forall u\ [t = u + 1 \rightarrow \neg Holds(u, \mathsf{loc}(person), loc_3)] \rightarrow$
$Holds(t, \mathsf{loc}(pthing), val(t, \mathsf{loc}(person))) \wedge Occlude(t, \mathsf{loc}(pthing))]$

Appendix 3



Red and *green* stand for false and true values for boolean fluents, *gray* stands for true or false, while *black* after *gray* stands for "do not know the value, but will take the value *gray* ends up being". "*2*" means 2 possible values. They are not shown in the diagram due to lack of space. The diagram is automatically generated using the VITAL tool. You should have a colored copy of Appendix 3.

References

[1] M. Bjäreland. Two aspects of automating logics of action and change: Regression and tractability. Master's thesis, Linköping University, 1998. Thesis No 674. LiU-Tek-Lic 1998:09.

[2] M. Bjäreland and L. Karlsson. Reasoning by regression: Pre- and postdiction procedures for logics of action and change with nondeterminism. In *Proceedings of the 15th International Joint Conference on Artficial Intelligence (IJCAI'97)*, Nagoya, Japan, August 1997. Morgan Kaufmann.

[3] P. Doherty. Notes on PMON circumscription. Technical Report LITH-IDA-94-43, Dept. 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 (ECAI'94), Aug. 8-12, Amsterdam*, pages 401–405, 1994.

[5] P. Doherty. PMON+: A fluent logic for action and change. formal specification, version 1.0. Technical Report R-96-33, Dept. of Computer and Information Science, Linköping University, 1996. Also published in *Linköping Electronic Articles in Computer and Information Science*, Vol.2(1997):nr020. `http://www.ep.liu.se/ea/cis/1997/020/`.

[6] P. Doherty and J. Gustafsson. Delayed effects of actions = direct effects + causal rules. In *Linköping Electronic Articles in Computer and Information Science*. Linköping University Electronic Press, 1998. Available at: `http://www.ep.liu.se/ea/cis/1998/001/`.

[7] P. Doherty and J. Kvarnström. Tackling the qualification problem using fluent dependency constraints: Preliminary report. In *5th International Workshop on Temporal Representation and Reasoning (TIME'98)*, 1998.

[8] P. Doherty, W. Łukaszewicz, and E. Madalińska-Bugaj. The PMA and relativizing change for action update. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, 1998.

[9] P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited: Preliminary report. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, volume 2, pages 1502–1508, 1995.

[10] P. Doherty, W. Łukaszewicz, and A. Szałas. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18:297–336, 1997.

[11] P. Doherty and P. Peppas. A comparison between two approaches to ramification: PMON(R) and $\mathcal{AR}_0$. In X. Yao, editor, *Proceedings of the 8th Australian Joint Conference on Artificial Intelligence*, pages 267–274. World Scientific, 1995.

[12] P. Doherty and W. Łukaszewicz. 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.

[13] P. Doherty, W. Łukaszewicz, and A. Szałas. Explaining explanation closure. In *Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems (ISMIS'96)*, 1996.

[14] J. Gustafsson. Extending temporal action logic for ramification and concurrency. Master's thesis, Linköping University, 1998. Thesis No 719. LiU-Tek-Lic 1998:54.

[15] J. Gustafsson and P. Doherty. Embracing occlusion in specifying the indirect effects of actions. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, 1996.

[16] L. Karlsson. Specification and synthesis of plans using the features and fluents framework. Master's thesis, Linköping University, 1995. Thesis No 469. LiU-Tek-Lic 1995:01.

[17] L. 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.

[18] L. Karlsson. Planning, truth criteria and the systematic approach to action and change. In *Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems (ISMIS'96)*, Lecture Notes for Artificial Intelligence. Springer Verlag, 1996.

[19] L. Karlsson. Reasoning with incomplete initial information and nondeterminism in situation calculus. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence, (IJCAI'97)*, 1997.

[20] L. Karlsson. Anything can happen: on narratives and hypothetical reasoning. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*. Morgan Kaufmann, 1998.

[21] L. Karlsson and J. Gustafsson. Reasoning about actions in a multi-agent environment. In *Linköping Electronic Articles in Computer and Information Science*, volume 2. Linköping University Electronic Press, 1997. Available at: `http://www.ep.liu.se/ea/cis/1997/014/`.

[22] L. Karlsson and J. Gustafsson. Reasoning about concurrent interaction. *Submitted to the Journal of Logic and Computation*, 1998. Submitted in August, 1997. Revised version submitted in May, 1998.

[23] L. Karlsson, J. Gustafsson, and P. Doherty. Delayed effects of actions. In *Proceedings of the 13th European Conference on Artificial Intelligence, (ECAI'98)*, pages 542–546, 1998.

[24] J. Kvarnström and P. Doherty. VITAL research tool, 1997. Available at: `http://anton.ida.liu.se/vital/vital.html`.

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

[26] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[27] E. Sandewall. Filter preferential entailment for the logic of action and change. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence, (IJCAI'89)*, 1989.

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

[29] E. Sandewall. Cognitive robotics and its metatheory. In *Linköping Electronic Articles in Computer and Information Science*, volume 3(1998): nr 016. Linköping University Electronic Press, 1998. Available at: `http://www.ep.liu.se/ea/cis/1998/016/`. Submitted for publication to the 3rd quarter ETAI.

[30] L. Schubert. Monotonic solution of the frame problem in situation calculus. In H. E. Kyburg, R. P. Loui, and G. N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, 1990.