Alan: An Action Language for Non-Markovian Domains

Graciela González
Dept. of Computer Science
Sam Houston State University
Huntsville, TX, 77341 U.S.A.
csc_ghg@shsu.edu

Chitta Baral
Dept. of Computer Science and Eng.
Arizona State University
Tempe, AZ 85287, U.S.A.
chitta@asu.edu

Michael Gelfond
Dept. of Computer Science
Texas Tech University
Lubbock, TX, 79409 U.S.A.
mgelfond@cs.ttu.edu

Abstract

In this paper we present the syntax and semantics of a temporal action language named *Alan*. The language allows the specification of systems where the future state of the world depends not only on the current state, but also on the past states of the world, that is, where the Markov property does not hold. To the best of our knowledge, *Alan* is the first action language which incorporates causality with temporal formulas. In the process of defining the effect of actions we define the closure with respect to a path rather than to a state, and show that the non-Markovian model is an extension of the traditional Markovian model. Finally, we establish the relationship between theories of *Alan* and logic programs.

1 Introduction and motivation

There are certain areas of knowledge where conditions for actions are better described when one is allowed to reference not only the current state of the world, but also past states. For example, consider a multimodal interaction navigation system that needs to process utterances and touch-screen input. If the user utters "go there" and then points to an intersection on the screen, then the system should generate driving directions from the current location of the user to the given intersection. Just the utterance or just the pointing might have different meanings if not in the given sequence. would not have the same effect. Consider also a diagnostic system where facts that occur over a period of time are used to help make more accurate diagnosis. For example, a patient who presents a dry cough and trouble breathing with no other previous symptoms might not be suspected of having SARS, unlike one with the same current symptoms, but who suffered from fever *before* generalized body aches, all of these *preceded* by a trip to a SARS-infected area.

Multimedia presentations are also rich in temporal relationships. Consider the following description.

A tutorial about endangered species is available at the local zoo for the public to explore. All the contents of the tutorial are in a database, and it shows each animal as a multimedia presentation. A video clip of the animal is followed by descriptive paragraph, read aloud and shown as text. With the description, pictures of the animal and its habitat are displayed, each for 5 seconds. Once the paragraph is read, background music is played while all the pictures are shown.

Now suppose we would like to study weather the user clicked anywhere on the screen while the video was playing, or if showing the pictures took more than twice the time it took to read the descriptive paragraph. In order to be able to answer such queries, and many others, we first need to model the presentation. A model of a presentation provides a declarative description of its contents and the relationships among the different elements that define it.

A particularly challenging aspect of the application area is to model constraints that require knowledge of events that occurred sometime during the presentation, but not necessarily in the current state. Say that at the end of the presentation of one animal, the user is asked to answer a question about it, and is given two chances to answer it. If the wrong answer is entered twice, the portion of the presentation that showed the relevant content is shown again. To model the effect of submitting an incorrect answer, we would need to refer to weather or not an incorrect answer has been submitted sometime in the past. The two incorrect attempts might not even be one after the other, since the user could have gone back and watched some portion of the presentation again before coming back to the questions.

We present here the syntax and semantics of a temporal action language named *Alan*, introduced in [8], which can handle such domains. It was originally designed to model complex interactive multimedia presentations, joining the quest for a model of the complex temporal relationships in multimedia objects and presentations ongoing in the literature for some years [10; 11; 12; 18; 20].

The syntax of the proposed language is based on the action description language $\mathcal{A}$ [6] and its successors (such as [3; 14]). Other action description languages are also outlined in [19]. In these languages the effect of actions are represented using effect propositions of the form a **causes** λ *if* p, where a is an action and λ and p are fluent formulas. Intuitively, the meaning of such a proposition is that if p is true in a state then executing a in that state causes λ to be true in the resulting state. Most of these languages (except [16; 5]) are Markovian, in the sense that the effect of an action depends only on the state the action is executed.

We propose instead a non-Markovian action description

language, allowing p to be a formula with past temporal operators and action occurrence statements. In that case the effect of a is not with respect to a state but is with respect to a history of states. Infinite domains are avoided by using only past temporal operators in the language. The idea of using temporal logic to extend Markovian formalisms to capture non-Markovian domains has been explored before in [1], where the area of interest is rewarding behaviors and planning, and by one of the authors of this paper in a limited-audience workshop.

The new contribution of this paper is in regards to constraints, i.e. statements of the form λ *if* p which relate fluents of the domain. In earlier formulations of causality p and λ were fluent formulas. In this paper we allow p to have temporal operators and thus refer to the past history. This necessitates a new definition of transition due to actions, which we provide, showing it is an extension of the definition of possible next states for the Markov case first introduced in [13], and later used for action language AC in [14] and B [7]. We also discuss how the language we propose can be used in modelling multimedia presentations. The non-Markovian nature of *Alan* results in a succinct representation of complex temporal relations. Examples of such representations are presented in [9]. Some alternative approaches are discussed in [18], though there is still some way to go towards an ontology of multimedia presentations.

Often it is noted that references to the history (using temporal operators) can be eliminated by recording the history as part of a state. Similarly, instead of using action occurrence as part of the p above, new fluents can be introduced that record the action occurrence in a state. That is, it is always possible for a Markovian model to replace a non-Markovian one with more fluents, and possibly even more actions. The major drawback of such an approach is that it leads to an increase in the number of fluents and hence in an increase in the total number of possible states, making processing more time-consuming. This is also a step backward from the perspective of knowledge representation and elaboration tolerance. In this regard note that the reason we use action languages in the first place (rather than a transition table listing state transitions due to actions) is to have a succinct representation. The motivation behind using constraints instead of compiling them (which will lead to additional effect propositions) is also similar. Our proposal of using temporal operators and action occurrences to refer to the history rather than adding new fluents is along the same lines.

The emphasis of this paper is on the semantics and formalization of the language, and is organized as follows. We first present the language Alan, with its three component languages: an action description language Alan$_A$, a language for describing observation Alan$_O$, and a query language Alan$_Q$. For each language we include its semantics. We then establish the relationship between theories of *Alan* and logic programs and briefly outline how this relationship can be used to reason about various properties of dynamic domains. For more extended examples of its application in modelling multimedia, the reader is referred to [9].

2 The Language Alan

We refer to our proposed language as *Alan* as a short form for (**A**ction **LA**nguage for **N**on-markovian domains). We assume a fixed action signature $\Sigma = \langle \text{F, A} \rangle$ for the language, consisting of two disjoint, nonempty sets of symbols: a set F of *fluents*, and a set A of *elementary actions*. As in all action languages, a fluent is a proposition whose truth value might vary from one state of the world to the next. A *fluent literal* is a fluent f or its negation $\neg f$. By a *compound action* we will mean a set $\{a_1, \ldots, a_n\}$ of elementary actions. Intuitively, execution of a compound action corresponds to the simultaneous execution of its components. Similar to other action languages *Alan* can be divided in three parts: an action description language *Alan*$_A$, a language for recording the agent's observations *Alan*$_O$, and a query language *Alan*$_Q$. In the following subsections we present each of these languages, together with their corresponding semantics.

2.1 Alan$_A$: an action description Language

We start with describing the syntax of Alan$_A$.

Syntax of Alan$_A$.
We define two kinds of formulas over Σ: "state formulas" and "temporal formulas". We refer to state and temporal formulas simply as *formulas* when their kind is clear from context or something applies to both.

Definition 1 ***(State Formula)*** *A state formula is an expression defined as follows:*

1. *A fluent literal is a state formula.*
2. *A statement of the form "occurs a_e", where a_e is an elementary action from Σ, is a state formula.*
3. *If p is a state formula, then $\neg p$, is a state formula.*
4. *If p_1 and p_2 are state formulas, then $p_1 \wedge p_2$ and $p_1 \vee p_2$ are state formulas.*
5. *Nothing else is a state formula.* □

Definition 2 ***(Temporal Formula)*** *A temporal formula in Alan is defined as follows:*

1. *A state formula is a temporal formula.*
2. *If p_1 and p_2 are temporal formulas, then*
   - ***lasttime*** p_1,
   - ***previously*** p_1
   - p_1 ***before*** p_2
   - p_1 ***since*** p_2

   are temporal formulas (sometimes referred to as temporal atoms).
3. *If p is a temporal formula, then $\neg p$, is a temporal formula.*
4. *If p_1 and p_2 are temporal formulas, then $p_1 \wedge p_2$ and $p_1 \vee p_2$ are temporal formulas.*
5. *Nothing else is a temporal formula.* □

The temporal connectives specified above [16] can be used[1] to express other useful temporal notions. For example, a statement **always** p, which says that p has always been true in the past, can be written as $\neg$**previously** $\neg p$. The statement **never** p (p has never been true in the past) can be written as $\neg$**previously** p or **always** $\neg p$.

Action descriptions of *Alan* consist of propositions, sometimes referred to as *causal laws*, which are defined as follows:

Definition 3 ***(Proposition)*** *If a is an action, a_e is an elementary action, λ is a fluent literal, and p is a formula, in the language Alan$_A$:*

1. *a* ***constraint*** *is an expression*
   λ ***if*** p
2. *a* ***causal proposition*** *is an expression*
   a_e ***causes*** λ ***if*** p
3. *an* ***impossibility proposition*** *is an expression*
   impossible a ***if*** p
4. *a* ***definition proposition*** *is an expression*
   defined λ ***if*** p

and a proposition is any expression of the form (1) through (4) above. □

In the above definition, p is said to occur in the "body" of the expressions, and λ and a occur in the "head"; p's are often referred to as *preconditions* of causal laws.

Definition 4 ***(Action Description)*** *A collection of propositions is called an action description.* □

Intuitively, to model a multimedia presentation, the possible behaviors of the display elements are encoded as an action description in *Alan*. These includes the behavior of the video and audio players, and others such as buttons or links. Particularities of the display system (buttons or other GUI elements) are represented as objects and are used as parameters for actions and fluents in predicates.

Semantics of Alan$_A$.
In the theories of actions an action description α serves as a formal model capturing aspects of reality relevant to the agent. Its main goal is to concisely define a collection of acceptable paths or "possible trajectories" of the agent's domain. In Markovian models such paths are characterized by transition diagrams in which possible states of the domain after the execution of action a depend only on a and the current state of the system. In contrast, to determine to which states the domain can move after the execution of a in a non-Markovian model the current state is not enough. One needs to know the domain's previous history. To describe the set of possible trajectories of α we will need some auxiliary definitions.

Let α be an action description of *Alan*. A set X of fluent literals from Σ is called *complete* if for any fluent f in Σ, $f \in X$ or $\neg f \in X$; X is called *consistent* if there is no fluent f such that $f, \neg f \in X$. A *partial state* of α is a consistent set of fluent literals of Σ. Partial states will be denoted by consecutively indexed letters s. If s is complete and consistent, it is called a *complete state* or simply, a *state*. A *path* π of α is a sequence $\langle s_0, a_0, \ldots, a_{n-1}, s_n \rangle$, where a's are (possibly compound) actions and $s_1, \ldots, s_{n-1}$ are states and s_n is a partial state. When necessary, n will be referred to as the *length* of π. By π_j we denote the prefix of π that ends at s_j. π_0 is equal to s_0, and it is called the *initial state* of the path. π is an *incomplete path* if s_n is not a complete state; otherwise we refer to it as a complete path. Given a fluent literal λ, by $\bar{\lambda}$ we denote $\neg f$ if $\lambda = f$, or f if $\lambda = \neg f$.

Now we will define the truth of formulas (F) with respect to paths (π).

Definition 5 ***(Truth of formulas)*** *Given a complete path π_n of length n, an action a, and state formulas p, q:*

1. *A fluent literal λ is true in π_n iff $\lambda \in s_n$.*
2. ***'occurs*** *a' is true in π_n iff $a \in a_{n-1}$.*
3. ***p*** $\wedge$ ***q*** *is true in π_n iff both p and q are true in π_n.*
4. ***p*** $\vee$ ***q*** *is true in π_n iff either p or q are true in π_n.*
5. *$\neg p$ is true in π_n iff p is not true in π_n*
6. ***lasttime p*** *is true in π_n iff $n > 0$ and p is true in π_{n-1}.*
7. ***previously p*** *is true in π_n iff $n > 0$ and for some $0 \leq i < n$, p is true in π_i.*
8. ***p before q*** *is true in π_n iff for some $0 \leq j \leq n$, p is true in π_j, and for every $i \leq j$, q is false in π_i*
9. ***p since q****, is true in π_n iff for some $0 \leq j \leq n$, q is true in π_j, and for every $j \leq i \leq n$, p is true in π_i* □

Our next goal is to define truth of formulas for incomplete paths, but first we need some preliminary notions. A path $\pi = \langle s_0, a_0, \ldots, a_{n-1}, s_n \rangle$ is said to be closed under the constraints and definition propositions of an action description α if:

1. for every constraint (λ **if** p) from α, if p is true in π then $\lambda \in s_n$, and
2. for every definition proposition (**defined** λ **if** p) from α, if p is true in π then $\lambda \in s_n$; if $\neg p$ is true in π then $\bar{\lambda} \in s_n$.

We now say that a complete path $\pi = \langle s_0, a_0, \ldots, a_{n-1}, s_n \rangle$ is an *extension* of an incomplete path
$\pi' = \langle s_0, a_0, \ldots, s_{n-1}, a_{n-1}, s'_n \rangle$
if $s'_n \subseteq s_n$, s_n is a complete state and π is closed under the constraints and definition propositions.

Definition 6 ***(Truth of formulas for incomplete paths)*** *A formula p is true in a (possibly incomplete) path π_n if it is true in every extension of π_n; and p is false in π_n if it is false in every extension of π_n.* □

Using the above definitions we can now say that a formula F is unknown in an incomplete path π if neither F nor $\neg F$ is true in π. We now characterize the collection of possible trajectories defined by an action description α.

Definition 7 ***(Executable actions)*** *Let α be an action description over signature Σ, a be an action of α, and π be a path over Σ. We say that a is executable after π if α contains*

[1] This is true only when we have a history of complete states. Otherwise, they may need to be written independently. We will explore this further in the sequel.

*no proposition (**impossible** a_e **if** p) such that $a_e \subseteq a$ and p is true in π.* □

We now define the effects of an action, also with respect to a path.

Definition 8 ***(E(a, π), direct effects of an action)*** *Let α be an action description, π be a path over α's signature Σ, and a be an action executable after π. The* direct effects *of a on π, denoted by $E(a, \pi)$, is the set of all fluent literals λ from Σ such that α contains an effect proposition (a_e **causes** λ **if** p) where*

1. *$a_e \in a$;*
2. *p is true in π.* □

The next important definition adapts a simpler definition from [13] to the non-Markovian case.

Definition 9 ***(Closure $C_\alpha(\pi)$).*** *Consider an action description α over signature Σ and a path $\pi = \langle s_0, a_0, \ldots, a_{n-1}, v\rangle$ where s_i's are states, v is a partial state, and a's are actions of α. The closure of π with respect to α, denoted by $\boldsymbol{C}_\alpha(\pi)$, is the path $\pi' = \langle s_0, a_1, s_1, \ldots a_{n-1}, w\rangle$ such that w is the smallest set of fluent literals from Σ satisfying the following conditions:*

1. *$v \subseteq w$,*
2. *for every constraint (λ **if** p) from α, if p is true in π' then $\lambda \in w$,*
3. *for every definition proposition (**defined** λ **if** p) from α, if p is true in π then $\lambda \in w$; if $\neg p$ is true in π then $\bar{\lambda} \in w$.* □

We are now ready to define the semantics of action description α of *Alan*, i.e. the set of possible trajectories of α.

Definition 10 ***(Possible Trajectory)*** *Let α be an action description of* Alan*.*

1. *For a complete state s, $\langle s\rangle$ is a possible trajectory of α if $\langle s\rangle = C_\alpha(\langle s\rangle)$.*
2. *If π_{n-1} is a possible trajectory of α then a path $\pi_n = \langle \pi_{n-1}, a, s_n\rangle$ is a possible trajectory of α whenever:*
   (a) a is executable after π_{n-1},
   (b) s_n is complete,
   (c) $\pi_n = C_\alpha(\langle \pi_{n-1}, a, (s_{n-1} \cap s_n) \cup E(\alpha, \pi_{n-1})\rangle)$ □

Note that because of the non-Markovian nature of *Alan*, executability of an action in a non-Markovian system can only be determined in reference to a path, rather than to a state, as is done in traditional action languages. Thus we can only speak about an executable sequence of actions having a possible trajectory from a *particular* state in a particular history, and not from every state, since even determining whether the sequence is executable or not depends on that history. The same applies to the direct effects of an action since the conditions upon which the direct effects of the action are determined (the body of propositions) can only be evaluated over a path.

The following example shows how we can model one part of the behavior of the multimedia presentation described in the introduction. It illustrates the non-Markovian features of *Alan*. More extensive examples were included in [9].

Example 1 *A multimedia presentation shows a question and gives the user two chances to answer it. If the wrong answer is entered twice, the portion of the presentation that showed the relevant content is shown again. Otherwise, the presentation ends.*

We let *Click*(submit) be the action of clicking on the "submit" button. The fluent CLOCK(X, t) indicates the presentation X is at point t in time (the value of t would be 0 at the start of the presentation X), and DISPLAY_PERIOD(X, p) that X will be displayed for a period p. Two other fluents, IS_SELECTED(y) and IS_CORRECT(y) indicate that a choice y is selected and is considered a correct selection, respectively.

The propositions used to describe the above conditions include:

Click(submit) **causes** CLOCK(main, $replay_start$),
DISPLAY_PERIOD(main, $replay_period$) **if**
IS_SELECTED(ans), ¬IS_CORRECT(ans),
previously 'occurs *Click*(submit)'

Click(submit) **causes 'occurs** *End*(main)' **if**
IS_SELECTED(ans), IS_CORRECT(ans)

Some details have been simplified for this example to emphasize the temporal aspect over implementation details, like the fact that there is only one question in one presentation of an animal. However, the formalization can easily be extended to account for more than one question and the fact that the presentation of an animal in the tutorial is followed by the presentation of another animal.

Our proposed model, and in particular the definition of possible next states using histories, is an extension of the definition of possible next states introduced in [13] and later used in action languages AC [14], and B [7]. We state this formally in the following proposition, where Res_C(a,s) refers to the set of states to which the Markovian system can move after the execution of a in state s.

Proposition 1. Let α be an action description consisting of constraints, causal propositions, impossibility propositions, and definition propositions such that all actions in those propositions are elementary actions, and the body of the expressions (p's in Definition 4) are only fluent formulas. Let s be a state and a be an action. Then $s' \in Res_C(a, s)$ iff $\langle s, a, s'\rangle$ is a possible trajectory of α. □

Intuitively, Proposition 1 means that by restricting the use of causal laws of *Alan* to the syntax of [13], we have a formulation of causality equivalent to that in [13]. Thus Definition 10 is an extension of the definition of Res_C(a,s) presented in [13]. The full proof of this appears in the principal author's Doctoral dissertation [8]. The fix-point definition of Res_C(a,s) can be found in [13].

2.2 Alan$_O$: the Observation Language

In addition to knowing the set of possible trajectories of the domain the agent may need to record the domain history up to a given point n. We will follow [2] and limit such recording to statements of the form

1. **initially** λ
2. a_e **occurs_at** t
3. λ **observed_at** t

frequently referred to as *axioms* or *observations*. As usual we use λ to denote a fluent literal, a_e denotes an elementary action, and t is a non-negative integer from $[0..n]$. Axioms of type 1 state what is true in the initial state. Those of type 2 say that "action a_e occurred at time point t". Axioms of type 3 state that "the fluent literal λ was observed to be true at t". The set of axioms is often referred to as the *recorded history* of the domain. Given a set of axioms S, the current time point denoted by t_c is the maximum element of the set $\{t+1 : a \textbf{ occurs_at } t \in S\} \cup \{t : \lambda \textbf{ observed_at } t \in S\}$.

Definition 11 ***(Domain Description)*** *A domain description D of Alan is a tuple $\langle \alpha, \Gamma \rangle$ where α is an action description and Γ is its recorded history.* □

Domain descriptions in *Alan* are used in conjunction with the following informal assumptions:

1. changes in the values of fluents can only be caused by execution of actions;
2. there are no action occurrences other than those observed; and
3. there are no direct effects of actions except those specified by the effect propositions of the domain.

Intuitively domain description D limits possible trajectories of α to those which satisfy axioms from Γ. More precisely,

Definition 12 ***(Model)*** *We say that a possible trajectory $\pi =< s_0, a_0, s_1, \ldots, a_{n-1}, s_n >$ of α is a model of a domain description $D = \langle \alpha, \Gamma \rangle$ if for any $0 < k < n$*

1. $a_k = \{a : (a \textbf{ occurs_at } k) \in \Gamma\}$
2. *if* ***initially*** $\lambda \in \Gamma$ *then* $\lambda \in s_0$.
3. *if* λ ***observed_at*** $k \in \Gamma$ *then* $\lambda \in s_k$. □

2.3 Alan$_Q$: the Query Language

One of the main purposes of a representation such as the one given in a domain description is to be able to extract answers to queries about a particular domain. We now define queries and their interpretation in Alan$_Q$.

Definition 13 ***(Query)*** *A query in Alan$_Q$ is an expression of one of the following types:*

1. p ***holds_at*** t
2. p ***holds_in*** $[t_1, t_k]$
3. p ***after*** A ***at*** t

where p is a formula, A is a sequence of actions, t_1 is a time point, and t and t_k are either time points or t_c denoting the current time. □

Queries of type 1 ask whether the given formula holds at a time point. Queries of type 2 ask whether the given formula holds over a period of time between time points t_1 and t_k. Queries of type 3 can be read as "if the sequence A of user actions were executed at time point t, would p hold afterward?". Simple queries are used to inquire about general properties of the domain, and to verify its validity. If the formula p includes variables, the system will respond with specific ground values for that variable (if any) that will make the formula true in the given domain.

Definition 14 ***(Entailment)*** *Given a domain description D = $\langle \alpha, \Gamma \rangle$ and a model $\pi = \langle s_0, a_1, s_1 \ldots, s_n \rangle$ of D, π entails query Q (or Q is true in π) if:*

1. *Q = p* ***holds_at*** *t and p is true in π_t*
2. *Q = p* ***holds_in*** *$[t_1, t_k]$ and, for every π_j such that $t_1 \leq j \leq t_k$, p is true in π_j.*
3. *$Q = p$* ***after*** *$b_1, \ldots, b_j$* ***at*** *t, $\pi' =< s_0, a_0, \ldots, a_{t-1}, s_t, b_1, s_{t+1}, \ldots, b'_j, s'_{t+j} >$ is a possible trajectory of α, and p is true in π'.* □

3 Computing Models of Domain Descriptions

Various reasoning algorithms associated with a domain description D of *Alan* are based on our ability to compute the models of D not exceeding some given length N. In the theory of action languages this is frequently done by

(a) mapping a domain description D and integer $N \geq 0$ into a logic program $T(D, N)$ whose stable models (answer sets) [6] correspond to models of D; and

(b) using an answer set finder (e.g. smodels [17], dlv [4]) to compute the answer sets of $T(D, N)$.

The existing answer set finders are reasonably efficient and allow computation of answer sets for programs with hundreds of thousands ground rules. They were successfully used for various sizeable applications. Moreover the corresponding systems are improving at a very high rate which allows us to hope for a higher scalability. In this section we will outline a construction of $T(D, N)$ for a simple case of domain descriptions whose propositions have preconditions consisting of a single fluent literal or a temporal atom. In addition we only consider action descriptions not containing definition propositions. The restrictions are not essential and are caused by the space limitations. In what follows F will be used as a variable for fluents from signature Σ of D, P and Q will stand for formulas occurring in preconditions of its causal laws, and H will be a shorthand for $holds$. Propositions of *Alan* will be written as atoms of Prolog, e.g. L **if** P will have a form $if(L, P)$, etc. The first collection of rules of $T(D, N)$ correspond to D's causal laws.

Causal Laws

```
H(F,T+1) :- CAUSES(A,F,P), OCCURS(A,T), H(P,T).
H(F,T) :- IF(L,P), H(P,T).
:- IMPOSSIBLE(A,P), H(P,T), OCCURS(A,T).
```

Recall that the last rule is a constraint which guarantees that no answer set of the program will satisfy its premise. The rules are written for propositions in which F and P are atomic fluents. If they are negative literals the corresponding rules can be obtained by putting the negation symbol '¬' in front of the corresponding h, e.g.

$\neg$H$(F, T+1)$:- CAUSES$(A, \neg F, \neg P)$, OCCURS(A, T), $\neg$H(P, T).

The next two rules formalize the Inertia Axiom from [15].

Inertia

H(F,T+1) :- H(F,T), NOT $\neg$H(F,T).
$\neg$H(F,T+1):- $\neg$H(F,T), NOT H(F,T).

The above rules are similar to those used in translations of other action formalisms. The next group of rules defining the truth of temporal literals is new.

Truth of Temporal Literals

H(**lasttime** P, T) :- H(P,T-1).
H(**previously** P, T) :- $T_0 < T$, H(P, T_0).
H(P **before** Q, T) :- $T_0 < T$, H(P, T_0), H_BETWEEN$(\neg Q, 0, T_0)$.
H(P **since** Q,T) :- $T_0 < T$, H(Q, T_0), H_BETWEEN(P, T_0, T), H(P,T).

The new relation H_BETWEEN(L, T_1, T_2) says that literal L holds in the interval $[T_1, T_2)$. It is defined by the following rules:

H_BETWEEN$(L, T-1, T)$:- H$(L, T-1)$.
H_BETWEEN(L, T_1, T_2) :- H(L, T_1), H_BETWEEN$(L, T_1 + 1, T_2)$.

Finally, the rule

:- $\neg$H(F,T), OBSERVED(F,T).

guarantees that the agent's observations do not contradict its expectations.

The program $T(D, N)$ consists of the above rules and statements from α and Γ represented as Prolog atoms. (An actual program contains a few auxiliary axioms omitted in the presentation).

It is not difficult to see that for every answer set A of $T(D, N)$ the statements formed by the relations *occur* and h from A form a path of α of length N. We say that this path is *defined by* A. Computation of models of D are based on the following proposition.

Proposition 2.
Given an action description α if $\pi_{n-1} = \langle s_0, a_0, \ldots, a_{n-2}, s_{n-1}\rangle$ is a possible trajectory of α then
a path $\pi_n = \langle s_0, a_0, \ldots, a_{n-2}, s_{n-1}, a, s_n\rangle$ is a possible trajectory of α iff
there is an answer set A of the program $T(\alpha, n) \cup \{holds(f, j) \mid f \in s_j, 0 \leq j \leq n-1\} \cup \{occurs(a', j) \mid a' \in a_j, 0 \leq j \leq n-2\} \cup \{occurs(a', n-1) \mid a' \in a\}$ such that $s_n = \{l \mid holds(l, n) \in A\}$. □

The above Proposition can be used to design algorithms for performing a large number of reasoning tasks. First we can use a large number of results from the theory of logic programming as well as answer set finders to establish that the program $T(D, N)$ is consistent, i.e. has answer sets. (Such consistency is expected for all 'reasonable' action descriptions.) Suppose now we are given a domain description D whose recorded history uniquely describes the initial state of the domain, and a query Q of the form, say, p **h***olds_atk* (where $k \leq N$). To check if this query holds in all model of D we expand the program $T(D, N)$ by the constraint

:- Q.

and ask an answer set finder to find an answer set of the resulting program. If no such answer set exists then Q holds in all models of D. Simple additions to the program allow us to answer similar queries even if the initial situation is not completely described by Γ. Similar techniques can be used for performing substantially more complex reasoning tasks. For instance to find a plan of length K, which will allow a reasoner to achieve goal G in a number of steps limited by N one can expand the program by the rules:

GOAL :- H(G,T).
:- NOT GOAL.
OCCURS(A,T) :- $T_c < T < T_c + K$, not $\neg$occurs(A,T).
$\neg$OCCURS(A,T) :- $T_c < T < T_c + K$, not OCCURS(A,T).

Answer sets of the resulting program will correspond to possible plans for achieving G from the current moment T_c. (This is a corollary of Proposition 2.) Unlike previous answer set planners in the literature the planners of *Alan* are able to achieve goals of the form 'Move to a state in which fluent f is true but make sure that if g is true now then it should be true on your way to f.' Similar techniques can be used to combine planning and other tasks such as diagnostic reasoning.

4 Conclusion

We have presented a temporal action language named *Alan* that can be used to model non-Markovian systems. We defined the notion of closure of a trajectory, used it to define transition between trajectories, and showed that this formulation is an extension of traditional markovian definitions that define closure with respect to states and transition between states. *Alan* is used in the first author's Ph.D thesis to model multimedia displays where temporal preconditions are prevalent. We expect it to be also useful in modelling other dynamic environments, such as the Web, multimodal interaction, diagnostic systems, workflow systems, graphical user interfaces, monitoring systems, and active databases.

The third author was partially supported by NASA under contracts 1314-44-1476 and 1314-44-1769.

References

[1] F. Bacchus, C. Boutilier, and A. Grove. Structured solution methods for non-markovian decision processes. In *AAAI 97*, pages 112–117, 1997.

[2] C. Baral and M. Gelfond. Reasoning agents in dynamic domains. In J Minker, editor, *Logic Based AI*. Kluwer, 2000.

[3] C. Baral, M. Gelfond, and A. Provetti. Representing Actions: Laws, Observations and Hypothesis. *Journal of Logic Programming*, 31(1-3):201–243, May 1997.

[4] S. Citrigno, T. Eiter, W. Faber, G. Gottlob, C. Koch, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The dlv system: Model generator and application front ends. In *Proceedings of the 12th Workshop on Logic Programming*, pages 128–137, 1997.

[5] A. Gabaldon. Non-markovian control in the situation calculus. In *Proceedings of the Second International Workshop on Cognitive Robotics*, Berlin, Germany, 2000.

[6] M. Gelfond and V. Lifschitz. Representing actions and change by logic programs. *Journal of Logic Programming*, 17(2,3,4):301–323, 1993.

[7] M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3(6), 1998.

[8] Graciela Gonzalez. A Display Specification Language for Multimedia Databases, 2000. Ph. D. Thesis, Department of Computer Science, University of Texas at El Paso.

[9] Chitta Baral Graciela Gonzalez and Peter Cooper. Modeling multimedia displays using action based temporal logic. In *Visual and Multimedia Information Management, IFIP TC2/WG2.6 Sixth Working Conference on Visual Database Systems, May 29-31, 2002, Bisbane, Australia*, volume 216 of *IFIP Conference Proceedings*, pages 141–155. Kluwer, 2002.

[10] Jürgen Hauser. Realization of an extensible document model. in: Proceedings of the eurographics multimedia '99 workshop. Technical report.

[11] Na'el Hirzalla, Benjamin Falchuk, and Ahmed Karmouch. A temporal model for interactive multimedia scenarios. *IEEE MultiMedia*, 2(3):24–31, Fall 1995.

[12] Cherif Keramane and Andrzej Duda. Operator based composition of structured multimedia presentations. In *COST 237 Workshop*, pages 1–17, 1997.

[13] N. McCain and H. Turner. A causal theory of ramifications and qualifications. In C. Mellish, editor, *Proc. of IJCAI 95*, pages 1978–1984. Morgan Kaufmann, 1995.

[14] N. McCain and H. Turner. Causal theories of action and change. In *Proc. of AAAI*, pages 460–465, 1997.

[15] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[16] G. Mendez, J Llopis, J. Lobo, and C. Baral. Temporal logic and reasoning about actions. In *Common Sense 96*, 1996.

[17] I. Niemela and P. Simons. Smodels – an implementation of the stable model and well-founded semantics for normal logic programs. In J. Dix, U. Furbach, and A. Nerode, editors, *Proc. 4th international conference on Logic programming and non-monotonic reasoning*, pages 420–429. Springer, 1997.

[18] P. Pazandak and J. Srivastawa. The language components of damsel: An embedable eventdriven declarative multimedia specification language, 1995.

[19] E. Sandewall. Special issue. *Electronic Transactions on Artificial Intelligence*, 2(3-4):159–330, 1998. http://www.ep.liu.se/ej/etai/.

[20] Thomas Wahl and Kurt Rothermel. Representing time in multimedia systems. In *International Conference on Multimedia Computing and Systems*, pages 538–543, 1994.