Chapter 1

Temporal Action Logics

Patrick Doherty and Jonas Kvarnström

First Draft Version: February 28, 2006
Chapter 15
To be included in The Handbook of Knowledge Representation
Eds. V. Lifschitz, F. van Harmelen, F. Porter

1.1 Introduction

The study of frameworks and formalisms for reasoning about action and change [64, 56, 58, 62, 67, 3] has been central to the knowledge representation field almost from the inception of Artificial Intelligence as a general field of research [51, 55].

The phrase ”Temporal Action Logics” represents a class of logics for reasoning about action and change that evolved from Sandewall’s book on *Features and Fluents* [58] and owes much to this ambitious project. There are essentially three major parts to Sandewall’s work. He first developed a narrative-based logical framework for specifying agent behavior in terms of action scenarios. The logical framework is state-based and uses explicit time structures. He then developed a formal framework for assessing the correctness (soundness and completeness) of logics for reasoning about action and change relative to a set of well-defined intended conclusions, where reasoning problems were classified according to their ontological or epistemological characteristics. Finally, he proposed a number of logics defined semantically in terms of definitions of preferential entailment[1] and assessed their correctness using his assessment framework.

Several of these logics were intended to correspond directly to existing logics of action and change proposed by others at the time, while the rest were new and were intended to characterize broad classes of reasoning problems which subsumed some of the existing approaches. Each of these definitions of preferential entailment were then analyzed using the assessment framework, giving upper and lower bounds in terms of the classes of reasoning

[1] Preferential entailment reduces the set of classical models of a theory by only retaining those models that are minimal according to a given preference relation, a strict partial order over logical interpretations [63].

problems for which they produced exactly the intended conclusions. Much insight was gained both in terms of advantages and limitations of previously proposed logics of action and change and in how one might go about proposing new logics of action and change in a principled manner with formal assessments included.

The starting point for Temporal Action Logics was one of the definitions of preferential entailment in Sandewall's book called PMON (Pointwise Minimization of Occlusion with Nochange premises). It was one of the few preferential entailment methods that were assessed correct for the $\mathcal{K}$–**IA** class of action scenario descriptions, where $\mathcal{K}$ is an epistemological characteristic stating approximately that explicit, correct and accurate knowledge is provided (with no requirements on complete knowledge in the initial state and no restrictions on knowledge about other states), and **IA** is an ontological characteristic stating approximately that discrete integer time is used together with plain inertia (without surprises or other complicating factors).

PMON solved the frame problem relative to an explicit statement of assumptions under which it could be assessed correct. In addition, the nature of the definition of preferential entailment was somewhat related to explanation closure [61, 11], although a partitioning of action scenario theories was used where only parts of the theory were minimized and other parts used as filters on the preferred model set for the theory. Though ramifications and qualifications to actions were not allowed in $\mathcal{K}$–**IA**, the class is in fact quite broad, permitting the use of conditional effects, non-deterministic effects, incomplete specification of states and the timing of actions, actions with duration and specification of dynamics within action durations.

1.1.1 PMON and TAL

While the original PMON was characterized semantically in terms of a preferential entailment method, Doherty later developed an equivalent syntactic characterization in classical 2nd-order logic, using a circumscription axiom to formalize the PMON definition of preferential entailment [12, 19]. In these papers, he also showed that the 2nd-order circumscription axiom was equivalent to a 1st-order pointwise circumscription axiom, enabling the use of standard first-order theorem proving techniques to reason about PMON action narratives. In extended versions of PMON which led to TAL, it has also been shown that quantifier elimination techniques or predicate completion techniques can be used to reduce TAL circumscribed theories to logically equivalent 1st-order theories under certain assumptions.

This new logic was also called **PMON**, and used two languages for representing and reasoning about narratives. The surface language $\mathcal{L}(\mathrm{SD})$, Language for Scenario Descriptions, provided a convenient high-level notation for describing narratives, and could be described as a set of macros easily translated into a base language $\mathcal{L}(\mathrm{FL})$, which was initially a many-sorted first-order language and was later altered to be an order-sorted[2] first-order language. The $\mathcal{L}(\mathrm{SD})$ language was later renamed to $\mathcal{L}(\mathrm{ND})$, Language for Narrative Descriptions.

The original **PMON** logic was further extended and generalized in several steps in order to deal with such issues as the ramification and qualification problems, use of con-

[2]Essentially, an order-sorted language allows the use of sub-sorts; for example, CAR and BICYCLE may be sub-sorts of the VEHICLE sort.

current actions, use of structured object-oriented action theories, and use as a specification formalism for TALplanner. Each extension generally implied adding new macros to $\mathcal{L}(\mathrm{ND})$, adding additional predicates to $\mathcal{L}(\mathrm{FL})$, extending the translation definition to $\mathcal{L}(\mathrm{FL})$ and providing slight modifications to the circumscription policies used. It is important to observe that all extensions proposed have been made in a manner which preserves the property of reducibility of the 2nd-order circumscription theory to a 1st-order theory. This is essential for practical reasons.

A number of the main extensions to **PMON** which led to the TAL family of logics include:

- **PMON-RC** [25], provides a solution to the ramification problem for a broad, but as yet unassessed class of action scenarios. The main idea is the addition of a new statement type for causal constraints, where changes taking place in the world can automatically trigger new changes at the same time-point or at a specified delay from the original change. The solution is very fine-grained in the sense that one can easily encode dependencies between individual objects in the domain, work with both boolean and non-boolean fluents and represent both Markovian and non-Markovian dependencies [21]. PMON-RC also correctly handles chains of side effects.

- **TAL 1.0** (**PMON**$^+$) [13], **PMON**$^+$ is an extended version of the original **PMON** logic incorporating the changes made in **PMON-RC** together with other useful extensions. This logic was later renamed **TAL 1.0** and provided the first stable kernel for the TAL family of logics.

- **TAL-C** [34], uses fluent dependency constraints (an extended form of causal constraints) as a basis for representing concurrent actions. A number of phenomena related to action concurrency such as interference between one action's effects and another's execution, bounds on concurrency, and conflicting, synergistic, and cumulative effects of concurrent actions are supported.

- **TAL 2.0** [15] provides a basic stable kernel of TAL. It is essentially TAL-C with some useful extensions and includes a tutorial on TAL and how it is used.

- **TAL-Q** [16, 41], introduces the idea of combining an encoding of default values for features using persistence statements together with dependency constraints for representing qualifications to actions.

TAL 2.0 (TAL-C) extended with additions from **TAL-Q**, has been used as the basis for much of the recent work with Temporal Action Logics and will be described in some detail in this chapter. In the remainder of the chapter, we will use "TAL" as a term to denote the latest stable kernel of this family of logics.

1.1.2 Previous Work

There has been a great deal of previous work in the development of the material described in this chapter. We briefly summarize this work chronologically.

The root node from which TAL originated is the Features and Fluents (F&F) monograph [58]. Later developments with F&F are summarized in Sandewall [60]. Doherty [12] provides a syntactic characterization of PMON using pointwise circumscription and shows

how a particular class of narratives can be characterized as first-order theories. Doherty [6] contains a detailed account of PMON circumscription theories and provides additional characterizations of PMON in terms of predicate circumscription and predicate completion, where syntactic transformations are defined on narratives to provide a definition of the $Occlude$ predicate. Doherty and Łukaszewicz [19] provide syntactic characterizations of 7 out of the 9 definitions of preferential entailment considered in F&F, using different forms of circumscription. Doherty and Peppas [10] incorporate the use of primary and secondary fluents in PMON to model a subclass of indirect effects of actions. A framework is also introduced for comparing linear time logics such as PMON with branching time logics such as the situation calculus. Karlsson [28] considers how to formally characterize different modal truth criteria used in planning algorithms such as TWEAK and NONLIN using PMON. Karlsson [30, 29] extends this work. Doherty [13] provides a detailed description of TAL 1.0 used as a basis for early implementations of TAL. Doherty, Łukaszewicz and Szałas [8, 9] develop a quantifier elimination algorithm which constructively 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 circumscription theories, thus making them amenable to classical theorem proving techniques. Gustafsson and Doherty [25] extend TAL to deal with ramifications of actions by introducing causal constraints. Causal constraints have later been subsumed by the use of dependency constraints in TAL. In addition, we show how to represent delayed effects of actions in TAL. Doherty, Łukaszewicz and Szałas [11] consider the relation between the automatic generation of a definition for the *Occlude* predicate using circumscription and quantifier elimination, with the manual generation of Explanation Closure axioms considered in Schubert [61]. Karlsson [31] investigates a number of weaknesses in situation calculus and provides an alternative semantics grounded in intuitions derived from work with TAL. Bjäreland and Karlsson [5] investigates the use of regression operators as a means of doing inference in TAL related formalisms. Bjäreland [4] provides a detailed presentation of the approach in [5] and other approaches using tractable temporal logics. Karlsson, Gustafsson and Doherty [14, 35] examine the use of delayed effects of actions and various problems of interference which arise with their introduction. Doherty and Kvarnström [16] present an initial solution to simple forms of qualification to actions. Kvarnström and Doherty [41] provide a more detailed solution to the qualification problem described in this chapter. Karlsson and Gustafsson [34] consider the problem of modeling concurrent actions in TAL and the variety of interactions that may ensue between actions executing concurrently. Gustafsson [23] provides a detailed study of extensions to TAL involving dependency constraints, concurrency, and delayed effects of actions. Karlsson [32] 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. Doherty, Łukaszewicz and Madalińska-Bugai [7] study the relation between TAL and belief update. Karlsson [33] provides detailed accounts of narratives as 1st-class citizens in action logic, concurrent actions and additional extensions to TAL. Gustafsson [24] provides a detailed description of many of the extensions to TAL up to 2001. Gustafsson and Kvarnström [26, 27] provide a novel means of structuring large TAL narratives based on the use of intuitions from object-oriented programming.

Doherty and Kvarnström [17] present a new forward chaining planner which uses TAL as a semantic framework for its development. In Kvarnström and Doherty [42], an early detailed account of TALplanner is provided. Kvarnström, Doherty and Haslum [43] provide

an extension to TALplanner which integrates concurrent actions and resources. Doherty and Kvarnström [18] provide a concise overview of TALplanner. Kvarnström and Magnusson [44] provide a description of control rules used in TALplanner in the AIPS Third International Planning Competition, 2002. Kvarnström [39] discusses application of domain analysis techniques to control rules in TALplanner. Kvarnström [40] provides the most recent and most detailed description of TALplanner.

The thesis work of both Karlsson [33] and Gustafsson [24] provide excellent references to much of the later extensions to TAL. The thesis work of Kvarnström [40] provides an excellent description of TALplanner. A software system VITAL [38] for reasoning about action and change using TAL is available for download and on-line use.

1.1.3 Chapter Structure

In Section 1.2, the main concepts and ideas used in the development of TAL are presented. In Section 1.3, action narratives used in TAL are defined and a complex scenario, the Russian Airplane Hijack (RAH) Scenario, is presented. This will be used throughout the chapter to explain the different features provided by TAL. Section 1.4 considers the relation between a high level macro language $\mathcal{L}(\mathrm{ND})$ used to specify action narratives and the base logical language $\mathcal{L}(\mathrm{FL})$ in which it is translated to. In Section 1.5 we provide a formal description of the language $\mathcal{L}(\mathrm{ND})$ and in Section 1.6, we provide a formal definition of the base logical language $\mathcal{L}(\mathrm{FL})$. In Section 1.7, the circumscription policy used to define the definition of preferential entailment used in TAL is presented. In addition, we show how the resulting 2nd-order circumscription theories which characterize action narratives can be reduced to logically equivalent 1st-order theories under certain conditions. Section 1.8 proposes a solution to the ramification problem which is used in TAL. The RAH scenario is modified to incorporate this solution. Section 1.9 proposes a solution to the qualification problem which is used in TAL. The RAH scenario is again modified to incorporate this solution. Section 1.10 presents an extension to TAL which models the use of concurrent actions where complex types of interaction between such actions may occur. Section 1.11 presents an application of TAL to planning where it is shown how TAL can be used as a semantic framework in the development and implementation of TALplanner, an award winning automated planner. In Section 1.12, we conclude.

1.2 Basic Concepts

When using TAL, we assume there is an *agent* interested in reasoning about a specific *world*. This world might be formally defined, or it might be the "real world", in which case the agent can only reason about a formally defined abstraction of the real world. In either case, it is assumed that the world is dynamic, in the sense that the various properties or *features* of the world can change over time. Conceptually, any feature has a fluent function associated with it representing the stream of values associated with the feature at each state or temporal entity used in the formalism.

The TAL framework also permits the use of multiple *value domains*, which can be used for modeling different types of *objects* that might occur in the world which is being modeled. For example, the well-known blocks world contains blocks that can be stacked on top of each other. The blocks world can then be modeled using a value domain for blocks,

containing values such as A, B and C, together with parameterized Boolean-valued features representing relations such as on($block_1, block_2$), which holds iff $block_1$ is on top of $block_2$, and clear($block$), which holds iff there is no block on top of the given block. Of course, values can also be used to represent properties of objects rather than the objects themselves. For example, if the color of each block should be modeled, then this could be done using a value domain for colors containing values such as red, green and blue, together with a color-valued (non-boolean) feature color($block$). In summary, instantiated parameterized features take specific values (Boolean or non-Boolean) at specific timepoints. In this manner, both relations and properties are capable of being represented.

Time itself can be viewed differently depending on the nature of the world being reasoned about and the reasoning abilities of the agent. TAL offers a modular means of choosing the temporal structure to be used. Currently, TAL uses linear time structures, as opposed to branching time structures, and allows the use of either continuous real-valued time or discrete integer time. Research within the TAL framework has mostly been focused on discrete non-negative integer time structures, and such a structure will be used throughout this chapter.

The development of the world over a (possibly infinite) period of discrete time can be viewed in two different ways. Figure 1.1 shows what would happen in a simple blocks world scenario where block A is initially on top of B, which is on the table, and where one unstacks A from B, places it on the table, picks up B, and finally stacks this block on top of A. The information about this scenario can be viewed as a sequence of *states*, where each state provides a value to all features (or "state variables") for a single common timepoint, or as a set of *fluents*, where each fluent is a function of time which specifies the development of a single feature. We sometimes use the terms "feature" and "fluent" interchangeably to refer to either a specific property of the world or the function specifying its value over time.

Consequently, a logical model in TAL is a sequence of states indexed by time, where each state contains a value for each feature in the vocabulary at the time entity associated with the state. In the logical language, the assertion that a feature has a value at a specific time is denoted as $[\tau]\ f(\overline{\omega}) \mathrel{\hat{=}} \omega$ in the macro language $\mathcal{L}(\mathrm{ND})$ and $\mathit{Holds}(\tau, f(\overline{\omega}), \omega)$ in the logical language $\mathcal{L}(\mathrm{FL})$, where τ is a temporal expression, $f(\overline{\omega})$ is a parameterized feature and ω is a value from the feature's value domain.

Since there is an agent, there is usually also a set of *actions* that the agent can perform. Such actions can only be performed when the requisite *preconditions* are satisfied. Performing an action changes the state of the world according to a set of given rules. Such rules are not necessarily deterministic. For example, the action of tossing a coin can be modeled within the TAL framework, and there will be two possible result states. TAL offers a highly expressive language for specifying actions where non-deterministic, context-dependent, concurrent and durational actions are expressible, among other types of actions.

Background knowledge associated with a reasoning domain can be modeled in a number of ways in TAL. *Observation statements* represent observations made by an agent. *Domain Constraint statements* represent facts true in all scenarios associated with a particular reasoning domain. *Dependency Constraint statements* can be used to represent causal theories or assertions which model intricate dependencies describing how and when features change relative to each other.

All of these concepts are modeled in a narrative specified in the language $\mathcal{L}(\mathrm{ND})$.

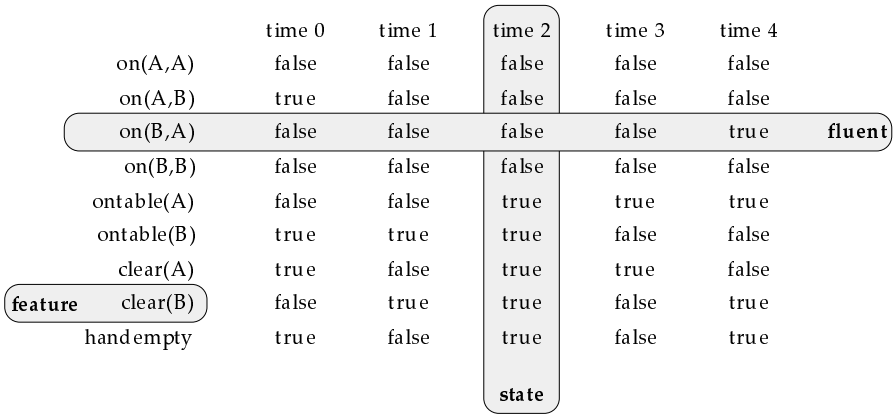

Figure 1.1: Viewing a Development as Fluents or States

$\mathcal{L}(\mathrm{ND})$ is a high-level extendable macro language which provides support to the knowledge engineer when constructing narratives and permits specification of narratives at a higher level of abstraction than logical statements. An extendable translation function is provided which translates narratives specified in $\mathcal{L}(\mathrm{ND})$ into 1st- and 2nd-order logical theories.

One of the fundamental problems in developing logics for reasoning about action and change has been in finding both representational and computationally efficient ways to encode the fact that there is a great deal of invariant structure in the world at a particular level of abstraction in which agents often reason about and describe the world. Even though the world is often dynamic and changing, from the perspective of an agent functioning in the world, properties and relations among entities are more often than not inert. On the other hand, there are often reasons for features in the world to change or reasons that provide the possibility for change. Many of these are obvious. For example, if an agent executes a physical action, the intent is usually to change some aspect of the world to the agent's advantage in completing a task. Others are less obvious, for example the subtle ramifications and aftereffects of an action. Developing theories of action and change is very much about identifying and representing normative rules which capture invariant and non-invariant epistemic and physical structure in environments in which agents are embedded and in which they operate.

Many of the representational and computational problems associated with modeling action and change have been given names, such as the frame, ramification and qualification problems, while others have not. Many useful techniques for capturing normative behavior have also been developed such as default reasoning. The principle intuition used in the development of TAL to deal with many of these issues is very simple to state, but quite difficult to make operational in an efficient manner in a logical formalism such as TAL.

In any TAL model, a time series is implicitly associated with any feature in the vocabulary. Whether a feature may change value or not in a transition from one time-point to another in the time series is specified by occluding or marking that feature as being given the *possibility* of changing value relative to other constraints in the theory. Policies for occluding features at time-points are both contextually and temporally dependent on a number of factors and done for a number of reasons. The definitions of the frame, ram-

ification and qualification problems specify some of these reasons. For whatever reason this is done, to the greatest extent possible, this labeling process should be achieved in a principled manner and remain more or less hidden from the knowledge engineer via the use of macro mechanisms in the $\mathcal{L}(\mathrm{ND})$ language and the translation into the base logical language $\mathcal{L}(\mathrm{FL})$.

At the level of $\mathcal{L}(\mathrm{ND})$, there are a number of ways to incrementally provide an occlusion policy for a feature, some more explicit than others. At the $\mathcal{L}(\mathrm{FL})$ level, the policies result in a set of labels for each feature represented as $Occlude(\tau, f(\overline{\omega}))$. The generation of such policies provide sufficient conditions for features being given the possibility to change value in state transitions (from $\tau - 1$ to τ). A circumscription policy then provides the necessary conditions and a definition of the occlusion predicate in the logical theory. An additional specification of whether and when a feature is persistent, durational, or dynamic in nature is also provided. These statements provide a means of filtering *bad* models out of the model set for a particular narrative, such as models where persistent features change value without being occluded.

If one uses this technique in a principled manner and restricts the generation of such policies to only include positive occurrences of the predicate $Occlude$ in the theory, then a reduction of the 2nd-order circumscription theory to a logically equivalent 1st-order theory is always guaranteed. It is in this manner we provide partial solutions to the frame, ramification and qualification problems in the context of TAL.

1.3 TAL Narratives

A narrative in $\mathcal{L}(\mathrm{ND})$ can be said to consist of two parts: The *narrative background specification* (NBS), which provides background information that is common to all narratives for a particular domain, and the *narrative specification* (NS), which provides information specific to a particular instance of a reasoning problem. Most of this information is represented as a set of labeled narrative statements in the surface language $\mathcal{L}(\mathrm{ND})$.

Before providing a formal definition of the $\mathcal{L}(\mathrm{ND})$ language, we will introduce most of the macros, formula types and statement classes using a rather complex example scenario called the *Russian Airplane Hijack Scenario* (RAH), which in order to be adequately represented in any logical formalism would require robust solutions to the frame, ramification and qualification problems. We say robust because a complete description of the RAH world requires the representation of concurrent actions, incomplete specifications of states, ramification with chaining, the use of non-boolean features, fine-grained dependencies among objects in different feature value domains, actions with duration, two types of qualification (*weak* and *strong*) and the use of explicit time, in addition to other features.

The RAH narrative description will be used as a vehicle for considering different facets of Temporal Action Logics and demonstrating how various aspects of a domain can be modeled in TAL. This will be done in stages. In this section, we will represent the narrative without the use of side effects and under the assumption that actions always succeed if their basic preconditions are satisfied. In other words, we will omit solutions to the ramification and qualification problems.[3] After having provided formal specifications of the $\mathcal{L}(\mathrm{ND})$ and

[3] This will initially result in a scenario where it is assumed that any attempt to board a plane always succeeds, regardless of whether a person carries a gun or is drunk. In addition, ramifications of action effects will be included in the action specification.

$\mathcal{L}(\mathrm{FL})$ languages (Sections 1.5 and 1.6), we will once more return to the RAH scenario in order to consider how ramification constraints (Section 1.8) and qualification constraints (Section 1.9) can be modeled in TAL.

1.3.1 The Russian Airplane Hijack Scenario

The Russian Airplane Hijack scenario[4] can be described as follows.

Example 1.1 (Russian Airplane Hijack Scenario) *A Russian businessman, Boris, 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.*

One ramification of moving between locations is that objects in your pocket will follow you from location to location. Similarly, a person on board a plane will follow the plane as it flies between cities.

Generally, when boarding a plane, the only preconditions are that you are at the gate and you have a ticket. However, if you try to board a plane carrying a gun in your pocket, which will be the case for Boris, this should qualify the action. Also, a condition that could sometimes qualify the boarding action is if you arrive at the gate in a sufficiently inebriated condition, as will be the case for Dimiter. When the boarding action is qualified, attempting to board should have no effect.

Boris, Erik and Dimiter already have their tickets. They start (concurrently) from their respective homes, stop by the office, go to the airport, and try to board flight SAS609 to Stockholm. Both Erik and Boris put combs in their pockets at home, and Boris picks up a gun at the office, while Dimiter is already drunk at home and may or may not already have a comb in his pocket. Who will successfully board the plane? What are their final locations? What will be in their pockets after attempting to board the plane and after the plane has arrived at its destination? ■

Let us assume that the scenario is encoded correctly in TAL and that we agree on our commonsense intuitions regarding what solutions to the frame, ramification and qualification problems would imply. Then the following inferences should be entailed by the logical theory associated with the RAH scenario:[5]

1. Erik will board the plane successfully, eventually ending up at his destination.

2. An indirect effect of flying is that a person ends up at the same location as the airplane he is on. In addition, because items in pockets follow a person, a transitive effect results where items in a person's pocket are at the same location as the plane which that person is on. Consequently, Erik's comb, comb2, will also end up at his destination.

[4] This scenario is an elaboration and concretization of a sketch for a scenario proposed by Vladimir Lifschitz in on-line discussions in the Electronic Transactions on Artificial Intelligence (ETAI/ENAI), and was previously published in [16, 41].

[5] Assume that Boris, Erik and Dimiter own the combs comb1,comb2 and comb3, respectively.

3. Boris will get as far as the airport with a gun and comb1 in his pocket. He will be unable to board the plane.

4. Dimiter will get as far as the airport, and may or may not be able to board the plane. If he is able to board the plane, he will eventually end up at his destination. Otherwise, he will remain at the airport. In any case, if he initially carried a comb, it will end up in the same location.

1.3.2 Narrative Background Specification

A narrative background specification contains a collection of statements of the following types:

- *Persistence statements* (labeled **per**) allow each fluent to be specified as being persistent (normally retaining its value from the previous timepoint), durational (normally reverting to a default value), or dynamic (varying freely, subject to other constraints involving this fluent).
- *Domain constraints* (labeled **dom**) characterize acausal information which is always true in the world being modeled.
- *Action type specifications* (labeled **acs**) provide generic definitions of action types.
- *Dependency constraints* (labeled **dep**) characterize causal and directional dependencies among fluents.

A narrative background specification also contains a vocabulary for the narrative. In the following subsections, each of the statement types and vocabulary specification will be described in detail and correlates to the RAH scenario will be listed.

Vocabulary

The vocabulary of an $\mathcal{L}(\mathrm{ND})$ narrative defines the constant symbols, feature symbols, action symbols, and other symbols that are available for use in narrative formulas. Since narrative examples used in the literature have traditionally been quite simple, the vocabulary has usually either been considered to be implicit in the remainder of the narrative specification or has been described informally in the main text of the article. Here, however, vocabularies will be described in terms of labeled narrative declaration statements using a syntax borrowed from the software tools VITAL [38] and TALplanner [40].

For the Russian Airplane Hijack scenario, we define a domain LOCATION for locations, and a domain THING containing everything that has a location. We also define the subdomains RUNWAY for LOCATIONs that are runways, PLANE for THINGs that are airplanes, PERSON for THINGs that are people, and PTHING for THINGs that people can pick up.

domain LOCATION :elements { home1,home2,home3,office,airport,run609,run609b,air }
domain THING :elements { gun,comb1,comb2,comb3,boris,dimiter,erik,sas609 }
domain RUNWAY :parent LOCATION :elements { run609,run609b }
domain PLANE :parent THING :elements { sas609 }
domain PERSON :parent THING :elements { boris,dimiter,erik }
domain PTHING :parent THING :elements { gun, comb1, comb2,comb3 }

We also use the boolean domain, which is present by default in all narratives and behaves as if it had been specified in the following manner:

domain BOOLEAN :elements { true, false }

Note that the domain specification in $\mathcal{L}(\text{ND})$ describes a type hierarchy. This will translate into the order-sorted vocabulary in the base logic $\mathcal{L}(\text{FL})$.

Finally, four fluents and four actions are used where the arguments to these are typed relative to the domain specification above.

fluent loc(THING) :domain LOCATION
fluent inpocket(PERSON, PTHING) :domain BOOLEAN
fluent onplane(PLANE, PERSON) :domain BOOLEAN
fluent drunk(PERSON) :domain BOOLEAN

action pickup(PERSON, PTHING)
action travel(PERSON, LOCATION, LOCATION)
action board(PERSON, PLANE)
action fly(PLANE, RUNWAY, RUNWAY)

Persistence Statements

Persistence statements are a novel feature of TAL and offer a very powerful and fine-grained mechanism for specifying inertia and default value assumptions for individual features when used together with the occlusion labeling mechanism mentioned previously. The majority of existing formalisms for action and change build in an assumption that a property or relation is either *always* assumed to be inert and subject to nochange by default or to be dynamic and subject to change by default. Through the use of persistence statements TAL permits the specification of contextually and temporally-dependent inertia and default assumptions per feature and down to the feature object level. This is an important feature of any action and change formalism since the inertial granularity of physical and other objects differs greatly. For example, a mountain will remain in place much longer than a ball on the ground which under certain weather conditions is in fact not inert at all.

Persistence statements can be used to classify features as being *persistent*, *durational*, or *dynamic*. In fact, a specific instantiated feature or set of features may be classified differently in the same scenario relative to context.

A feature declared as *persistent* at a time-point is only allowed to change value when an action, dependency constraint, or other constraint in the scenario allows it to change by implicitly labeling the feature at that time-point as being occluded (the *persistence assumption* or *inertia assumption*). Otherwise, it retains the same value it had at the previous timepoint. For example, the persistence statement below declares that all instantiated features of the form loc(*thing*) are inert at all timepoints: [6]

per $\forall t, \mathit{thing}\ [\mathit{Per}(t+1, \mathsf{loc}(\mathit{thing}))]$.

The translation of a *Per* declaration in $\mathcal{L}(\text{ND})$ into $\mathcal{L}(\text{FL})$ would be [7],

[6] Note that $t, t+1$ is used instead of $t, t-1$ due to the assumption of a non-negative time structure. Without this increment, the boundary condition $t = 0$, would not make sense.

[7] Note that in the following, any unbound variables as a result of a translation from $\mathcal{L}(\text{ND})$ to $\mathcal{L}(\text{FL})$ are assumed to be universally quantified.

$$\textit{Trans}(\textit{Per}(\tau, f)) = \forall t.\tau = t + 1 \wedge \neg \textit{Occlude}(t + 1, f) \rightarrow \forall v[\textit{Holds}(t + 1, f, v) \leftrightarrow \textit{Holds}(t, f, v)] \; .$$

Unless the feature is occluded at τ, it will retain its previous value.

A feature declared as *durational* is associated with a default value, and can only take on another value when an action, dependency, or other constraints allows it to (the *default value assumption*). At timepoints when no action or other constraint explicitly allows it to take on another value, it will immediately revert back to its default value. For instance, in the presence of a qualification to an action (see Section 1.9), the default assumption that it is possible to execute the action is violated. If the qualification does not apply, then by default it is possible to execute the action. TAL, through the use of durational features, can encode simple types of default rules and assumptions. For example, the persistence statement below declares that all instantiated features of the form $\mathsf{poss\text{-}board}(\textit{person}, \textit{plane})$ should have the default value true at all timepoints:

per $\forall t, \textit{person}, \textit{plane}\ [\textit{Dur}(t, \mathsf{poss\text{-}board}(\textit{person}, \textit{plane}), \mathsf{true})]$

The translation of a *Dur* declaration in $\mathcal{L}(\mathrm{ND})$ into $\mathcal{L}(\mathrm{FL})$ would be,

$$\textit{Trans}(\textit{Dur}(\tau, f, \omega)) = \neg \textit{Occlude}(\tau, f) \rightarrow \textit{Holds}(\tau, f, \omega)$$

Unless the feature is occluded at τ, it will retain its default value.

A TAL feature can also be *dynamic* if it is not declared to be persistent or durational. Since no persistence or default value assumption is applied, dynamic fluents can vary freely over time to satisfy observations and domain constraints.

Note that some earlier TAL logics (including **PMON**) used a fixed *nochange axiom* instead of persistence statements, forcing all fluents to be persistent. Using persistence statements provides a more flexible and fine-grained approach to controlling the default behavior of fluents and is currently the technique used in TAL to specify inertia and default value assumptions.

Intuitively, the features used in the Russian Airplane Hijack scenario describe properties that do not change unless something changes them. These features are all declared to be persistent. The declarations for the RAH scenario are,

per1 $\forall t, \textit{thing}\ [\textit{Per}(t + 1, \mathsf{loc}(\textit{thing}))]$
per2 $\forall t, \textit{person}, \textit{pthing}\ [\textit{Per}(t + 1, \mathsf{inpocket}(\textit{person}, \textit{pthing}))]$
per3 $\forall t, \textit{person}\ [\textit{Per}(t + 1, \mathsf{drunk}(\textit{person}))]$
per4 $\forall t, \textit{plane}, \textit{person}\ [\textit{Per}(t + 1, \mathsf{onplane}(\textit{plane}, \textit{person}))]$

Domain Constraints

Domain constraints represent knowledge about logical feature dependencies which are not specific to a particular reasoning problem instance but which are known to hold in every possible scenario taking place within a domain. An even stronger assumption often made in other formalisms is that these are formulas true in all states (universally quantified over all time-points, situations or states) and behave much as a classical logical formula would behave in a standard theory. In domain constraints, as well as other TAL formulas, the fact that a feature f takes on a particular value ω is denoted by the elementary fluent formula $f \hat{=} \omega$. For the boolean domain, the formula $f \hat{=} \mathsf{true}$ ($f \hat{=} \mathsf{false}$) can be abbreviated f ($\neg f$). Elementary fluent formulas can be combined using boolean connectives and quantification over values to form fluent formulas. The fixed fluent formula $[\tau]\ \phi$ states that the fluent formula ϕ holds at the timepoint τ.

For the Russian Airplane Hijack scenario we will define three domain constraints: No PTHING can be carried by two PERSONs at the same time, no PERSON can be on board two PLANEs at the same time, and any PTHING in a PERSON's pocket must be at the same location as that PERSON.

dom1 $\forall t, pthing, person_1, person_2\ [person_1 \neq person_2 \wedge [t]\ \mathsf{inpocket}(person_1, pthing) \rightarrow [t]\ \neg\mathsf{inpocket}(person_2, pthing)]$

dom2 $\forall t, person, plane_1, plane_2\ [plane_1 \neq plane_2 \wedge [t]\ \mathsf{onplane}(plane_1, person) \rightarrow [t]\ \neg\mathsf{onplane}(plane_2, person)]$

dom3 $\forall t, person, pthing\ [[t]\ \mathsf{inpocket}(person, pthing) \rightarrow [t]\ \mathsf{loc}(pthing) \hat{=} value(t, \mathsf{loc}(person))]$

Action Types

Actions can be invoked by the agent in order to change some properties in the world. If *person* picks up a thing *pthing* in the Russian Airplane Hijack scenario, then this should cause $\mathsf{inpocket}(person, pthing)$ to become true, for example. But since the inpocket feature is persistent, simply stating the fact that $\mathsf{inpocket}(person, pthing)$ will be true at the end of the action invocation is not sufficient. Instead, it is necessary to use a *reassignment macro* to explicitly release this feature from the persistence assumption at the specific point in time where it should change values from false to true.

There are three different reassignment macros: X, R and I. They can all be used with a temporal interval, for example $R((\tau, \tau')\ \alpha)$, or a single timepoint, for example $I([\tau]\ \alpha)$. Each of these operators has the effect of releasing the features occurring in α from the persistence and default value assumptions during the given interval or at the given timepoint. However, the operators differ in whether they place further constraints on the values of these features, and if so, at what time.

The X operator is used for *occlusion*. Its purpose is simply to allow the value of the features in the formula α to vary at a timepoint or during an interval, and therefore it does not further constrain the features occurring in α. Intuitively, the X operator occludes (hides) any changes in a feature value from the persistence or default value constraints generated by the persistence statements in the narrative.

The R operator is used for *reassignment*, and ensures that α will hold at the final timepoint in the interval. During the rest of the interval, the features occurring in α are allowed to vary freely, unaffected by the persistence or default value assumption (but still subject to other constraints that may also be present in the narrative).

The I operator is used for *interval reassignment* and ensures that α will hold during the entire interval. Note that if α is a disjunctive formula, features occurring in α may still vary during the interval as long as the formula remains satisfied throughout the interval.

An *action type specification* uses reassignment macros to define what will happen if and when a particular action is invoked. Note that it does not state that an action does occur. This is specified in the narrative specification using action occurrence statements.

In many existing action formalisms, actions do not have duration and are essentially single step. If actions with duration are introduced, it is often the case that during the duration nothing can happen or be specified to happen. TAL offers highly expressive action types. They can be single-step or durational, inert during the duration or highly dynamic. Additional constraints specifying what goes on during the execution of an action can easily be included in the action specification.

In the Russian Airplane Hijack scenario, four actions were declared in the narrative background specification. Here, those actions will be defined without taking qualifications into account and without making use of ramification constraints to specify side effects, resulting in a narrative where guns do not qualify the boarding action and where the fact that people inside an airplane move when the airplane moves must be expressed explicitly in the action definition. These action definitions will later be modified in Section 1.8.

acs1 $[t_1, t_2]\ \mathsf{fly}(\mathit{plane}, \mathit{runway}_1, \mathit{runway}_2) \rightsquigarrow \mathsf{loc}(\mathit{plane}) \mathrel{\hat{=}} \mathit{runway}_1 \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(\mathit{plane}) \mathrel{\hat{=}} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(\mathit{plane}) \mathrel{\hat{=}} \mathit{runway}_2) \wedge$
$\forall \mathit{person}[[t_1]\ \mathsf{onplane}(\mathit{plane}, \mathit{person}) \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(\mathit{person}) \mathrel{\hat{=}} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(\mathit{person}) \mathrel{\hat{=}} \mathit{runway}_2) \wedge$
$\forall \mathit{pthing}[[t_1]\ \mathsf{inpocket}(\mathit{person}, \mathit{pthing}) \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(\mathit{pthing}) \mathrel{\hat{=}} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(\mathit{pthing}) \mathrel{\hat{=}} \mathit{runway}_2)]]$

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

acs3 $[t_1, t_2]\ \mathsf{travel}(\mathit{person}, \mathit{loc}_1, \mathit{loc}_2) \rightsquigarrow [t_1]\ \mathsf{loc}(\mathit{person}) \mathrel{\hat{=}} \mathit{loc}_1 \rightarrow$
$R([t_2]\ \mathsf{loc}(\mathit{person}) \mathrel{\hat{=}} \mathit{loc}_2) \wedge$
$\forall \mathit{pthing}[[t_1]\ \mathsf{inpocket}(\mathit{person}, \mathit{pthing}) \rightarrow$
$I((t_1, t_2)\ \mathsf{loc}(\mathit{pthing}) \mathrel{\hat{=}} \mathsf{air}) \wedge R([t_2]\ \mathsf{loc}(\mathit{pthing}) \mathrel{\hat{=}} \mathit{loc}_2)]$

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

For reasons of representational efficiency, it is quite clear from observing these action specifications that a solution to the ramification problem is really necessary.

1.3.3 Narrative Specification

In the narrative specification, *observation statements* (labeled **obs**) represent observations of feature values at specific timepoints while *action occurrence statements* (labeled **occ**) specify which instances of the generic action types occur and during which time intervals.

Observation Statements

Observation statements are intended to describe specific facts that have been observed to hold in the world, permitting complete or incomplete specifications of the initial state or any other state in the world development corresponding to a narrative. They provide information about a particular reasoning problem instance within a domain, and are therefore part of the narrative specification.[8]

For this scenario, we define the initial locations of all things, as well as who is drunk in the initial state. On the other hand, we do not observe which things are in whose pockets.

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

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

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

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

[8] In some earlier versions of TAL, an explicit $Observe$ predicate was introduced in the base logical language $\mathcal{L}(\mathrm{FL})$ to which observation statements are translated. Distinguishing sensor generated facts about the world from other facts is useful when interfacing such logics to robotic systems. One might choose to view observation statements as perception statements, although this is not done in the current version of TAL.

Action Occurrence Statements

Action occurrence statements specify which actions actually do take place in a narrative. Like observations, they are part of the narrative specification – the instance-specific part of the narrative.

For the Russian Airplane Hijack scenario, the following action occurrences are also required. The exact timepoints used below were not specified in the RAH scenario, but have been chosen arbitrarily. Alternatively, exact timepoints could have been avoided by using non-numerical temporal constants. Note, however, that many of the actions are concurrent, sometimes with partially overlapping intervals.

occ1 $[1, 2]$ pickup(boris, comb1)
occ2 $[1, 2]$ pickup(erik, comb2)
occ3 $[2, 4]$ travel(dimiter, home3, office)
occ4 $[3, 5]$ travel(boris, home1, office)
occ5 $[4, 6]$ travel(erik, home2, office)
occ6 $[6, 7]$ pickup(boris, gun)
occ7 $[5, 7]$ travel(dimiter, office, airport)
occ8 $[7, 9]$ travel(erik, office, airport)
occ9 $[8, 10]$ travel(boris, office, airport)
occ10 $[9, 10]$ board(dimiter, sas609)
occ11 $[10, 11]$ board(boris, sas609)
occ12 $[11, 12]$ board(erik, sas609)
occ13 $[13, 16]$ fly(sas609, run609, run609b)

Note that this action scenario has been simplified for expository purposes. A number of additional extensions to the scenarios would in fact make it more realistic. For example, one could add more realistic timing actions, perhaps by explicitly modeling distances between locations and dividing by expected speed. In addition, upon introducing truly concurrent actions, one must be aware that there may be different types of interactions and these would have to be dealt with in an appropriate manner. TAL allows such extensions and we refer the interested reader to Section 1.10 where a summary of TAL concurrent actions is provided.

1.4 The Relation between the TAL languages $\mathcal{L}(\mathbf{ND})$ and $\mathcal{L}(\mathbf{FL})$

In order to reason about a particular narrative, it is first mechanically translated into the base language $\mathcal{L}(\text{FL})$, an order-sorted classical first-order language with equality using a linear discrete time structure (Figure 1.2). A circumscription policy is applied to the resulting theory, foundational axioms are added, and quantifier elimination techniques are used to reduce the resulting second order theory to first order logic. This is possible only under certain assumptions pertaining to the use of the $Occlude$ predicate and the nature of the temporal structure used.

In section 1.5, we will present the TAL surface language $\mathcal{L}(\text{ND})$. In section 1.6, we will present the TAL base language $\mathcal{L}(\text{FL})$, and in section 1.7, we will consider the circumscription policy used in TAL and reducibility results.

1.5 The TAL Surface Language $\mathcal{L}(\mathbf{ND})$

This section defines the surface language $\mathcal{L}(\text{ND})$. The translation to the first-order language $\mathcal{L}(\text{FL})$ is presented in Section 1.6.1. In the following, the overline is used as an

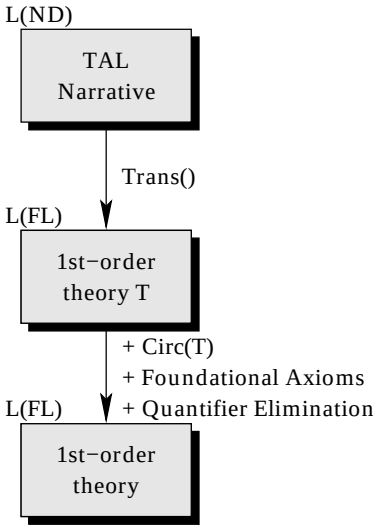

Figure 1.2: The relation between $\mathcal{L}(\text{ND})$ and $\mathcal{L}(\text{FL})$

abbreviation for a sequence, when the contents of the sequence are obvious. For example, $f(\overline{x}, \overline{y})$ means $f(x_1, ..., x_n, y_1, ..., y_m)$.

1.5.1 Sorts, Terms and Variables

Definition 1.1 (Basic Sorts) There are a number of sorts for values $\mathcal{V}_i$, including the boolean sort $\mathcal{B}$ with the constants $\{\mathsf{true}, \mathsf{false}\}$. TAL is order-sorted, and a sort may be specified to be a subsort of another sort. The sort $\mathcal{V}$ is a supersort of all value sorts.

There are a number of sorts for features $\mathcal{F}_i$, each one associated with a value sort $dom(\mathcal{F}_i) = \mathcal{V}_j$ for some j. The sort $\mathcal{F}$ is a supersort of all fluent sorts.

There is also a sort for actions $\mathcal{A}$ and a temporal sort $\mathcal{T}$. ■

The sort $\mathcal{T}$ is often assumed to be interpreted and semantic attachment is used in implementations, but it can be axiomatized in various ways, For example, in first-order logic, it can be axiomatized as a subset of Presburger arithmetic [37] (natural numbers with addition), or in second-order logic as Peano arithmetic.

Definition 1.2 (Terms) A *value term*, often denoted by ω, is a variable v or a constant v of sort $\mathcal{V}_i$ for some i, an expression *value*(τ, f) where τ is a temporal term and f is a fluent term, or an expression $\mathsf{g}(\omega_1, \ldots, \omega_n)$ where $\mathsf{g} : \mathcal{V}_{k_1} \times \ldots \times \mathcal{V}_{k_n} \rightarrow \mathcal{V}_i$ is a value function symbol and each ω_j is a value term of sort $\mathcal{V}_{k_j}$.

A *temporal term*, often denoted by τ, is a variable t or a constant $0, 1, 2, 3, \ldots$ or $\mathsf{s}_1, \mathsf{t}_1, \ldots$, or an expression of the form $\tau_1 + \tau_2$, all of sort $\mathcal{T}$.

A *fluent term*, often denoted by f, is a feature variable or a feature expression $\mathsf{f}(\omega_1, \ldots, \omega_n)$ where $\mathsf{f} : \mathcal{V}_{k_1} \times \ldots \times \mathcal{V}_{k_n} \rightarrow \mathcal{F}_i$ is a feature symbol and each ω_j is a value term of sort $\mathcal{V}_{k_j}$.

An *action term* Ψ is an expression $A(\omega_1, \ldots, \omega_n)$ where $A : \mathcal{V}_{k_1} \times \ldots \times \mathcal{V}_{k_n} \rightarrow \mathcal{A}$ is an action symbol and each ω_j is a value term of sort $\mathcal{V}_{k_j}$. ■

Variables are typed and range over the values belonging to a specific sort. Although the sort is sometimes specified explicitly in narratives, it is more common to simply give the variable the same name as the sort but (like all variables) written in italics, possibly with a prime and/or an index. For example, the variables $plane$, $plane'$ and $plane_3$ would be of the sort PLANE. Similarly, variables named t or τ are normally temporal variables, and variables named n are normally integer-valued variables.

The function *value*(τ, f) returns the value of the fluent f at the timepoint τ, where $[\tau]\ f \hat{=} v$ iff *value*$(\tau, f) = v$. The expression $[\tau]\ f \hat{=} g$, where f and g are fluent terms, is shorthand notation for $[\tau]\ f \hat{=}$ *value*(τ, g).

1.5.2 Formulas

Definition 1.3 (Temporal and Value Formulas) If τ and τ' are temporal terms, then $\tau = \tau'$, $\tau < \tau'$ and $\tau \leq \tau'$ are *temporal formulas*. A *value formula* is of the form $\omega = \omega'$ where ω and ω' are value terms, or $\mathsf{r}(\omega_1, \ldots, \omega_n)$ where $\mathsf{r} : \mathcal{V}_{k_1} \times \ldots \times \mathcal{V}_{k_n}$ is a relation symbol and each ω_j is a value term of sort $\mathcal{V}_{k_j}$. ■

We will sometimes write $\tau \leq \tau' < \tau''$ to denote the conjunction $\tau \leq \tau' \wedge \tau' < \tau''$, and similarly for other combinations of the relation symbols $\leq$ and $<$.

Definition 1.4 (Fluent Formula) An *elementary fluent formula*, sometimes called an *is-value expression*, has the form $f \hat{=} \omega$ where f is a fluent term of sort $\mathcal{F}_i$ and ω is a value term of sort $dom(\mathcal{F}_i)$. A *fluent formula* is an elementary fluent formula or a combination of fluent formulas formed with the standard logical connectives and quantification over values. ■

The elementary fluent formula $f \hat{=} \mathsf{true}$ ($f \hat{=} \mathsf{false}$) can be abbreviated f ($\neg f$).

Definition 1.5 (Timed Formulas) Let τ and τ' be temporal terms and α a fluent formula. Then:

- $[\tau, \tau']\ \alpha$, $(\tau, \tau']\ \alpha$, $[\tau, \tau')\ \alpha$, $(\tau, \tau')\ \alpha$, $[\tau, \infty)\ \alpha$, $(\tau, \infty)\ \alpha$ and $[\tau]\ \alpha$ are *fixed fluent formulas*,
- $C_T([\tau]\ \alpha)$, $C_F([\tau]\ \alpha)$ and $C([\tau]\ \alpha)$ are *change formulas*,
- $R([\tau, \tau']\ \alpha)$, $R((\tau, \tau']\ \alpha)$, $R([\tau, \tau')\ \alpha)$, $R((\tau, \tau'])\ \alpha)$ and $R([\tau]\ \alpha)$ are *reassignment formulas*,
- $I([\tau, \tau']\ \alpha)$, $I((\tau, \tau']\ \alpha)$, $I([\tau, \tau')\ \alpha)$, $I((\tau, \tau'])\ \alpha)$ and $I([\tau]\ \alpha)$ are *interval reassignment formulas*, and
- $X([\tau, \tau']\ \alpha)$, $X((\tau, \tau']\ \alpha)$, $X([\tau, \tau')\ \alpha)$, $X((\tau, \tau'])\ \alpha)$ and $X([\tau]\ \alpha)$ are *occlusion formulas*.

Fixed fluent formulas, change formulas, reassignment formulas, interval reassignment formulas and occlusion formulas are called *timed formulas*. ■

Definition 1.6 (Static Formula) A *static formula* is a temporal formula, a value formula, a fixed fluent formula, a change formula, `true`, `false`, or a combination of static formulas formed using the standard logical connectives together with quantification over values and time. ■

Note that the formulas `true` and `false` are not the same as the boolean values true and false.

Definition 1.7 (Change Formula) A *change formula* is a formula that has (or is regrettable to) the form $\mathcal{Q}\overline{v}(\alpha_1 \vee ... \vee \alpha_n)$ where $\mathcal{Q}\overline{v}$ is a sequence of quantifiers with variables, and each α_i is a conjunction of static, occlusion and reassignment formulas. The change formula is called *balanced* iff the following two conditions hold. (a) Whenever a feature $f(\overline{\omega})$ appears inside a reassignment or occlusion formula in one of the α_i disjuncts, it must also appear in all other α_i's inside a reassignment or occlusion formula with exactly the same temporal argument. (b) Any existentially quantified variable v in the formula, whenever appearing inside a reassignment or occlusion formula, only does so in the position $f \hat{=} v$. ■

Definition 1.8 (Application Formula) An *application formula* is any of the following: (a) a balanced change formula; (b) $\Lambda \rightarrow \Delta$, where Λ is a static formula and Δ is a balanced change formula; or (c) a combination of elements of types (a) and (b) formed with conjunction and universal quantification over values and time. ■

Application formulas will be used on the *rhs* of an implication in dependency constraints or action occurrence statements. The reason for these structural constraints on such formulas is to guarantee the proper generation of the occlusion predicate in the translation from $\mathcal{L}(\mathrm{ND})$ to $\mathcal{L}(\mathrm{FL})$. Restricting the structure of these formulas will guarantee 1st-order reducibility of the circumscription policy applied to the narrative.

Definition 1.9 (Occurrence Formula) An *occurrence formula* has the form $[\tau, \tau']\ \Psi$, where τ and τ' are temporal terms and Ψ is an action term. ■

Definition 1.10 (Persistence Formula) A *persistence formula* is an expression of the form *Per*(τ, f) where τ is a temporal term and f is a fluent term, an expression of the form *Dur*(τ, f, ω) where τ is a temporal term, f is a fluent term and ω is a value term, or a combination of persistence formulas formed with conjunction and universal quantification over values or time. ■

1.5.3 Statements

Definition 1.11 (Narrative Statements) The following types of narrative statements are available in the current version of TAL.

An action type specification or action schema (labeled **acs**) has the form $[t, t']\ \Psi \rightarrow \phi$, where t and t' are temporal variables, Ψ is an action term and ϕ is an application formula.

A dependency constraint (labeled **dep**) consists of an application formula.

A domain constraint (labeled **dom**) consists of a static formula.

A persistence statement (labeled **per**) consists of a persistence formula.

An observation statement (labeled **obs**) consists of a static formula.

An action occurrence statement (labeled **occ**) consists of an occurrence formula $[\tau, \tau']\ \Psi$ where τ and τ' are variable-free temporal terms and Ψ is a variable-free action term. ■

All of these statements types have been provided with intuitive meanings in the previous section except dependency constraints, which will be used to model side effects of actions (Sections 1.8) and qualifications to actions (Section 1.9).

1.6 The TAL Base Language $\mathcal{L}(\mathbf{FL})$

This section defines the current base language $\mathcal{L}(\mathrm{FL})$ used in TAL. The translation from $\mathcal{L}(\mathrm{ND})$,which has already been described, to the first-order language $\mathcal{L}(\mathrm{FL})$ is presented in Section 1.6.1. The base language $\mathcal{L}(\mathrm{FL})$ is an order-sorted classical first-order language with equality. We assume familiarity with standard ways to define vocabulary and variable types in sorted logics. Additionally, a temporal structure must be chosen for the temporal sort $\mathcal{T}$. This would include a domain such as the natural numbers or integers and associated operators. It was mentioned previously that a number of choices as to temporal structure could be made.

$\mathcal{L}(\mathrm{FL})$ currently uses the following predicates where $\mathcal{T}$ is the temporal sort, $\mathcal{F}$ is a supersort of all fluent sorts and $\mathcal{V}$ is a supersort of all value sorts.

- *Holds* : $\mathcal{T} \times \mathcal{F} \times \mathcal{V}$ – The *Holds* predicate expresses what value a feature has at each timepoint, and is used in the translation of fixed fluent formulas; for example, the formula $[0]\ \mathsf{loc}(\mathsf{boris}) \mathrel{\hat{=}} \mathsf{home1} \wedge \mathsf{loc}(\mathsf{gun}) \mathrel{\hat{=}} \mathsf{office}$ can be translated into $\mathit{Holds}(0, \mathsf{loc}(\mathsf{boris}), \mathsf{home1}) \wedge \mathit{Holds}(0, \mathsf{loc}(\mathsf{gun}), \mathsf{office})$.
- *Occlude* : $\mathcal{T} \times \mathcal{F}$ – The *Occlude* predicate expresses the fact that a persistent or durational feature is exempt from its persistence or default value assumption, respectively, at a given timepoint. It is used in the translation of the R, I and X operators, which are intended to change the values of features.
- *Occurs* : $\mathcal{T} \times \mathcal{T} \times \mathcal{A}$ – The *Occurs* predicate expresses that a certain action occurs during a certain time interval, and is used in the translation of action occurrence statements and action type specifications.

1.6.1 Translation from $\mathcal{L}(\mathbf{ND})$ to $\mathcal{L}(\mathbf{FL})$

The following translation function is used to translate $\mathcal{L}(\mathrm{ND})$ formulas into $\mathcal{L}(\mathrm{FL})$.

Definition 1.1 (*Trans* Translation Function) *Trans* is called the *expansion transformation,* and is defined as follows. All variables occurring only on the right-hand side are assumed to be fresh variables.

The formulas `true` and `false` need no translation:

$$\mathit{Trans}(\texttt{true}) = \texttt{true}$$
$$\mathit{Trans}(\texttt{false}) = \texttt{false}$$

Basic macros are translated into $\mathcal{L}(\mathrm{FL})$ predicates:

$$\mathit{Trans}([\tau]\ f(\overline{\omega})) = \mathit{Holds}(\tau, f(\overline{\omega}), \mathsf{true})$$
$$\mathit{Trans}([\tau]\ f(\overline{\omega}) \mathrel{\hat{=}} \omega) = \mathit{Holds}(\tau, f(\overline{\omega}), \omega)$$

$$\begin{aligned}
\mathit{Trans}(X([\tau]\ f(\overline{\omega}))) &= \mathit{Occlude}(\tau, f(\overline{\omega})) \\
\mathit{Trans}(X([\tau]\ f(\overline{\omega}) \mathrel{\hat{=}} \omega)) &= \mathit{Occlude}(\tau, f(\overline{\omega})) \\
\mathit{Trans}([\tau, \tau']\ A(\overline{\omega})) &= \mathit{Occurs}(\tau, \tau', A(\overline{\omega}))
\end{aligned}$$

In some versions of TAL, the $\mathcal{L}(\mathrm{ND})$ functions *Per* and *Dur* are also translated into $\mathcal{L}(\mathrm{FL})$ predicates. Here, they are translated directly into constraints on fluent values and occlusion.

$$\begin{aligned}
\mathit{Trans}(\mathit{Per}(\tau, f)) &= \forall t.\tau = t + 1 \wedge \neg \mathit{Occlude}(t + 1, f) \rightarrow \\
&\qquad \forall v[\mathit{Holds}(t + 1, f, v) \leftrightarrow \mathit{Holds}(t, f, v)] \\
\mathit{Trans}(\mathit{Dur}(\tau, f, \omega)) &= \neg \mathit{Occlude}(\tau, f) \rightarrow \mathit{Holds}(\tau, f, \omega)
\end{aligned}$$

Top-level connectives and quantifiers are left unchanged:

$$\begin{aligned}
\mathit{Trans}(\neg\alpha) &= \neg \mathit{Trans}(\alpha) \\
\mathit{Trans}(\alpha\ \mathcal{C}\ \beta) &= \mathit{Trans}(\alpha)\ \mathcal{C}\ \mathit{Trans}(\beta), \text{ where } \mathcal{C} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}. \\
\mathit{Trans}(\mathcal{Q}v[\alpha]) &= \mathcal{Q}v[\mathit{Trans}(\alpha)], \text{ where } \mathcal{Q} \in \{\forall, \exists\}.
\end{aligned}$$

Fixed fluent formulas can contain nested connectives and quantifiers, which are transferred outside the scope of the temporal context $[\tau]$.

$$\begin{aligned}
\mathit{Trans}([\tau]\ \mathcal{Q}v[\alpha]) &= \mathcal{Q}v[\mathit{Trans}([\tau]\ \alpha)], \text{ where } \mathcal{Q} \in \{\forall, \exists\}. \\
\mathit{Trans}([\tau]\ \neg\alpha) &= \neg \mathit{Trans}([\tau]\ \alpha) \\
\mathit{Trans}([\tau]\ \alpha\ \mathcal{C}\ \beta) &= \mathit{Trans}([\tau]\ \alpha)\ \mathcal{C}\ \mathit{Trans}([\tau]\ \beta), \text{ where } \mathcal{C} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}.
\end{aligned}$$

Nested connectives and quantifiers can also occur within occlusion formulas. However, the translation of these formulas has to be modified somewhat to take into account the fact that any occlusion formula should occlude *all* fluents occurring within the scope of the occlusion operator: Even a disjunctive formula such as $X([\tau]\ \alpha \vee \beta)$ should occlude all fluents in α *and* all fluents in β and is therefore not equivalent to $X([\tau]\ \alpha) \vee X([\tau]\ \beta)$ but to $X([\tau]\ \alpha) \wedge X([\tau]\ \beta)$. The translation procedure takes this into account by removing negations inside the X operator, translating connectives occurring inside X into conjunctions, and converting all quantifiers inside X into universal quantification.

$$\begin{aligned}
\mathit{Trans}(X([\tau]\ \neg\alpha)) &= \mathit{Trans}(X([\tau]\ \alpha)) \\
\mathit{Trans}(X([\tau]\ \alpha\ \mathcal{C}\ \beta)) &= \mathit{Trans}(X([\tau]\ \alpha) \wedge X([\tau]\ \beta)), \text{ where } \mathcal{C} \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}. \\
\mathit{Trans}(X([\tau]\mathcal{Q}v[\alpha])) &= \forall v[\mathit{Trans}(X([\tau]\alpha))], \text{ where } \mathcal{Q} \in \{\forall, \exists\}.
\end{aligned}$$

Fixed fluent formulas can contain infinite temporal intervals. This is a shorthand notation; infinity is not part of the temporal sort and disappears in the translation.

$$\begin{aligned}
\mathit{Trans}([\tau, \infty)\ \alpha) &= \forall t[\tau \leq t \rightarrow \mathit{Trans}([t]\alpha)] \\
\mathit{Trans}((\tau, \infty)\ \alpha) &= \forall t[\tau < t \rightarrow \mathit{Trans}([t]\alpha)]
\end{aligned}$$

Finite temporal intervals are permitted both in fixed fluent formulas and in the occlusion operator. Only one form of interval is shown; the extension to allow open, closed and semi-closed intervals is trivial.

$$\begin{aligned}
\mathit{Trans}([\tau, \tau']\ \alpha) &= \forall t[\tau \leq t \leq \tau' \rightarrow \mathit{Trans}([t]\alpha)] \\
\mathit{Trans}(X((\tau, \tau']\ \alpha)) &= \forall t[\tau < t \leq \tau' \rightarrow \mathit{Trans}(X([t]\alpha))]
\end{aligned}$$

The R and I operators are defined as follows. Again, one form of interval is shown.

$$\begin{aligned}
\mathit{Trans}(R((\tau, \tau']\ \alpha)) &= \mathit{Trans}(X((\tau, \tau'], \alpha)) \wedge \mathit{Trans}([\tau]\alpha) \\
\mathit{Trans}(R([\tau]\ \alpha)) &= \mathit{Trans}(X([\tau]\ \alpha)) \wedge \mathit{Trans}([\tau]\ \alpha) \\
\mathit{Trans}(I((\tau, \tau']\alpha)) &= \mathit{Trans}(X((\tau, \tau']\ \alpha)) \wedge \mathit{Trans}((\tau, \tau']\ \alpha) \\
\mathit{Trans}(I([\tau]\ \alpha)) &= \mathit{Trans}(X([\tau]\ \alpha)) \wedge \mathit{Trans}([\tau]\ \alpha)
\end{aligned}$$

Finally, the C_T "changes to true" operator is defined as follows, with the operators C_F (changes to false) and C (changes) added for symmetry.

$$\begin{aligned}
\mathit{Trans}(C_T([\tau]\ \alpha)) &= \forall t[\tau = t+1 \rightarrow \mathit{Trans}([t]\neg\alpha)] \wedge \mathit{Trans}([\tau]\ \alpha) \\
\mathit{Trans}(C_F([\tau]\ \alpha)) &= \forall t[\tau = t+1 \rightarrow \mathit{Trans}([t]\alpha)] \wedge \mathit{Trans}([\tau]\ \neg\alpha) \\
\mathit{Trans}(C([\tau]\ \alpha)) &= \mathit{Trans}(C_T([\tau]\ \alpha) \vee C_F([\tau]\ \alpha))
\end{aligned}$$

■

Example 1.2 (Narrative Translation) The following is a translation of several of the $\mathcal{L}(\mathrm{ND})$ statements in the Russian Airplane Hijack scenario into $\mathcal{L}(\mathrm{FL})$. For brevity, the translation is limited to one statement of each statement class; the remaining formulas are translated in a similar manner.

per1 $\forall t, \mathit{thing}, t'\ [t = t' + 1 \wedge \neg \mathit{Occlude}(t'+1, \mathsf{loc}(\mathit{thing})) \rightarrow$
$\forall v[\mathit{Holds}(t'+1, \mathsf{loc}(\mathit{thing}), v) \leftrightarrow \mathit{Holds}(t', \mathsf{loc}(\mathit{thing}), v)]]$

dom1 $\forall t, \mathit{pthing}_1, \mathit{person}_1, \mathit{person}_2\ [\neg(\mathit{person}_1 = \mathit{person}_2) \wedge$
$\mathit{Holds}(t, \mathsf{inpocket}(\mathit{person}_1, \mathit{pthing}_1), \mathsf{true}) \rightarrow$
$\neg \mathit{Holds}(t, \mathsf{inpocket}(\mathit{person}_2, \mathit{pthing}_1), \mathsf{true})]$

acs2 $\forall t_1, t_2, \mathit{person}, \mathit{pthing}\ [\mathit{Occurs}(t_1, t_2, \mathsf{pickup}(\mathit{person}, \mathit{pthing})) \rightarrow$
$(\mathit{Holds}(t_1, \mathsf{loc}(\mathit{person}), \mathit{value}(t_1, \mathsf{loc}(\mathit{pthing}))) \rightarrow$
$\mathit{Holds}(t_2, \mathsf{inpocket}(\mathit{person}, \mathit{pthing}), \mathsf{true}) \wedge$
$\forall t\ [t_1 < t \wedge t \leq t_2 \rightarrow \mathit{Occlude}(t, \mathsf{inpocket}(\mathit{person}, \mathit{pthing}))])]$

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

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

■

1.7 Circumscription and TAL

The commonsense intuition one would like to capture and formally model in TAL is the fact that at a particular level of abstraction, relations between and properties of objects generally have reasons for changing and if not, we can assume, unless observed otherwise, that these are the only *possible* changes we need be concerned about when reasoning about the specific environment around us and knowledge associated with that environment. So far, we have shown how one can encode in a principled manner sufficient reasons for the possibility of change by using a combination of the $Occlude$ predicate and automatic translations from the surface language $\mathcal{L}(\mathrm{ND})$ to the base language $\mathcal{L}(\mathrm{FL})$. When specifying narratives in TAL, all sufficient reasons for the possibility of change are specified using the reassignment macros R, I and X in dependency constraints and action type definitions. When translated, these statements result in constraints on the *Occlude* predicate. Unfortunately, what has been achieved so far under-constrains our notion of normative change, for we would also like to state that these are the only, or necessary, reasons for *possible* change.

In order to add this additional constraint to our action theories, we will appeal to the use of Circumscription [53, 54] with an additional twist. Rather than apply a circumscription policy to the whole action theory, the theory will be partitioned and we will apply circumscription selectively to different partitions. Although this technique, which we call filtered circumscription [19], is now commonly used in other action theories [62, 47], in the context of action and change, it was first proposed in Sandewall [57]. Here, it was

called filtered preferential entailment and was used as a basis for several of the definitions of preferential entailment in [58].

The basic idea will be to first circumscribe the predicate $Occlude$ in that part of the action theory containing action occurrence statements and dependency constraint statements. This will result in a set of preferred or minimal models for the action theory providing a definition of all time-points and features where it is possible for them to change value based on the constraints in the theory. Of course, these models will also contain spurious change since we have only provided sufficient and necessary conditions for the definition of $Occlude$. To rule out spurious change, we will then filter the resulting circumscriptive sub-theory with that part of the theory containing persistence statements. Persistence statements specify when features should not change value, assuming a predefined definition of $Occlude$ which circumscription provides. In this manner, any model containing feature change not mandated by the implicit occlusion policy in the action theory will be excluded as a model of the action theory. Additionally, a separate circumscription policy will be used for that part of the action theory containing action occurrence statements, where the predicate $Occurs$ will be circumscribed and all partitions will then be conjoined.

Due to the structural syntactic constraints built into statement definitions in $\mathcal{L}(\text{ND})$, we can show that the two circumscribed sub-theories which are 2nd-order due to the use of circumscription, can be reduced to logically equivalent first-order theories. In fact, since only positive occurrences of the predicates $Occlude$ and $Occurs$ occur in the two circumscribed partitions of the action theory, respectively, a standard syntactic transformation on formulas may be used to generate the necessary conditions for both predicates. This in fact is a form of predicate completion.

The formal definition of the circumscription policy used in TAL will use the following terminology:

- Let $\mathcal{N}$ denote the collection of narrative statements contained in a narrative in $\mathcal{L}(\text{ND})$, and let $\mathcal{N}_{\text{per}}$, $\mathcal{N}_{\text{obs}}$, $\mathcal{N}_{\text{occ}}$, $\mathcal{N}_{\text{acs}}$, $\mathcal{N}_{\text{domc}}$, and $\mathcal{N}_{\text{depc}}$ denote the sets of persistence statements, observation statements, action occurrence statements, action type specifications, domain constraint statements, and dependency constraint statements in $\mathcal{N}$, respectively.

- Let Γ denote the translation of $\mathcal{N}$ into $\mathcal{L}(\text{FL})$ using the *Trans* translation function, and let Γ_{per}, Γ_{obs}, Γ_{occ}, Γ_{acs}, Γ_{domc}, and Γ_{depc} denote the persistence formulas, observation formulas, action occurrence formulas, action type specifications, domain constraint formulas, and dependency constraint formulasin Γ, respectively.

- Let Γ_{fnd} denote the set of foundational axioms in $\mathcal{L}(\text{FL})$, containing unique names axioms, unique values axioms, etc.

- Let Γ_{time} denote the axiomatization of the particular temporal structure used in TAL.

In the following, we assume familiarity with circumscription [53, 54] and common notation used to denote circumscription policies [45]. Let

$$\Gamma = \Gamma_{\text{per}} \wedge \Gamma_{\text{obs}} \wedge \Gamma_{\text{domc}} \wedge \Gamma_{\text{occ}} \wedge \Gamma_{\text{depc}} \wedge \Gamma_{\text{acs}}$$

be the translation of an action narrative in $\mathcal{L}(\text{ND})$ into a first-order theory in $\mathcal{L}(\text{FL})$. Based on the discussion above, we use circumscription to minimize $Occurs$ in Γ_{occ} and $Occlude$

in $\Gamma_{\mathsf{depc}} \wedge \Gamma_{\mathsf{acs}}$ as follows,

$$\Gamma_1 = \Gamma_{\mathsf{per}} \wedge \Gamma_{\mathsf{obs}} \wedge \Gamma_{\mathsf{domc}} \wedge \mathit{Circ}(\Gamma_{\mathsf{occ}}; \mathit{Occurs}) \wedge \mathit{Circ}(\Gamma_{\mathsf{depc}} \wedge \Gamma_{\mathsf{acs}}; \mathit{Occlude}).$$

In addition, let

$$\Gamma_2 = \Gamma_{\mathsf{fnd}} \wedge \Gamma_{\mathsf{time}}.$$

For any narrative $\mathcal{N}$ in TAL, a preferred narrative theory in the base logic $\mathcal{L}(\mathrm{FL})$ is defined as,

$$\Delta_{\mathcal{N}} = \Gamma_2 \wedge \Gamma_1.$$

We say that a formula α in the base logic $\mathcal{L}(\mathrm{FL})$ is preferentially entailed by the narrative $\mathcal{N}$ whose translation into $\mathcal{L}(\mathrm{FL})$ is Γ iff

$$\Delta_{\mathcal{N}} \models \alpha.$$

Observe that there are several equivalent formalizations of $\Delta_{\mathcal{N}}$ due to the following general property of circumscription (p. 311, [45]): for any sentence B not containing P, Z (where P is minimized and Z is varied),

$$\mathit{Circ}(\Gamma(P, Z) \wedge B; P; Z) \equiv \mathit{Circ}(\Gamma(P, Z); P; Z) \wedge B \tag{1.1}$$

and the observation that

$$\mathit{Circ}(\Gamma_{\mathsf{occ}}; \mathit{Occurs}) \wedge \mathit{Circ}(\Gamma_{\mathsf{depc}} \wedge \Gamma_{\mathsf{acs}}; \mathit{Occlude}) \equiv \mathit{Circ}(\Gamma_{\mathsf{occ}} \wedge \Gamma_{\mathsf{depc}} \wedge \Gamma_{\mathsf{acs}}; \mathit{Occurs}, \mathit{Occlude})$$

From this, it follows that $\Delta_{\mathcal{N}}$ is equivalent to

$$\Delta' = \Gamma_{\mathsf{per}} \wedge \mathit{Circ}(\Gamma_2 \wedge \Gamma_{\mathsf{obs}} \wedge \Gamma_{\mathsf{domc}} \wedge \Gamma_{\mathsf{occ}} \wedge \Gamma_{\mathsf{depc}} \wedge \Gamma_{\mathsf{acs}}; \mathit{Occurs}, \mathit{Occlude})$$

Note that it is important that Γ_{per} is outside the circumscriptive theory due to the fact that it contains occurrences of the predicate $Occlude$. Consequently, filtered circumscription is fundamental to the approach used in TAL.

As it stands, $\Delta_{\mathcal{N}}$ is in fact, a second-order theory due to the fact that Γ_1 contains two second-order circumscription formulas, $\mathit{Circ}(\Gamma_{\mathsf{occ}}; \mathit{Occurs})$ and $\mathit{Circ}(\Gamma_{\mathsf{depc}} \wedge \Gamma_{\mathsf{acs}}; \mathit{Occlude})$. Due to structural syntactic constraints in the narrative $\mathcal{N}$ in $\mathcal{L}(\mathrm{ND})$ which are carried over to its translation Γ in $\mathcal{L}(\mathrm{FL})$, it can be shown that both $\mathit{Circ}(\Gamma_{\mathsf{occ}}; \mathit{Occurs})$ and $\mathit{Circ}(\Gamma_{\mathsf{depc}} \wedge \Gamma_{\mathsf{acs}}; \mathit{Occlude})$ are reducible to logically equivalent first-order formulas. We now show this. First some preliminaries.

An occurrence of a predicate symbol in a formula is *positive* if it is in the range of an even number of negations (this is assuming that the connectives $\rightarrow$ and $\leftrightarrow$ have been eliminated and replaced by other connectives in some equivalent normal form). A formula $A(P)$ is *positive* (relative to P) if all occurrences of P in $A(P)$ are positive.

Based on the definitions (1.7) and (1.8) for change and application formulas in the language of $\mathcal{L}(\mathrm{ND})$ and the definition of the translation function $Trans$ between formulas in $\mathcal{L}(\mathrm{ND})$ to their translation into $\mathcal{L}(\mathrm{FL})$, it is straightforward to show that the predicate $Occurs$ can only appear positively in Γ_{occ} and that the predicate $Occlude$ can only appear positively in $\Gamma_{\mathsf{depc}} \wedge \Gamma_{\mathsf{acs}}$. The following proposition can then be applied to show that both $\mathit{Circ}(\Gamma_{\mathsf{occ}}; \mathit{Occurs})$ and $\mathit{Circ}(\Gamma_{\mathsf{depc}} \wedge \Gamma_{\mathsf{acs}}; \mathit{Occlude})$ are reducible to logically equivalent first-order formulas [9]:

Proposition 1 (p. 316, [45]) If $A(P, Z)$ is positive relative to P, then the circumscription $Circ(A(P, Z); P; Z)$ is equivalent to

$$A(P, Z) \wedge \neg\exists x, z[P(x) \wedge A(\lambda y(P(y) \wedge x \neq y), z)].$$

■

In fact, it can be shown that predicate completion can be applied to Γ_{occ} and $\Gamma_{\text{depc}} \wedge \Gamma_{\text{acs}}$, respectively. The following proposition will be of use. Let $\overline{x}$ be a tuple of variables, and $F(\overline{x})$ be a formula with all parameters explicitly shown, then

Proposition 2 (p. 309, [45]) If $F(\overline{x})$ does not contain P, then the circumscription $Circ(\forall \overline{x} F(\overline{x}) \rightarrow P(\overline{x}); P)$ is equivalent to $\forall \overline{x} F(\overline{x}) \leftrightarrow P(\overline{x})$ ■

This proposition generalizes to conjunctions of formulas of the form $\forall \overline{x} F(\overline{x}) \rightarrow P(\overline{x})$. Using a number of syntactic transformations [13, 15], It can be shown that

$$\Gamma_{\text{occ}} = \bigwedge_{i=1}^{n} \forall \overline{x} F_i(\overline{x}) \rightarrow Occurs(\overline{x}) \tag{1.2}$$

where n is the number of $Occurs$ formulas in Γ_{occ}. By the generalization of proposition 2 and (1.2), it follows that

$$Circ(\Gamma_{\text{occ}}; \textit{Occurs}) = \forall \overline{x}[(\bigvee_{i=1}^{n} F_i(\overline{x})) \leftrightarrow Occurs(\overline{x})] \tag{1.3}$$

In addition it can be shown [13, 15] that

$$\Gamma_{\text{depc}} \wedge \Gamma_{\text{acs}} = [\bigwedge_{i=1}^{n} B_i \wedge \bigwedge_{j=1}^{k} C_j \wedge \forall \overline{x}[(\bigvee_{i=1}^{n} F_i(\overline{x}) \vee \bigvee_{j=1}^{k} G_j(\overline{x})) \rightarrow Occlude(\overline{x})]] \tag{1.4}$$

By (1.1), it follows that,

$$\begin{aligned} &Circ(\Gamma_{\text{depc}} \wedge \Gamma_{\text{acs}}; Occlude) = \\ &\textstyle\bigwedge_{i=1}^{n} B_i \wedge \bigwedge_{j=1}^{k} C_j \wedge Circ(\forall \overline{x}[(\bigvee_{i=1}^{n} F_i(\overline{x}) \vee \bigvee_{j=1}^{k} G_j(\overline{x})) \rightarrow Occlude(\overline{x})]; Occlude) \end{aligned} \tag{1.5}$$

By the generalization of proposition 2 and (1.5), it follows that

$$\begin{aligned} &Circ(\Gamma_{\text{depc}} \wedge \Gamma_{\text{acs}}; Occlude) = \\ &\quad \textstyle\bigwedge_{i=1}^{n} B_i \wedge \bigwedge_{j=1}^{k} C_j \wedge \forall \overline{x}[(\bigvee_{i=1}^{n} F_i(\overline{x}) \vee \bigvee_{j=1}^{k} G_j(\overline{x}) \leftrightarrow Occlude(\overline{x})] \end{aligned} \tag{1.6}$$

Consequently, one can reduce Γ_1 in $\Delta_{\mathcal{N}}$ to a logically equivalent first-order formula. Under the assumption that Γ_2 in $\Delta_{\mathcal{N}}$ is also first-order (the temporal structure has a first-order axiomatization), for any narrative $\mathcal{N}$, its translation into a preferred narrative in $\mathcal{L}(\text{FL})$, $\Delta_{\mathcal{N}}$, is a first-order theory.

Example 1.1 (Circumscription of the RAH scenario) Though the circumscription of the *Occlude* predicate can be translated into a single first order formula, we have instead chosen for expository purposes to generate a separate formula for each ground fluent in the narrative, where the conjunction of these formulas is entailed by the original 2nd-order

circumscription axiom. Here, we show a subset of these formulas for the Russian Airplane Hijack scenario.

First, the following are the necessary and sufficient conditions for loc(boris) to be occluded at any given point in time. For example, if boris is at home at time 3, which is a precondition for the action occurrence $[3, 5]$ travel(boris, home1, office), then his location will be occluded at time 5, when the final effects of the travel action take place.

$$\begin{aligned}&\forall t\ [\mathit{Occlude}(t, \mathsf{loc}(\mathsf{boris})) \leftrightarrow t = 5 \wedge \mathit{Holds}(3, \mathsf{loc}(\mathsf{boris}), \mathsf{home1}) \vee\\ &t = 10 \wedge \mathit{Holds}(8, \mathsf{loc}(\mathsf{boris}), \mathsf{office}) \vee t = 11 \wedge \mathit{Holds}(10, \mathsf{loc}(\mathsf{boris}), \mathsf{airport}) \vee\\ &14 \leq t \wedge t \leq 15 \wedge\\ &\mathit{Holds}(13, \mathsf{loc}(\mathsf{sas609}), \mathsf{run609}) \wedge \mathit{Holds}(13, \mathsf{onplane}(\mathsf{sas609}, \mathsf{boris}), \mathsf{true}) \vee\\ &t = 16 \wedge \mathit{Holds}(13, \mathsf{loc}(\mathsf{sas609}), \mathsf{run609}) \wedge \mathit{Holds}(13, \mathsf{onplane}(\mathsf{sas609}, \mathsf{boris}), \mathsf{true})]\end{aligned}$$

The conditions for occlusion of loc(dimiter) are quite similar, but differ in certain timepoints given that boris and dimiter do not travel at the same time.

$$\begin{aligned}&\forall t\ [\mathit{Occlude}(t, \mathsf{loc}(\mathsf{dimiter})) \leftrightarrow t = 4 \wedge \mathit{Holds}(2, \mathsf{loc}(\mathsf{dimiter}), \mathsf{home3}) \vee\\ &t = 7 \wedge \mathit{Holds}(5, \mathsf{loc}(\mathsf{dimiter}), \mathsf{office}) \vee t = 10 \wedge \mathit{Holds}(9, \mathsf{loc}(\mathsf{dimiter}), \mathsf{airport}) \vee\\ &14 \leq t \wedge t \leq 15 \wedge \mathit{Holds}(13, \mathsf{loc}(\mathsf{sas609}), \mathsf{run609}) \wedge\\ &\mathit{Holds}(13, \mathsf{onplane}(\mathsf{sas609}, \mathsf{dimiter}), \mathsf{true}) \vee\\ &t = 16 \wedge \mathit{Holds}(13, \mathsf{loc}(\mathsf{sas609}), \mathsf{run609}) \wedge \mathit{Holds}(13, \mathsf{onplane}(\mathsf{sas609}, \mathsf{dimiter}), \mathsf{true})]\end{aligned}$$

The fluent inpocket(boris,gun) may be occluded at time 7 if boris is at the required location when attempting to pick it up, but inpocket(dimiter,gun) can never be occluded.

$$\begin{aligned}&\forall t\ [\mathit{Occlude}(t, \mathsf{inpocket}(\mathsf{boris}, \mathsf{gun})) \leftrightarrow t = 7 \wedge \mathit{Holds}(6, \mathsf{loc}(\mathsf{boris}), \mathsf{loc}(\mathsf{gun}))]\\ &\forall t\ [\mathit{Occlude}(t, \mathsf{inpocket}(\mathsf{dimiter}, \mathsf{gun})) \leftrightarrow \mathtt{false}]\end{aligned}$$ ■

1.8 Representing Ramifications in TAL

The ramification problem [46, 20, 36, 47, 50, 21, 59, 25, 65] states that it is unreasonable to explicitly specify all the effects of an action in an action specification itself. One would rather prefer to state the direct effects of actions in the action specification and then use deductive machinery to derive the indirect effects of actions using the direct effects of actions together with general knowledge of *directional* dependencies among features specified in some background theory. The feature dependencies specified do not necessarily have to be based solely on notions of physical or other causality, but often are. A solution to the ramification problem is important from the representational perspective, where one strives after incremental, modular and intuitive characterizations of action and change. When one thinks of actions at a certain level of abstraction, one normally thinks of actions in terms of their direct effects and one would like to represent actions as such. On the other hand, causality plays an important role in any type of reasoning about action and change, therefore modular and incremental theories of causal and other dependencies among features is equally important to represent as is the interaction between actions and causal theories.

Some earlier approaches to solving the ramification problem made use of pure domain constraints (essentially logical implication) in order to infer side effects of actions. For the Russian Airplane Hijack scenario, for example, one might specify that everyone onboard an airplane is always in the same physical location as the airplane. Should one fly the airplane to another location, a direct effect would constrain the airplane to be in the new location, and the locations of everyone onboard would have to change location to the

airplane's new location in order to still satisfy the domain constraint. This type of solution is non-directional, which may sometimes be of advantage but may also lead to unexpected or unintended results. For example, in some representations, invoking an action that moves a single person would also cause the airplane and everyone else onboard to move.

The key insight in providing a good solution to the ramification problem is that of finding appropriate and representationally efficient ways of encoding directionality in dependencies among features which cause change in addition to allowing longer chains of directional feature change. Logical implication, for example, is one way to encode a dependency constraint among features, but it under-constrains the directionality of a dependency due to the fact that the contrapositive to an implication formula is logically equivalent to that formula. Another important point to keep in mind is the way in which dependencies are triggered. This is highly contextual and though it is often the case that change triggers change, it is also the case that state triggers change. Both types of context and combinations of both should be expressible in action theories.

The TAL solution to the ramification problem instead involves the use of dependency constraints, which were formally specified in Definition 1.11. In this definition, the similarity between action type specifications and dependency constraints may be noted. Whereas an action type specification is an application formula conditionalized by the occurrence of an action ($[t, t']\ \Psi$ where Ψ is an action term) and then a precondition once that action is invoked, a dependency constraint consists of an application formula without such an action occurrence precondition. In a sense, while actions must be explicitly invoked, dependency constraints are constantly active. In both cases, there is an explicit directionality of feature change implicit in the representation. Technically, this is achieved by noting that features are occluded via assignment operators on the rhs of implications, whereas they are not on the Lhasa. This together with the minimization policy for occlusion and persistence statements permits the encoding of directionality of change in a fine-grained manner.

This solution to the ramification problem can be directly applied to the Russian Airplane Hijack scenario. Recall that the definition of the travel and fly actions included formulas explicitly causing anything a person was carrying to move to the same destination (Section 1.3.2). Clearly it would be better if such a formula could be factored out and modeled as a side effect of a person moving between two locations in any manner, rather than having to be specified for every action that causes a person to move. This can be represented using the following feature dependency constraint, stating that if the fact that *person* is at *loc* becomes true (changes to true) – in other words, if the person has just moved to *loc* – then anything the person carries in his pockets will also move to the same location. The use of explicit reassignment with the R operator ensures that such changes are permitted despite the general persistence assumption for the loc fluent.

dep2 $\forall t, person, pthing, loc\ [[t]\ \mathsf{inpocket}(person, pthing) \wedge C_T([t]\ \mathsf{loc}(person) \hat{=} loc) \rightarrow R([t]\ \mathsf{loc}(pthing) \hat{=} loc)]$

With this change, the travel action can be simplified as follows:

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

The fly action can be simplified in a similar manner. Before showing the new definition of this action though, we will consider one more indirect effect: people on board an airplane move when the airplane moves.

dep1 $\forall t, plane, person, loc$ [
$[t]\ \mathsf{onplane}(plane, person) \wedge C_T([t]\ \mathsf{loc}(plane) \mathrel{\hat{=}} loc) \rightarrow R([t]\ \mathsf{loc}(person) \mathrel{\hat{=}} loc)]$

Note that the context for the dependency constraint in **dep1** has both a triggering condition (C_T) and a standard state condition. This is useful for encoding chaining of indirect effects.

Though this is quite similar to the previous indirect effect, it serves to illustrate an important property of fluent dependency constraints: It is possible to trigger not only a single indirect effect but a *chain* of indirect effects, which can be utilized to further modularize the specification of a narrative. In this particular scenario, causing an airplane to move will cause all people on board the airplane to move, which in turn will cause anything they are carrying to move, allowing the fly action to be modeled as follows:

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

1.9 Representing Qualifications in TAL

The qualification problem [64, 46, 20, 48, 50, 62, 16, 41, 66] was identified by McCarthy [52, 53] while developing systems for representing general commonsense knowledge. McCarthy showed a way to deal with the representational problem by using circumscription. In his own words,

> The "qualification problem" immediately arose in representing general commonsense knowledge. It seemed that in order to fully represent the conditions for the successful performance of an action, an impractical and implausible number of qualifications would have to be included in sentences expressing them. [53]

A solution to the qualification problem would involve a normative representation of an action which would model the fact that an action can be invoked unless *something* prevents it from being invoked, where that *something* is assumed by default not to exist unless explicitly represented in an action theory. Additionally, when qualifications to actions are learned, the representation should permit an incremental and elaboration tolerant means of adding such qualifications to the action theory.

We have now modeled most of the Russian Airplane Hijack Scenario in TAL, but we have not provided a means for modeling qualifications to actions in a representationally efficient, incremental and elaboration tolerant manner. Some examples of qualifications to actions in the RAH scenario would be: someone who carries a gun cannot board a plane, or someone who is drunk may or may not be able to board a plane. In fact, it may be the case that there are qualifications to qualifications. For example, a security personnel should be able to board a plane with a gun.

There are already a number of solutions to various aspects of the qualification problem in the literature, some of which would be applicable in TAL. However, many of these solutions are dependent on the two-state assumption with highly constrained action types. As we have shown, actions in TAL go far beyond this limited form of representation. We would like to provide a solution that retains at least the following features of TAL:

- Any state, including the initial state, can be completely or incompletely specified using observations and domain constraints.

- Actions can be context-dependent and non-deterministic. They can have duration and internal state, and the duration may be different for different executions of the action. There may be concurrent actions with partially overlapping execution intervals.

- There can be dynamic processes continuously taking place independently of any actions that may occur.

- Domain constraints can be used for specifying logical dependencies between fluents generally true in every state or across states. They may vary over time.

- Actions can have side effects, which may be delayed and may affect the world at multiple points in time. They may in turn trigger other delayed or non-delayed side effects.

We would also like to retain the first-order reducibility of the circumscription axiom in any solution to the qualification problem in TAL. In order to do this, the following restrictions and assumptions will apply. First, we will be satisfied with a solution where invoking a qualified action either has no effect or has some well-defined effect. Secondly, we will restrict the solution to the off-line planning and prediction problems and not claim a complete solution for the post-diction problem, which would require being able to conclude that an action was qualified because its successful execution would have contradicted an observation of some feature value after that action was invoked.

1.9.1 Enabling Fluents

To handle the qualification problem, we use a solution based on defaults where each action type in a narrative is associated with an *enabling fluent*, a boolean durational fluent with default value true and with the same number and type of arguments as the action type. This fluent will be used in the precondition of the action and will usually be named by prefixing "poss-" to the name of the action. For example, the boarding action in the RAH scenario will be associated with an enabling fluent poss-board$(person, plane)$. We also add a persistence statement for this fluent stating that it is a durational fluent. Recall that a durational feature retains its default value unless an additional constraint specifies that there is an exception to that value at a particular point or points in time. **acs4** is then modified as follows:

per5 $\forall t, person, plane\ [Dur(t, \text{poss-board}(person, plane), \text{true})]$

acs4′ $[t_1, t_2]\ \text{board}(person, plane) \rightsquigarrow$
$[t_1]\ \text{poss-board}(person, plane) \wedge \text{loc}(person) \hat{=} \text{airport} \rightarrow$
$R([t_2]\ \text{loc}(person) \hat{=} value(t_2, \text{loc}(plane)) \wedge \text{onplane}(plane, person))$

The other action types are modified in a similar way.

Now, suppose that board$(person, plane)$ is executed between timepoints t_1 and t_2. If poss-board$(person, plane)$ is false at t_1 for some reason, the action is qualified, or *disabled*. On the other hand, if the fluent is true at t_1, the action is *enabled*. Of course, it can still be the case that the action has no effects, if other parts of its precondition are false.

To generalize this, a context-independent action that should have no effect at all when qualified can be defined using a simple action definition of the form[9]

acs*m* $[t_1, t_2]$ *action* $\rightsquigarrow [t_1]$ *poss-action* $\wedge\, \alpha \rightarrow R([t_2]\ \beta)$

where α is the precondition and β specifies the direct effects of the action (context-dependent actions are defined analogously). However, we also wanted to be able to define actions that do have some effects when they are qualified. This can be done by defining a context-dependent action that defines what happens when the enabling fluent is false:

acs*n* $[t_1, t_2]$ *action* $\rightsquigarrow ([t_1]$ *poss-action* $\wedge\, \alpha_1 \rightarrow R([t_2]\ \beta_1)) \wedge$
$([t_1]\ \neg$*poss-action* $\wedge\, \alpha_2 \rightarrow R([t_2]\ \beta_2))$

For example, suppose that whenever anyone tries to board a plane but the action is qualified, they should try to find new transportation. In order to model this, we would add a new persistent fluent find-new-transportation(PERSON) : BOOLEAN and modify the boarding action from Section 1.3.2 as follows:

acs4″ $[t_1, t_2]$ board(*person*, *plane*) $\rightsquigarrow$
$([t_1]$ poss-board(*person*, *plane*) $\wedge$ loc(*person*) $\hat{=}$ airport $\rightarrow$
$R([t_2]$ loc(*person*) $\hat{=}$ *value*$(t_2,$ loc(*plane*)$) \wedge$ onplane(*plane*, *person*$))) \wedge$
$([t_1]\ \neg$poss-board(*person*, *plane*) $\wedge$ loc(*person*) $\hat{=}$ airport $\rightarrow$
$R([t_2]$ find-new-transportation(*person*$)))$

In this alternative scenario, if anyone is at the airport and tries to board a plane, and the action is qualified, they will have a goal of finding new transportation. If they are at the airport but the action is not qualified, they will board the plane. If they are not at the airport, none of the preconditions will be true, and invoking the action will have no effect. Note that it may very well be the case that they can not board for a more serious reason such as carrying a gun. This is a case where the original qualification might have to be qualified.

Regardless of whether a qualified action has an effect or not, its enabling fluent is a durational fluent with default value true. Therefore, the fluent will normally be true, and the action will normally be enabled. In the remainder of this section, we will examine some of the ways in which we can disable an action using strong and weak qualification.

1.9.2 Strong Qualification

When there is sufficient information to conclude that an action will definitely not succeed, it is *strongly qualified*. This can be modeled by forcing its enabling fluent to be false at the timepoint at which the action is invoked.

For example, suppose that when a person has a gun in his pocket, it should be impossible for that person to board a plane. Then, whenever inpocket(person, gun) holds, poss-board necessarily becomes false. This can be represented using a dependency constraint:

dep3 $\forall t, person, plane\ [[t]$ inpocket(*person*, gun$) \rightarrow I([t]\ \neg$poss-board(*person*, *plane*$))]$

At any timepoint t when a person has a gun in his pocket, we use the I macro both to occlude poss-board(*person*, *plane*) for all airplanes, thereby releasing it from the default

[9] Note that due to the regularity of the solution, such extensions could be implicit in an action macro, thus avoiding unneeded clutter in the representation and delegating representation responsibility to the system rather than the knowledge engineer.

value axiom, and to make it false. This implies that as long as a person has a gun in his pocket, poss-board will be false for that person on all airplanes. If the gun is later removed from the pocket, this dependency constraint will no longer be triggered. At that time, assuming no other qualifications affect the enabling fluent, it will automatically revert to its default value, true.

1.9.3 Weak Qualification

Although strong qualification can often be useful, we may sometimes have enough information to determine that an action *may* fail, even though we cannot conclusively prove that it will. We call this *weak* qualification.

For example, we may want to model the fact that when a person is drunk, he *may or may not* be able to board an airplane, depending on whether airport security discovers this or not. We may not be able to determine within our model of the RAH scenario whether airport security does discover that any given person is drunk. In this case, whenever $\mathsf{drunk}(\mathit{person})$ holds, we must release poss-board from the default value assumption:

dep4 $\forall t, \mathit{person}\ [[t]\ \mathsf{drunk}(\mathit{person}) \rightarrow X([t]\ \forall \mathit{plane}\ [\neg \mathsf{poss\text{-}board}(\mathit{person}, \mathit{plane})])]$

At any timepoint t when a person is drunk, we occlude $\mathsf{poss\text{-}board}(\mathit{person}, \mathit{plane})$ for all airplanes, but since we do not state anything about the *value* of the enabling fluent, it is allowed to be either true or false.

Although being able to state that an action *may* fail is useful in its own right, it is naturally also possible to restrict the set of models further by adding more statements to the scenario which could make it possible to infer whether $\mathsf{poss\text{-}board}(\mathsf{dimiter}, \mathsf{sas609})$ is true or false at some or all timepoints. For example, we may know that people boarding sas609 are always checked more carefully, so that it is impossible for anyone who is drunk to be on board that airplane, which could be expressed using an additional domain constraint. In the context of post-diction, observation statements could be used in a similar manner. For example, adding the observation statement **obs5** $[13]\ \mathsf{onplane}(\mathsf{sas609}, \mathsf{boris})$ to the narrative would allow us to infer that Boris did in fact board the plane and that $\mathsf{poss\text{-}board}(\mathsf{boris}, \mathsf{sas609})$ was in fact true. He would then end up at his intended destination. If instead we added the observation statement **obs6** $[13]\ \neg\mathsf{onplane}(\mathsf{sas609}, \mathsf{boris})$, we could infer that he was unable to board the plane and he did not end up at his destination.

It should be noted that this approach to modeling qualification has similarities to a standard default solution to the qualification problem, but with some subtle differences. For example, it permits more control of the enabling precondition, even allowing it to change during the execution of an action. More importantly, it involves no changes to the minimization policy already used in TAL to deal with the frame and ramification problems, consequently the circumscription theory is still first-order reducible.

1.9.4 Qualification: Not Only For Actions

As we have shown, this approach to qualification is based on general concepts such as durational features and fluent dependency constraints, instead of introducing new predicates, entailment relations or circumscription policies specifically designed for dealing with the qualification problem. This is appealing not only because we avoid introducing new complexity into the logic, but also because reusing these more general concepts adds to the

flexibility of the approach. In fact, exactly the same approach can be used for specifying qualifications to any rule or constraint. Most notably, one can provide qualifications for ramification constraints, thereby introducing defeasible side effects – or one can even qualify qualification constraints themselves.

As an example, when we initially considered the boarding action, the "natural" preconditions were that one had to be at the airport; this is the precondition encoded in the definition of board (**acs4**). Later, we found another condition that should qualify the action: No one should be able to board a plane carrying a gun. Now, however, we may discover that this qualification does not always hold: Airport security *should* be able to board a plane carrying a gun.

Assuming that there is a fluent is-security(PERSON) : BOOLEAN, this exception to the general qualification rule could of course be modeled by changing the dependency constraint **dep3** in the following way:

dep3′ $\forall t, person, plane\ [[t]\ \mathsf{inpocket}(person, \mathsf{gun}) \wedge \neg \mathsf{is\text{-}security}(person) \rightarrow I([t]\ \neg \mathsf{poss\text{-}board}(person, plane))]$

However, we may later discover additional conditions under which it should be possible for a person to board a plane with a gun, and we do not want to modify **dep3** each time. Instead, the qualification itself should be qualified. This can easily be done using the same approach as for actions. A new enabling fluent guns-forbidden(PERSON, PLANE) : BOOLEAN is added for the qualification constraint, and **dep3** is modified as follows:

dep3″ $\forall t, person, plane\ [[t]\ \mathsf{inpocket}(person, \mathsf{gun}) \wedge \mathsf{guns\text{-}forbidden}(person, plane) \rightarrow I([t]\ \neg \mathsf{poss\text{-}board}(person, plane))]$

Now, we can qualify the qualification **dep3** simply by making guns-forbidden false for some person and airplane. In order to do this, we add a new dependency constraint:

dep8 $\forall t, person, plane\ [[t]\ \mathsf{is\text{-}security}(person) \rightarrow I([t]\ \neg \mathsf{guns\text{-}forbidden}(person, plane))]$

1.9.5 Ramifications as Qualifications

A problem related to the qualification problem occurs in formalisms where ramification constraints and qualification constraints are expressed as domain constraints [20, 49]. Assume, for example, that we are reasoning about the blocks world, and that we have the following domain constraint (expressed using TAL syntax), stating that no two blocks can be on top of the same block:

dom $\forall t, x, y, z\ [[t]\ \mathsf{on}(x, z) \wedge \mathsf{on}(y, z) \rightarrow x = y]$

Now, suppose that the direct effect of the action put(A, C) is on(A, C), and the action is executed in a state where on(B, C) is true. Then, we cannot determine syntactically whether the domain constraint should be interpreted as a ramification constraint (since no two blocks can be on top of C, B must be removed) or as a qualification constraint (since no two blocks can be on top of C, the action should fail).

In TAL, however, all indirect effects of an action must be expressed as *directed* dependency constraints. Therefore, this problem simply does not arise. For example, if a ramification constraint is required, the following dependency constraint can be used:

dep $\forall t, x, y, z\ [[t]\ \mathsf{on}(x, z) \wedge C_T([t+1]\ \mathsf{on}(y, z)) \wedge x \neq y \rightarrow R([t+1]\ \neg \mathsf{on}(x, z))]$

If x is on z, and we then place y on z, then an indirect effect is that x is removed from z. On the other hand, if a qualification constraint is required, an enabling fluent

poss-put(BLOCK, BLOCK) can be used and the following qualification condition would then be added:

dep $\forall t, x, y, z\ [[t]\ \mathsf{on}(x, z) \wedge x \neq y \rightarrow I([t]\ \neg\mathsf{poss\text{-}put}(y, z))]$

Clearly, the problem of determining whether a constraint should be implicitly interpreted as a qualification or a ramification does not arise in this approach. One could criticize such a solution as over-constraining the action theory model, but then again, use of domain constraints could equally well be criticized for under-constraining the model.

A description of the TAL representation of the Russian Airplane Hijack scenario is now complete and the general methods used to resolve the frame, ramification and qualification problems have been described. The partial translations into $\mathcal{L}(\mathrm{FL})$ were done using VITAL [38], a research tool that can be used to study problems involving action and change within TAL and generate visualizations of action scenarios and preferred entailments.

1.10 Concurrent Actions in TAL

Much work in reasoning about action and change has been done under the (sometimes implicit) assumption that there is a single agent performing sequences of non-overlapping actions. The use of explicit metric time in TAL clearly enables the specification of narratives where action execution intervals are partly or completely overlapping, whether those actions are performed by a single agent or by multiple cooperating or adversarial agents. Similarly, the fact that actions can have non-unit duration and that one can specify in detail what happens during the execution interval enables richer domain models where a larger number of phenomena related to concurrency can be modeled. However, a complete treatment of concurrency also requires the ability to model interactions between concurrent effects of multiple actions. Such interactions can be *synergistic*, where two actions must be executed concurrently in order to achieve the desired effect. For example, moving a table requires lifting both sides of the table simultaneously in order to avoid the undesired side effects of everything on the table sliding off onto the floor. Interactions may also be *accumulative*, as when a number of agents are placing packages in a vehicle for transportation, or *harmful*, where one action provides the desired effect unless certain other actions are executed concurrently.

In each of these cases, the composite effect of executing several actions is not equivalent to the logical conjunction of the individual effects. For example, lifting the left side of the table causes the table to tilt, as does lifting the right side of the table, but lifting both sides at once cancels this effect. Though this could in theory be handled by modeling all possible interactions within each action definition, this would clearly be an extremely non-modular solution and would suffer from a combinatorial explosion in the number of conditional effects required in each action definition. This is especially true when dealing with actions with duration, where the number of combinations is determined not only by the number of actions but also by the number of ways two or more actions can overlap in time. The use of ramification constraints also complicates the issue by introducing interactions between actions and chains of (potentially delayed) ramification effects.

For these reasons, a more principled and indirect solution was proposed by Karlsson and Gustafsson [34], where actions do not directly change the state of the world but instead

produce a set of influences. Fluent dependency constraints can then be used to model how the world is affected by a combination of influences.

1.10.1 Independent Concurrent Actions

The use of independent concurrent actions involving disjoint sets of features is unproblematic in TAL. This is illustrated in the following narrative, describing a world with two types of actions (LightFire and PourWater), and a number of agents (bill and bob) and other objects (wood1). All variables appearing free are implicitly universally quantified.

acs1 $[s,t]\ \mathsf{LightFire}(a, \mathit{wood}) \rightarrow ([s]\ \mathsf{dry}(\mathit{wood}) \rightarrow R([s,t]\ \mathsf{fire}(\mathit{wood})))$
acs2 $[s,t]\ \mathsf{PourWater}(a, \mathit{wood}) \rightarrow R([s,t]\ \neg\mathsf{dry}(\mathit{wood}) \wedge \neg\mathsf{fire}(\mathit{wood}))$
obs1 $[0]\ \mathsf{dry}(\mathsf{wood1}) \wedge \neg\mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1})$
obs2 $[0]\ \mathsf{dry}(\mathsf{wood2}) \wedge \neg\mathsf{fire}(\mathsf{wood2}) \wedge \mathsf{wood}(\mathsf{wood2})$
occ1 $[2,7]\ \mathsf{LightFire}(\mathsf{bill}, \mathsf{wood1})$
occ2 $[2,7]\ \mathsf{LightFire}(\mathsf{bob}, \mathsf{wood2})$
occ3 $[9,12]\ \mathsf{PourWater}(\mathsf{bob}, \mathsf{wood1})$

The first action law states that if an agent a lights a fire using a piece of wood, and the wood is dry, then the wood will be on fire. The second action law states that if somebody pours water on an object, then the object will no longer be dry, and will cease being on fire. There are two pieces of wood (wood1 and wood2) which are initially dry and not burning. Two fires are lit by bill and bob during the temporal interval $[2,7]$, and then bob pours water on bill's fire at $[9,12]$. Since no concurrency is involved, the expected effects will take place: Both pieces of wood will be on fire at 7, and wood1 will no longer be burning at 12.

1.10.2 Interacting Concurrent Actions

Now consider the case where actions affecting the same fluents occur concurrently. For example, suppose bob pours water on wood1 while bill is still lighting the fire. Intuitively, the wood should not be on fire at 7. We formalize this in TAL by modifying **occ3**.

occ3 $[3,5]\ \mathsf{PourWater}(\mathsf{bob}, \mathsf{wood1})$

From the modified narrative one can still infer that wood1 is on fire at time 7, because the effects of $\mathsf{LightFire}(\mathsf{bill}, \mathsf{wood1})$ are only determined by whether the piece of wood is dry at time 2, whereas in reality the effects of any action may also be altered by the direct and indirect effects of other concurrent actions. A slight modification of the narrative above illustrates another problem. Assume that **occ3** is replaced with the following:

occ3 $[3,7]\ \mathsf{PourWater}(\mathsf{bob}, \mathsf{wood1})$

Now, the lighting and pouring actions end at the same time. From **acs1** and **occ1** one can infer the effect $[7]\ \mathsf{fire}(\mathsf{wood1})$ and from **acs2** and **occ3** one can infer $[7]\ \neg\mathsf{fire}(\mathsf{wood1})$. Both effects are asserted to be direct and indefeasible. Thus, the narrative becomes inconsistent. The conclusion one would like to obtain is again that the wood is not on fire.

1.10.3 Laws of Interaction

Karlsson and Gustafsson [34] considers two solutions to these problems.

In the first solution, action laws are extended to allow references to other action occurrences and the effects of LightFire are made conditional on the fact that there is no interfering PourWater action. As noted in the introduction, this solution makes action descriptions less modular and there may be a combinatorial explosion in the number of conditional effects for each action. Other problems include the fact that a concurrent action might only interfere with part of an action's effects, leading to further complexity in action laws.

The second solution is based on the assumption that interactions resulting from concurrency are best modeled not on the level of actions but on the level of features. Action laws encode the influences that an action has upon the environment of the agent; in the fire example, $[s,t]$ LightFire(a, wood) would have the effect $I((s,t]$ fire$^*(\mathit{wood}, \mathsf{true}))$ where fire$^*(\mathit{wood}, \mathsf{true})$ is a fluent representing an influence to make the feature fire(wood) true. This example follows the convention of representing the influences on an actual fluent $\mathsf{f}(\overline{\omega})$ with $\mathsf{f}^*(\overline{\omega}, v)$, where v is a value in the domain of f. Similarly, dependency constraints are modified to result in influences rather than actual fluent changes. The actual effects that these influences have on the environment are then specified in a special type of dependency laws called influence laws. Applying this solution to the fire example yields the following narrative:

dom1 $\mathit{Per}(\mathsf{fire}(\mathit{wood})) \wedge \mathit{Dur}(\mathsf{fire}^*(\mathit{wood}, v), \mathsf{false})$
dom2 $\mathit{Per}(\mathsf{dry}(\mathit{wood})) \wedge \mathit{Dur}(\mathsf{dry}^*(\mathit{wood}, v), \mathsf{false})$
acs1 $[s,t]$ LightFire$(a, \mathit{wood}) \rightarrow I((s,t]$ fire$^*(\mathit{wood}, \mathsf{true}))$
acs2 $[s,t]$ PourWater$(a, \mathit{wood}) \rightarrow I((s,t]$ dry$^*(\mathit{wood}, \mathsf{false}))$
dep1 $[s]\ \neg\mathsf{dry}(\mathit{wood}) \rightarrow I([s]$ fire$^*(\mathit{wood}, \mathsf{false}))$
inf1 $[s, s+3]$ fire$^*(\mathit{wood}, \mathsf{true}) \wedge \neg\mathsf{fire}^*(\mathit{wood}, \mathsf{false}) \rightarrow R([s+3]$ fire$(\mathit{wood}))$
inf2 $[s]$ fire$^*(\mathit{wood}, \mathsf{false}) \rightarrow R([s]\ \neg\mathsf{fire}(\mathit{wood}))$
inf3 $[s, s+3]$ dry$^*(\mathit{wood}, \mathsf{true}) \wedge \neg\mathsf{dry}^*(\mathit{wood}, \mathsf{false}) \rightarrow R([s+3]$ dry$(\mathit{wood}))$
inf4 $[s]$ dry$^*(\mathit{wood}, \mathsf{false}) \rightarrow R([s]\ \neg\mathsf{dry}(\mathit{wood}))$
obs1 $[0]\ \neg\mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{dry}(\mathsf{wood1})$
occ1 $[2,6]$ LightFire$(\mathsf{bill}, \mathsf{wood1})$
occ2 $[3,5]$ PourWater$(\mathsf{bob}, \mathsf{wood1})$

The action laws **acs1** and **acs2** and dependency law **dep1** produce influences; for example, **dep1** states that the fact that the wood is not dry produces an influence fire$^*(\mathit{wood}, \mathsf{false})$ to extinguish the fire (if there is one). The effects of these influences, alone and in combination, are specified in **infx**; for example, in order to affect the feature fire(wood), the influence fire$^*(\mathit{wood}, \mathsf{true})$ for starting the fire has to be applied without interference from fire$^*(\mathit{wood}, \mathsf{false})$ for an extended period of time. In the preferred models of this narrative, wood1 will be wet at $[4, \infty)$, implying that fire$^*(\mathsf{wood1}, \mathsf{false})$ will hold at $[4, \infty)$; consequently there is no interval $[s, s+3]$ where fire$^*(\mathsf{wood1}, \mathsf{true}) \wedge \neg\mathsf{fire}^*(\mathsf{wood1}, \mathsf{false})$, and fire$(\mathsf{wood1})$ will never become true.

The case when an effect of one action enables the effect of another action can also be handled with conditional influence laws. For instance, the following influence law states that opening a door requires initially keeping the latch open (the example is originally due to Allen [1]):

inf1 $[t]$ latch-open $\wedge\ [t, t+5]$ open$^*(\mathsf{true}) \rightarrow R([t+5]$ open$)$

Though not explicitly shown here, it is possible to use separate modular influence laws to specify the result of arbitrary combinations of influences, including combinations that lead to no effect at all. Influences can naturally also be combined with the TAL approach to ram-

ification, both in the sense that ramifications may lead to influences and in the sense that influences may cause chains of ramifications. One can also use influence laws to model resource conflicts, with either deterministic, non-deterministic or prioritized outcomes when two agents attempt to use the same resource [34, 24]. This results in a highly flexible and modular solution to many problems associated with concurrency, regardless of whether that concurrency is due to actions, ramifications or delayed effects.

1.11 An Application of TAL: TALplanner

The flexibility of TAL as a language for describing and modeling actions with concurrent effects, dependencies between fluents and other commonly occurring aspects of dynamic domains also makes it eminently suitable for modeling planning domains. This is especially true for planners that make extensive use of domain knowledge in various forms. For this reason, TAL has been used as the semantic basis for a planner called TALplanner [17, 18, 40], where TAL is used for modeling not only actions, initial states and standard state-based goals but also a set of control formulas acting as constraints on the set of valid plans. This latter use of logical formulas was initially inspired by the planner TLPLAN [2].

One of the intended uses of TAL in TALplanner is as a specification language providing a declarative semantics for planning domains and plans. This is an important difference from TLPLAN where only control formulas are based on the use of logic and actions are instead modeled using an operational semantics. But unlike Green's approach [22], which involved not only representing planning domains in logic but also *generating* plans using a resolution theorem prover, the declarative semantics of TAL currently serves mainly as a specification for the proper behavior of the planning algorithm. The TALplanner implementation generates plans using a procedural forward-chaining search method together with a search tree which is pruned with the help of temporal control formulas.

Given that performance is of paramount importance in a planner, the full expressivity of TAL is intended to be introduced into the planner implementation in stages; the full power of the language, including non-deterministic actions, chains of ramifications and arbitrary interactions between concurrent actions, must be approached carefully. Having the specification of the proper semantics of such constructs available from the beginning is useful even in the initial phase, providing a better view of what extensions will be desired in the future, which sometimes affects the basic framework of an implementation. Nonetheless, the language currently used for domain specifications in TALplanner is a subset of the full language for TAL described in this chapter.

Planning domains and planning problems also require the specification of certain types of information that were not originally supported in TAL or its predecessors. This required a set of new additions to the language which will be described.

Thus, both extensions and limitations relative to TAL are in order. This falls neatly within the TAL policy of providing macro languages adapted to specific tasks together with a translation into a single unified first-order base language $\mathcal{L}(\mathrm{FL})$ with a well-defined semantics and circumscription policy. While the details of the new macro language $\mathcal{L}(\mathrm{ND})^*$ is beyond the scope of this chapter [40], most of the extensions are used in the example planning domain discussed below.

Planning as Narrative Generation

TAL is based on the use of narratives, and automated planning can be viewed as a form of narrative generation where an initial narrative, specifying an initial state as well as various forms of domain knowledge, is incrementally extended by adding new action occurrences – in other words, new steps in the plan. The intention, then, is to generate a suitable set of action occurrences such that the desired goals are satisfied in the resulting complete narrative.

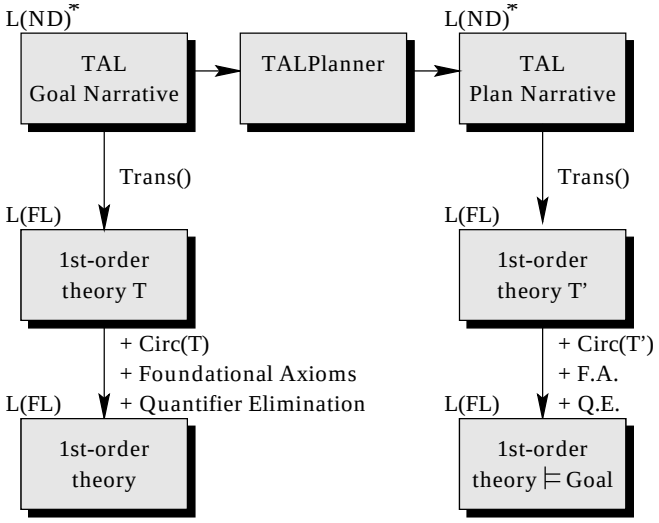

Figure 1.3: The relation between TAL and TALplanner

Figure 1.3 contains an extended version of the diagram previously shown in Figure 1.2. As seen in the top row of this figure, the input to TALplanner is a narrative in the extended macro language $\mathcal{L}(\mathrm{ND})^*$. This narrative is sometimes called a *goal narrative*, emphasizing the fact that it specifies a planning problem instance, and is usually denoted by $\mathcal{N}$. The goal narrative consists of two parts: A domain description, defining among other things the operators that are available to the planner, and a problem instance description, defining the initial state and the goal. TALplanner uses this high-level description of a planning problem to search for a set of TAL action occurrences (plan steps) that can be added to this narrative so that in the corresponding logical model, a goal state is reached. If this succeeds, the output is a new TAL narrative in $\mathcal{L}(\mathrm{ND})^*$ where the appropriate set of TAL action occurrences has been added. This narrative is sometimes called a *plan narrative*, emphasizing the fact that it represents a solution to a planning problem. Both goal narratives and plan narratives can be translated into $\mathcal{L}(\mathrm{FL})$ (the second row in the figure). As in pure TAL, a number of foundational axioms are required, and a standard TAL circumscription policy is applied, yielding complete definitions of the *Occlude* and *Occurs* predicates (the third row). Further details are available in [40].

Adding action occurrences to a standard TAL narrative is a non-monotonic operation, in the sense that conclusions entailed by the original narrative may have to be retracted

once a new action occurrence is added. However, at each step in the planning process, one would also prefer to be able to determine whether a certain conclusion will remain valid regardless of what new actions may be added to a plan. This is especially important in the context of temporal control formulas, where a candidate plan should not necessarily be discarded for violating a control formula if this violation might be "repaired" by adding new actions.

The key to solving this problem lies in the flexibility of the TAL solution to the frame problem. By selecting a search space where new action occurrences are constrained not to begin before any of the actions already present in the plan – that is, if there are actions beginning at times 0, 10 and 273, one cannot add a new action beginning at 272 – one can guarantee that along any branch of the forward-chaining search tree, there is a monotonically increasing temporal horizon such that any new effects introduced by future actions will take place strictly after this horizon [10]. Then, the standard definition of inertia can be altered to ensure that persistence is applied up to and including this temporal horizon, while leaving fluents unconstrained at all later timepoints. This is easily done by changing the TAL translation function while retaining the same circumscription policy.

It should be noted that this approach is not equivalent to assuming a complete lack of knowledge after the temporal horizon. On the contrary, any fluent constraints resulting from action effects or (in a future implementation) domain constraints are still equally valid after the temporal horizon; only the persistence assumption has been relaxed at those timepoints where the complete set of effects is unknown. Thus, it can still be possible to prove that a control formula has been definitely violated after the temporal horizon, which is essential for the performance of the concurrent version of TALplanner.

An Example Planning Domain

We will now show some examples of the use of $\mathcal{L}(\mathrm{ND})^*$ in modeling the timed version of the ZenoTravel domain, originally used in the AIPS 2002 International Planning Competition [44]. Due to space limitations, the complete domain description will not be provided. Nevertheless, the most pertinent aspects of the modeling language will be presented in sufficient detail.

The ZenoTravel domain contains a number of aircraft that can fly people between cities. There are five planning operators available: Persons may board and debark from aircraft, and aircraft may fly, zoom (fly quickly, using more fuel), and refuel. There are no restrictions on how many people an aircraft can carry. Flying and zooming are equivalent except that zooming is generally faster and uses more fuel. Figure 1.4 shows a tiny example problem, with arrows pointing out goal locations.

Objects in a planning problem are modeled using standard TAL values, and state variables are modeled using TAL fluents.

```
domain   THING :elements {...}
domain   AIRCRAFT :parent THING :elements {...}
domain   PERSON :parent THING :elements {...}
domain   CITY :elements {...}
```

[10] Note that this does not rule out the generation of plans with concurrent actions and one version of TALplanner does generate actions concurrently.

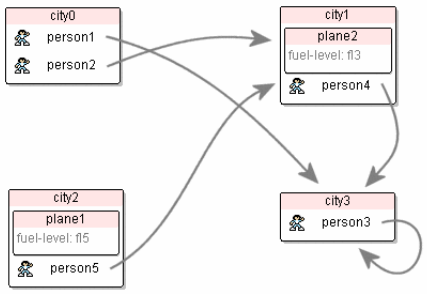

Figure 1.4: A ZenoTravel problem instance

feature at(THING, CITY), in(PERSON, AIRCRAFT) :domain BOOLEAN
feature fuel(AIRCRAFT) :domain INTEGER
...

Operators are modeled using a new form of operator statement, which uses a new syntax with explicit preconditions, prevail conditions, durations and effects. As specified by the competition organizers, the time required to board a plane is specified using the boarding-time fluent, which is here multiplied by 1000 in order to provide higher precision timing. Note also that the plane is required to remain at its location during boarding. The time required to fly between two cities is proportional to the distance and inversely proportional to the speed of the aircraft.[11]

operator board(*person*, *aircraft*, *city*) :at *s*
:duration *value*(t, 1000 * boarding-time) :as *dur*
:precond [*s*] at(*person*, *city*) $\wedge$ at(*aircraft*, *city*)
:prevail [*s*+1, *s*+*dur*] at(*aircraft*, *city*)
:effects [*s*+1] at(*person*, *city*) := false , [*s*+*dur*] in(*person*, *aircraft*) := true

operator fly(*aircraft*, $city_1$, $city_2$) :at *s*
:duration *value*(t, 1000 * distance($city_1$,$city_2$) / slow-speed(*aircraft*) :as *dur*
:precond [*s*] at(*aircraft*, $city_1$) $\wedge$ $city_1 \neq city_2$ $\wedge$
[*s*] fuel(*aircraft*) $\geq$ distance($city_1$, $city_2$) * slow-burn(*aircraft*)
:effects [*s*+1] at(*aircraft*, $city_1$) := false , [*s*+1] fuel-level(*aircraft*, $flevel_1$) := false ,
[*s*+*dur*] at(*aircraft*, $city_2$) := true , [*s*+*dur*] fuel-level(*aircraft*, $flevel_2$) := true

Control formulas specify constraints that must be satisfied in the logical model corresponding to a solution plan. In some respects, the central use of explicit control formulas is really what makes TALplanner stand out from other automated planing paradigms. Control formulas are intended to represent the high-level heuristics or commonsense smarts that one usually assumes a human might use when faced with specific planning problems in well-defined domains. Initially, a person may not have sufficient competence about a domain. Consequently, the plans generated may not be the best and will certainly take

[11] We appeal to the use of *semantic attachment* [68] techniques in the implementation of TAL and TALplanner by liberal use and invocation of built in mathematical and other functions associated with value domains for features.

longer to generate. As a person acquires a feel for a domain, certain constraints are then applied when generating plans which in turn minimize the search space. It is this intuition which is behind the use of control formulas as a domain dependent means of limiting the huge search space of action combinations one is faced with when using a forward chaining planner. The technique is also incremental in nature. Control formulas may be added incrementally as one learns more about the domain in question thus improving the efficiency of the planner.

The following two control formulas used in the ZenoTravel domain state that passengers should only board an aircraft when they desire to be in another city, and that they should only debark when they have reached their destination. Free variables are assumed to be universally quantified.

control :name "only-board-when-necessary"
$[t]\ \neg\mathsf{in}(person, aircraft) \wedge [t{+}1]\ \mathsf{in}(person, aircraft) \rightarrow$
$\exists city, city_2\ [\ [t]\ \mathsf{at}(person, city) \wedge \mathsf{goal}(\mathsf{at}(person, city_2)) \wedge city \neq city_2\ ]$

control :name "only-debark-when-in-goal-city"
$[t]\ \mathsf{in}(person, aircraft) \wedge [t{+}1]\ \neg\mathsf{in}(person, aircraft) \rightarrow$
$\exists city\ [\ [t]\ \mathsf{at}(aircraft, city) \wedge \mathsf{goal}(\mathsf{at}(person, city))\ ]$

In addition to these statements, which are valid in an entire planning domain, the planner also needs a complete specification of the initial state (using TAL observation statements) and a specification of the state-based goals that should be achieved. The latter is specified using goal statements, consisting of TAL fluent formulas that must hold in the final state resulting from executing a solution plan.
The following are possible goal and initial state statements for the example in Figure 1.4:

goal $\mathsf{at}(\mathsf{person1},\mathsf{city3}) \wedge \mathsf{at}(\mathsf{person2},\mathsf{city1}) \wedge \mathsf{at}(\mathsf{person3},\mathsf{city3}) \wedge \mathsf{at}(\mathsf{person4},\mathsf{city3}) \wedge \mathsf{at}(\mathsf{person5},\mathsf{city1})$

obs $\forall city\ [\ [0]\ \mathsf{at}(\mathsf{person1}, city) \leftrightarrow city = \mathsf{city0}\ ]$
obs $\forall city\ [\ [0]\ \mathsf{at}(\mathsf{person2}, city) \leftrightarrow city = \mathsf{city0}\ ]$
obs $\forall city\ [\ [0]\ \mathsf{at}(\mathsf{plane1}, city) \leftrightarrow city = \mathsf{city2}\ ]$
obs $[0]\ \mathsf{fuel}(\mathsf{plane1}) \hat{=} \mathsf{fl5}$

The main statement types for goal and plan narratives have now been introduced. A goal narrative is input to the forward-chaining TALplanner system and if possible, a TAL plan narrative is output from the planner which contains action occurrence statements and timings for such statements which entail the goal and control statements originally included in the goal narrative.

It was stated that the strategy used in TALplanner is not "planning as theorem-proving", but using TAL as a specification language for developing planners. Perhaps a better way to describe TAL and its relation to TALplanner is not only as a specification framework, but as "theorem-proving as an aid to plan generation". One can clearly see from Figure 1.3 that in the plan generation process, one can use TAL to reason about partial plans being generated. In fact, during plan generation, a simple form of inference is currently used to verify that control formulas are satisfied in theories associated with partial plans. In the plan execution process, one can use TAL to verify and monitor the plan execution process by querying the current state of a robotic system with TAL formulas. This is a form of on-line model-checking similar to the progression algorithms used in TALplanner

and TLplan. TALplanner also uses a limited form of resolution to reason about control formulas and operators during the initial (preprocessing) phase of the planning process, inferring a set of facts that must necessarily be true during the invocation of an operator and thereby improving the performance of checking control formulas during the planning phase. The flexible framework described in this section offers great potential for leveraging the use of logic with planning in a pragmatic and efficient manner.

1.12 Summary

This chapter provides a presentation of the latest stable version of TAL, a temporal action logic for reasoning about action and change. In the article, we present the basic narrative framework for specifying action scenarios using two languages $\mathcal{L}(\mathrm{ND})$ and $\mathcal{L}(\mathrm{FL})$. A definition of the circumscription policy used for TAL is provided in addition to proposals for partial solutions to the frame, ramification and qualification problems. Solutions are obviously dependent on the nature of the application domains to which they are applied. We say the solutions are partial because it is unclear whether such solutions would hold up practically unless one had specifications of the environments in which TAL would be used and a means of assessing whether the formalism would cover the spectrum of reasoning problems associated with a particular domain. Such a qualification would apply to any action and change formalism and assessments should be done using either formal assessment frameworks such as that described in the introduction to this chapter or empirical testing. TAL has been partially assessed for a particular type of application domain but much remains to be done in terms of assessing many of the newer extensions to TAL. That being said, TAL is one of the most expressive logical formalisms for reasoning about action and change, the underlying semantic framework is highly intuitive and TAL has been shown to correctly model the majority of benchmark problems proposed in the action and change research community. In the chapter, we have also provided a description as to how one could deal with the very complex problem of true concurrent actions and their interactions. We have concluded with an application of TAL to an award winning automated planner, TALplanner.

Acknowledgments

The TAL framework described in this chapter is very much a team effort and development has continued with spurts and lags beginning in 1994. We acknowledge contributions by the following members of the Knowledge Processing Lab at the Department of Computer and Information Science, Linköping University, Sweden: Marcus Bjäreland, Joakim Gustafsson, Lars Karlsson, Martin Magnusson, Andrzej Szałas and Witold Łukaszewicz.

The writing of this article and research included in it has been supported by funding from the Wallenberg Foundation under the WITAS UAV Project, Swedish Research Council grants (50405001,50405002) and a Swedish National Aeronautics Research grant (NFFP4-S4203).

Bibliography

[1] James Allen. Temporal reasoning and planning. In Pelavin Allen, Kautz and Tenenberg, editors, *Reasoning about Plans*. Morgan Kaufmann, 1991.

[2] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1–2):123–191, 2000. Available at ftp://newlogos.uwaterloo.ca/pub/bacchus/BKTlplan.ps.

[3] C. Baral and M. Gelfond. Logic programming and reasoning about actions. In Dov Gabbay Michael Fisher and Lluis Vila, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier Publications, 2005.

[4] 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.

[5] 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.

[6] P. Doherty. Notes on PMON circumscription. Technical Report LITH-IDA-94-43, Dept. of Computer and Information Science, Linköping University, Linköping, Sweden, December 1994. http://www.ida.liu.se/publications/techrep/94/trl94.html.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] Patrick Doherty. Reasoning about action and change using occlusion. In Anthony G. Cohn, editor, *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-1994)*, pages 401–405, Amsterdam, The Netherlands, 1994. John Wiley and Sons, Chichester, England. Available at ftp://ftp.ida.liu.se/pub/labs/kplab/people/patdo/ecai94.ps.gz.

[13] Patrick Doherty. PMON$^+$: A fluent logic for action and change, formal specification, version 1.0. Technical Report LITH-IDA-96-33, Department of Computer and Information Science, Linköping University, Linköping, Sweden, December 1996. Available at http://www.ida.liu.se/publications/techrep/96/tr96.html.

[14] Patrick Doherty and Joakim Gustafsson. Delayed effects of actions = direct effects + causal rules. *Linköping Electronic Articles in Computer and Information Science*, 3, 1998. Available at http://www.ep.liu.se/ea/cis/1998/001.

[15] Patrick Doherty, Joakim Gustafsson, Lars Karlsson, and Jonas Kvarnström. TAL:

Temporal Action Logics – language specification and tutorial. *Electronic Transactions on Artificial Intelligence*, 2(3–4):273–306, September 1998. Available at http://www.ep.liu.se/ej/etai/1998/009/.

[16] Patrick Doherty and Jonas Kvarnström. Tackling the qualification problem using fluent dependency constraints: Preliminary report. In Lina Khatib and Robert Morris, editors, *Proceedings of the Fifth International Workshop on Temporal Representation and Reasoning (TIME-1998)*, pages 97–104, Los Alamitos, California, USA, May 1998. IEEE Computer Society Press.

[17] Patrick Doherty and Jonas Kvarnström. TALplanner: An empirical investigation of a temporal logic-based forward chaining planner. In Claire Dixon and Michael Fisher, editors, *Proceedings of the Sixth International Workshop on Temporal Representation and Reasoning (TIME-1999)*, pages 47–54, Orlando, Florida, USA, May 1999. IEEE Computer Society Press. Available at ftp://ftp.ida.liu.se/pub/labs/kplab/people/patdo/time99-final.ps.gz.

[18] Patrick Doherty and Jonas Kvarnström. TALplanner: A temporal logic-based planner. *AI Magazine*, 22(3):95–102, 2001. See also http://www.aaai.org/Library/Magazine/Vol22/22-03/vol22-03.html.

[19] Patrick Doherty and Witold Łukaszewicz. Circumscribing Features and Fluents: A fluent logic for reasoning about action and change. In Dov M. Gabbay and Hans Jürgen Ohlbach, editors, *Proceedings of the First International Conference on Temporal Logic (ICTL-1994)*, volume 827 of *Lecture Notes in Artificial Intelligence*, pages 82–100. Springer Verlag London, 1994.

[20] M. L. Ginsberg and D. E. Smith. Reasoning about action II: The qualification problem. *Artificial Intelligence*, 35(3):311–342, 1988.

[21] E. Giunchiglia and V. Lifschitz. Dependent fluents. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-1995)*, pages 1964–1969, Montral, Qubec, Canada, 1995. Morgan Kaufmann Publishers, San Mateo, California, USA.

[22] Cordell Green. Applications of theorem proving to problem solving. In *Proceedings of the First International Joint Conference on Artificial Intelligence (IJCAI-1969)*. Morgan Kaufmann, 1969.

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

[24] Joakim Gustafsson. *Extending Temporal Action Logic*. PhD thesis, Linköping Studies in Science and Technology, Dissertation No. 689, 2001.

[25] Joakim Gustafsson and Patrick Doherty. Embracing occlusion in specifying the indirect effects of actions. In Luigia Carlucci Aiello, Jon Doyle, and Stuart C. Shapiro, editors, *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-1996)*, pages 87–98. Morgan Kaufmann Publishers, San Francisco, California, USA, 1996. Available at ftp://ftp.ida.liu.se/pub/labs/kplab/people/patdo/final-kr96.ps.gz.

[26] Joakim Gustafsson and Jonas Kvarnström. Elaboration tolerance through object-orientation. In *Proceedings of the Fifth Symposium on Logical Formalizations of Commonsense Reasoning (Common Sense-2001)*, 2001. Available at http://www.cs.nyu.edu/faculty/davise/commonsense01/final/kvarnstrom.ps.

[27] Joakim Gustafsson and Jonas Kvarnström. Elaboration tolerance through object-orientation. *Artificial Intelligence*, 153:239–285, March 2004.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] Lars Karlsson. *Actions, Interactions and Narratives*. PhD thesis, Linköping Studies in Science and Technology, Dissertation No. 593, 1999.

[34] Lars Karlsson and Joakim Gustafsson. Reasoning about concurrent interaction. *Journal of Logic and Computation*, 9(5):623–650, October 1999.

[35] Lars Karlsson, Joakim Gustafsson, and Patrick Doherty. Delayed effects of actions. In Henri Prade, editor, *Proceedings of the Thirteenth European Conference on Artificial Intelligence (ECAI-1998)*, pages 542–546, Brighton, UK, Aug 1998. John Wiley and Sons, Chichester, England. Available at ftp://ftp.ida.liu.se/pub/labs/kplab/people/patdo/ecai98.ps.gz.

[36] G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects. In *Proceedings of the International Conference on Knowledge Representation and Reasoning (KR)*, pages 341–350. Morgan Kaufmann, 1994.

[37] M. Koubarakis. Complexity results for first-order theories of temporal constraints. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-1994)*, pages 379–390. Morgan Kaufmann Publishers, San Francisco, California, USA, 1994.

[38] Jonas Kvarnström. VITAL. An on-line system for reasoning about action and change using TAL. Software available at http://www.ida.liu.se/~jonkv/vital/, 1997–2005.

[39] Jonas Kvarnström. Applying domain analysis techniques for domain-dependent control in TALplanner. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2002)*, pages 101–110, Toulouse, France, April 2002. AAAI Press, Menlo Park, California, USA.

[40] Jonas Kvarnström. *TALplanner and Other Extensions to Temporal Action Logic*. PhD thesis, Linköpings universitet, April 2005. Linköping Studies in Science and Technology, Dissertation no. 937.

[41] Jonas Kvarnström and Patrick Doherty. Tackling the qualification problem using fluent dependency constraints. *Computational Intelligence*, 16(2):169–209, May 2000.

[42] Jonas Kvarnström and Patrick Doherty. TALplanner: A temporal logic based forward chaining planner. *Annals of Mathematics and Artificial Intelligence*, 30:119–169, 2000.

[43] Jonas Kvarnström, Patrick Doherty, and Patrik Haslum. Extending TALplanner with concurrency and resources. In Werner Horn, editor, *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI-2000)*, Frontiers in Artificial Intelligence and Applications, pages 501–505, Berlin, Germany, August 2000. IOS Press, Amsterdam, The Netherlands. Available at ftp://ftp.ida.liu.se/pub/labs/kplab/people/patdo/www-ecai.ps.gz.

[44] Jonas Kvarnström and Martin Magnusson. TALplanner in the Third International Planning Competition: Extensions and control rules. *Journal of Artificial Intelligence Research*, 20:343–377, December 2003. Available at http://www.jair.org/contents/v20.html.

[45] V. Lifschitz. Circumscription. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Artificial Intelligence and Logic Programming*, volume 3, pages 297–352. Oxford University Press, 1991.

[46] Vladimir Lifschitz. Formal theories of action. In Frank M. Brown, editor, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*, pages 35–58, Lawrence, Kansas, USA, April 1987. Morgan Kaufmann Publishers, Los Altos, California, USA.

[47] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-1995)*, Montral, Qubec, Canada, August 1995. Morgan Kaufmann Publishers, San Francisco, California, USA.

[48] Fangzhen Lin and Ray Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994.

[49] Fangzhen Lin and Ray Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994.

[50] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1978–1984, 1995.

[51] John McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London, 1959. Her Majesty's Stationary Office. Available at http://www-formal.stanford.edu/jmc/mcc59.html.

[52] John McCarthy. Epistemological problems of artificial intelligence. In Raj Reddy, editor, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-1977)*, pages 1038–1044, Cambridge, Massachusetts, USA, 1977. William Kaufmann.

[53] John McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1–2):27–39, 1980. Reprinted in **(author?)** [55].

[54] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28(1):89–116, 1986. Reprinted in [55].

[55] John McCarthy. *Formalization of common sense, papers by John McCarthy edited by V. Lifschitz*. Ablex, 1990.

[56] Ray Reiter. *Knowledge in Action*. MIT Press, 2001.

[57] E. Sandewall. Filter preferential entailment for the logic of action and change. In N. S. Sridharan, editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-1989)*, San Francisco, August 1989. Morgan Kaufmann Publishers, San Mateo, California, USA.

[58] E. Sandewall. *Features and Fluents: A Systematic Approach to the Representation of Knowledge about Dynamical Systems*, volume 1. Oxford University Press, 1994.

[59] E. Sandewall. Assessments of ramification methods that use static constraints. In *Proceedings of the International Conference on Knowledge Representation and Reasoning (KR)*, pages 99–110. Morgan Kaufmann, 1996.

[60] E. Sandewall. Cognitive robotics and its metatheory:features and fluents revisited. *Electronic Transactions on Artificial Intelligence*, 2(3-4), 1998. http://www.ep.liu.se/ej/etai/1998/010/.

[61] 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.

[62] M. Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. The MIT Press, Cambridge, Massachusetts, USA, 1997.

[63] Yoav Shoham. Nonmonotonic logics: Meaning and utility. In John P. McDermott, editor, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-1987)*, pages 388–393, Milan, Italy, August 1987. Morgan Kaufmann Publishers, Los Altos, California, USA.

[64] Yoav Shoham. *Reasoning about Action and Change*. MIT Press, 1987.

[65] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1-2):317–364, 1997.

[66] M. Thielscher. Ramification and causalitythe qualification problem: A solution to the problem of anomalous models. *Artificial Intelligence*, 131(1-2):1–37, 2001.

[67] Michael Thielscher. *Reasoning Robots – The Art and Science of Programming Robotic Agents*. Springer, 2005.

[68] R. Weyhrauch. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, 13(1-2), 1980.