Computer Science
University of Manchester

A Reified Temporal Logic For Nonlinear Planning

Ying Zhang, Howard Barringer

Technical Report UMCS-94-7-1

A Reified Temporal Logic For Nonlinear Planning *

Ying Zhang, Howard Barringer

Department of Computer Science
University of Manchester
Oxford Rd., Manchester, U.K.
yzhang@cs.man.ac.uk, howard@cs.man.ac.uk

July 1994

Abstract

Deductive planning treats planning as a reasoning process within a formal logic. Several formal logics have been applied in various deductive planning systems, but none is completely satisfactory and widely accepted by planning researchers. Almost all the formal logics applied in deductive planning are proposed for more general purpose than planning. Usually, the more general and the more expressive a language is, the less efficient that language is for dealing with problems in a specific domain. In this report, we propose a temporal logic specifically for representing and reasoning about nonlinear planning. Our temporal logic naturally captures the persistence assumption and, when applied in deductive planning, can deal with the frame problem efficiently. Also, in this temporal logic, planning problem can be concisely and clearly specified. Directly upon this temporal logic, we build a deductive nonlinear planning framework, constraint logic planning, which is proved to be sound, complete and is showed to be efficient. In this report, we briefly describe how the temporal logic provides the basis for constraint logic planning.

Recent technical reports issued by the Department of Computer Science, Manchester University, are available by anonymous ftp from `ftp.cs.man.ac.uk` in the directory `pub/TR`. The files are stored as PostScript, in compressed form, with the report number as filename. They can also be obtained on WWW via URL `http://www.cs.man.ac.uk/csonly/cstechrep/index.html`. Alternatively, all reports are available by post from The Computer Library, Department of Computer Science, The University, Oxford Road, Manchester M13 9PL, UK.

1 Introduction

Deductive planning is a reasoning process within a formal logical system. In such a system, a planning problem is specified by a set of well-formed logical formulas. Generally, the initial conditions, the domain constraints and the goals are described by logical formulas. The primitive actions are described as planning rules and also represented by logical formulas. To solve a given planning problem is to prove constructively, by means of some automatic or interactive theorem proving system, that the formula describing the goals is a logical consequence of the planning rules and the initial conditions.

Deductive planning provides a rigorous planning approach such that the capabilities and limitations of planning systems can be clearly defined. In particular, in a deductive planning framework, a planning problem to be solved can be examined independently from the procedural details of finding a plan in the same way as solving a problem in a declarative programming language. In addition, deductive planning systems are more expressive than classical planning systems because of their use of formal logics (Allen, 1990).

Most of the deductive planning systems are state-based. In these systems, to describe the fact that most of the properties that hold true in a state will remain true after an execution of an action requires an enormous number of axioms. This problem is called *frame problem.* In addition to the frame problem, most deductive planning systems which use various kinds of formal logics depend on *linearity.* In other words, these deductive planning systems are linear planners. In these frameworks, plans can only be represented as sequences of actions (sequential composition of actions). Deductive planning systems based on various situation logics (Green, 1969; Waldinger, 1977; Kowalski, 1979; Manna & Waldinger, 1980; Manna & Waldinger, 1986) and dynamic logics (Rosenschein, 1981; Kautz, 1982) fall into this category. Linear planning is believed to be exponentially less efficient than nonlinear planning (Chapman, 1987) and has rarely been adopted since NOAH (Sacerdoti, 1975). Furthermore, because of this linearity restriction, concurrent actions and interval actions cannot be represented and reasoned about.

Allen's work (Allen & Koomen, 1983; Allen, 1991a; Allen, 1991b) is an exception. In addition to the ability to represent and reason about interval, concurrent and overlapping actions, the most significant advantage of Allen's planner over the state-based deductive planners, such as those mentioned above, is that it captures many important aspects of nonlinear planning systems. In Allen's temporal logic, the fact that two given actions are disjoint in time can be simply expressed as a formal constraint. Therefore, any decision on the ordering between two actions in a plan can be deferred for as long as possible.

Allen's planner is built on an interval-based temporal logic. Vilian and Kautz (86) showed that computing the consequences of temporal assertions is computationally intractable in the interval-based representation. To avoid this NP-hard execution time problem, the completeness of Allen's temporal reasoner, which is the heart of Allen's planner, has to be sacrificed (Allen, 1991b), i.e. some inconsistencies cannot be discovered by the temporal reasoner. As a result, soundness and completeness of the planner cannot be guaranteed.

As indicated, several formal logics have been applied to deductive planning, but none is completely satisfactory and widely accepted by planning researchers. Such logics were proposed for a more general purpose than planning. Usually, the more general and the more expressive a language is, the less efficient that language is for dealing with problems in a specific domain. In regard to a formal language for planning, we can have either a language that is powerful enough for most problems in AI and is less efficient for planning or a language that is efficient for planning and is not powerful enough to cover all the domains in AI. It is the second choice which we opt for in this report.

In this report, we first discuss the basic considerations of a formal language for nonlinear planning. Then we present a temporal logic as such a language for nonlinear planning. Finally, we briefly describe how the temporal logic is used to provide the basis for deductive nonlinear planning.

2 Basic Considerations of a Formal Language for Nonlinear Planning

2.1 Temporal Logic as the Presentation Language

The basic proposition types required for planning are actions and properties. The dominant view of actions in planning is that actions are state transformations, i.e. functions from one state to another. The truth values of properties are asserted over states. This view of actions underlies all the state-based classical planning systems (STRIPS (Fikes & Nilsson, 1971) and its successors) and state-based deductive planning systems. As a function from one state to another, an action thus uniquely defines a resulting state. Therefore, simultaneous and overlapping actions are difficult to express in these systems. Also, the execution time of an action is meaningless under this view and actions are considered to be instantaneous. To deal with these problems, some of planning approaches have involved the ability to reason about time (Allen & Koomen, 1983; Allen, 1991a; Allen, 1991b; McDermott, 1982; Vere, 1983; Vere, 1985a; Vere, 1985b; Miller *et al.*, 1985; Dean, 1985; Tate, 1986; Currie & Tate, 1989; Tsang, 1987; Cranefield, 1990; Biundo *et al.*, 1992).

To move away from the "toy world" domains that characterize much of the earlier work, a planning representation language should be able to characterize phenomena in the real world such as:

1. more than one action can be executed at the same time;
2. there exists an environment influence beyond the direct control of the agent; and
3. actions may take time.

To meet these requirements, the representation language should have an ability to reason about time. For deductive planning, the representation language should be a temporal logic of some form.

For a temporal logic, there are three ways to associate a proposition type with time.

Firstly, time can be simply added as an extra pair of arguments to a proposition type to denote the assertion of the proposition type over the time interval. For example, $On(t_1, t_2, A, B)$ can be used to express that the block A is on the top of block B over the time interval (t_1, t_2). Apart from in earlier works, this "naive first order treatment of time" (Reichgelt, 1987) is adopted neither in AI nor in theoretical computer science. Shoham (87) pointed out that this method "accords no special status of time, neither conceptual nor notational". It is this problem which causes part of the frame problem

(Kowalski, 1979). With time listed as a pair of the ordinary arguments, there is nothing in general one can say about the temporal aspect of assertions. At most, one can describe temporal aspect about a specific predicate. This leads to a large number of frame axioms in planning.

The second method to associate an assertion with time is that adopted in modal temporal logics. In this method, the time is associated with a proposition type in an interpretation. In classical logic, a formula φ may be true wrt an interpretation M (written $M \models \varphi$). In a modal temporal logic, a formula φ may be true wrt an interpretation M over a time interval I (written $\langle M, I \rangle \models \varphi$). For example, the fact that the proposition type $On(A, B)$ holds true in a interpretation M over the time interval (t_i, t_j) can be expressed are $\langle M, (t_i, t_j) \rangle \models On(A, B)$. Modal operators can be used to implicitly refer to the times accessible to the current one over which the formula is to be interpreted. The standard modal operators used in point modal temporal logics are "always" ($\Box$) which refers to all the accessible times, "sometime" ($\Diamond$) which refers to some accessible times and "next" ($\bigcirc$) which refers to the next accessible time. In the interval modal temporal logic given in [Halpern and Shoham 86] and [Shoham 87], there are twelve basic modal operators. For example, the modal operator $\langle A \rangle$ refers to some intervals beginning immediately after the end of the current one. The modal operator $[B]$ refers to all the intervals during the current one, beginning when the current one begins.

This method is more expressively compact and natural and it is popular in theoretical computer science. However, there are two main reasons that we do not intend to adopt this logical form. First, in general, theorem proving methods for modal temporal logic are less efficient than that for classical logics. Therefore, some researchers in modal theorem proving prefer to translate modal logics first into some non-modal language, and then to adopt a non-modal theorem proving method, for example various resolution methods (Fariñas del Cerro, 1982; Fariñas del Cerro & Herzig, 1988; Fariñas del Cerro & Herzig, 1989; Fariñas del Cerro & Herzig, 1990; Ohlbach, 1988), the matrix method (Wallen, 1987) or the constraint logic approach (Frisch & Scherl, 1990). The second reason is that modal temporal logics are less expressive than reified temporal logics (Cranefield, 1990), which we will discuss next. For any modal temporal logic, there exists a reified temporal logic which can be used as its meta-language.

The third way to associate an assertion with time is to take both the proposition type and time as arguments of some "global predicates", say "$True$" or "$Hold$", for example, $True(t_i, t_j, On(A, B))$. This treatment of time is widely accepted in AI. Logics of (Allen & Hayes, 1985; McDermott, 1982; Shoham, 1987) are reified temporal logics. Reichgelt (87) pointed out that a reified temporal logic can be viewed as a meta-language, which is used to reason about the object level formulae of interest. The semantics of any modal temporal language can be encoded by a reified temporal language. In this way, the natural and compact expressiveness of modal temporal logics can be combined with the computational advantage of first order logic (Cranefield, 1990). Kowalski (79) pointed out that the ability of reasoning about object level formulae avoids part of frame problem. In our case, we need this ability to deal with the frame problem and we choose a reified temporal logic as the language for planning.

2.2 Interpretation of Assertions over Maximal Intervals

In all formal representations of planning, assertions of properties or actions are interpreted implicitly or explicitly over time. For each assertion, there is a non-temporal component (such as "the block A is on the top of block B") and a temporal component (time, world state or situation). Following McDermott's and Shoham's practice, we term the non-temporal component of an assertion a proposition type. In planning, assertions of proposition types can be interpreted over either time points, time intervals or both.

In situation calculus (McCarthy & Hayes, 1969) and some other formal languages for planning (Manna & Waldinger, 1986; Rosenschein, 1981), assertions of proposition types are interpreted over situations. A situation is a complete snapshot of the world at some instant in time, and, therefore, the assertions of the proposition types are implicitly interpreted at time points. In the planning representation language given by Christensen et al. (Christensen & Grove, 1991; Christensen, 1990), assertions of the proposition types are interpreted over *plan steps*. A plan step can be understood as an execution of an action. There is no way that a plan step can be connected with a time duration. Therefore, assertions of actions and properties are also interpreted at time points. The assertion interpretations in Cranefield's plan specification logic (Cranefield, 1990) and in Pednault's work (Pednault, 1991) are similar to that of Christensen.

In Allen's formalism, assertions over time points are actually ruled out. Assertions of proposition types (properties, events and processes) are interpreted exclusively over time intervals. In McDermott's temporal logic, assertions of *facts* are interpreted over time points. Unlike in situation calculus and dynamic logic, the assertion of a fact P over a time interval (t_1, t_2), $TT(t_1, t_2, P)$, is available as an abbreviation:

$$TT(t_1, t_2, P) \equiv \forall t(t_1 \leq t \leq t_2 \Longrightarrow T(t, P))$$

Actions in MeDermott's temporal logic are of specific *event* types. Unlike fact types, event types are exclusively interpreted over time intervals (Occur(t1,t2, E)).

In theoretical computer science and in AI, both kinds of assertion interpretation can be found. For a detailed discussion of this issue, see (Shoham, 1987).

In a temporal logic for classical planning, the proposition types to be asserted over time are actions and properties. We discuss the assertions for the two types separately.

2.2.1 Assertion for Actions

In the earlier planning systems, actions are assumed to be instantaneous and atomic. In all the formal models in which assertions are interpreted over time points either implicitly or explicitly, actions are restricted to take no time. In the real world, however, most actions take time and several actions can be executed simultaneously or may overlap. To reason about the real world, some of the planning systems focus on reasoning about actions which take time. In (Allen & Koomen, 1983; McDermott, 1982; Allen, 1991a) actions (events) are asserted exclusively over time intervals. In (Vere, 1983), actions are intervals with numerical endpoints.

Unlike a property type, the assertion of an action over a time interval cannot be expressed as an abbreviation of some formula that consists of assertions over time points. Actions are specific events and they correspond to both the "gestalt type" and "solid type" in (Shoham, 1987). As defined by Shoham, a proposition type is *gestalt* if it never holds over two time intervals one of which properly contains the other. A proposition type is *solid* if it never holds over two properly overlapping time intervals. For a proposition type that has these two properties, if it is asserted over a time interval, then the interval is both the maximal and the smallest possible for it.

To reason about real world problems, we need to interpret the assertions of actions directly over time intervals. In some cases, if we really want to reason about instantaneous actions or if the execution time of some actions is not important, the assertions of such actions may be interpreted over point time intervals (e.g., zero length intervals).

2.2.2 Assertion for Properties

Planning using situation calculus and other formal languages in which assertions of properties are interpreted over time points runs into the frame problem. The frame problem is essentially related to the assertion of properties. McDermott pointed out that the frame problem is even more "acute" in a temporal logic because anything could be happening in any time interval. In situation calculus, it is possible to assume that a situation will persist until the next action. In an interval temporal logic, however, there is no term to refer to the world when the next action occurs. The frame problem "becomes the problem of inferring what is true at the end of an arbitrary interval" (McDermott, 1982).

The most widely accepted approach to handle the frame problem in planning is to adopt the *STRIPS assumption*. In STRIPS representation, operators are structures consisting of a precondition list, a delete list and an add list. An operator is applicable in a state if all the properties on its precondition list hold true in that state. STRIPS assumes that one of the results of applying an operator to a state is that all the properties which hold true in that state and are not on the delete list of the operator will remain true in the resulting state. This is commonly known as the STRIPS assumption and virtually all subsequent planners make use of this assumption in some form or other.

McDermott and Allen generalized the STRIPS assumption to *the persistence assumption* in reified interval temporal logics. The persistence assumption states that, if a property is asserted to be true over a time interval, then the time interval is the maximal interval for the property. To quote McDermott (82)

> "Part of my expectation in developing a robust logic of time was that we could reason about facts "from the side" inferring that they were true for whole stretches of time. It's no loss that we can not work our way from one state to the "next" any more, that was always a bad idea."

In Allen's planning system, the persistence assumption is simply expressed as a general proposition constraint (Allen & Koomen, 1983):

> "Two intervals associated with the same proposition are equal or one is strictly before the other; in other words, if P holds over T_{p1} and T_{p2}, then $T_{p1}(<>=)T_{p2}$."

The formula $T_{p1}(<>=)T_{p2}$ expresses that the interval T_{p1} is before T_{p2} or is after T_{p2} or is equal to T_{p2}.

In our case, the language that we use is proposed solely for planning. Instead of adopting the persistence assumption as a constraint as in (Allen & Koomen, 1983), we prefer to interpret the assertion of a property directly over the maximal time intervals.

In other words, we prefer to interpret the assertions of properties over series of *disjoint time intervals*. Such kind of assertion interpretation for properties is consistent with that for actions. Executing an action over an interval implies that the interval is the maximal interval as well as the smallest possible interval for the execution.

In planning, if a property holds over a time interval, it holds over all of its sub-intervals, possibly excluding its end points, or at all the internal time points of the interval. And if a property holds true over all the proper sub-intervals or all the internal time points of an interval, it holds true over the interval itself. Shoham termed this kind of proposition type a *liquid proposition type*. According to this property, interpreting the assertions of properties over the biggest possible time intervals does not prevent us from referring the assertions over arbitrary time intervals or over individual time points. For example, if we use the expression $Hold(P, t_k, t_l)$ to denote an assertion of a property P over the maximal time interval (t_k, t_l), then the assertion of a property P over an arbitrary interval (t_i, t_j) can be expressed as:

$$\exists t'_i, t'_j . t'_i \preccurlyeq t_i \preccurlyeq t_j \preccurlyeq t'_j \wedge Hold(P, t'_i, t'_j)$$

If we choose time intervals as primitive, then a property P's holding over an arbitrary interval I can be expressed as:

$$\exists I'. In(I, I') \wedge Hold(P, I')$$

where $In(I, I')$ expresses that I is a subinterval of I'

2.3 Time Points as the Primitive Temporal Objects

We have decided to interpret the assertions of the properties and actions over the biggest possible time intervals. There are two approaches to refer to a time interval in temporal logics. The first approach uses intervals as primitive temporal objects and the other approach uses time points as primitive objects. Most interval temporal logics in theoretical computer science use time points as primitives (Halphern *et al.*, 1983; Moszkowski, 1986). However, in AI, both approaches can be found. For example, in (McDermott, 1982; Shoham, 1987), time points are temporal primitives whereas in (Allen, 1984; Allen & Hayes, 1985; Halpern & Shoham, 1986) time intervals are the primitives.

Using time points as primitive objects, any time interval can be referred to by its ends points. If the two end-points of an interval are the same, then the interval represents a time point (or point interval). There is only one relation, *precedence*, between any two time points.

Using time intervals as primitive objects, an interval can be referred to directly by its name. In Allen's temporal logic, there are thirteen possible relations between two time intervals such as "Meet", "Before", "Start" etc. and these relations can all be defined by one primitive relation "Meet". In the temporal logics in which time intervals are primitives, time points can be defined as derived time objects out of time intervals (Allen & Hayes, 1985; Lin, 1991).

Time points are a mathematical abstraction and they have no manifestation in the real world. They can be constructed from the objects that exist in the real world with which we are familiar, for example events or periods (Shoham, 1987; Lin, 1991). However, as Shoham pointed out, point-based formulas are more concise and more intuitive. Furthermore, in planning, considering implementation, we believe that to deal with one *precedence* relation among time points is easier than to deal with thirteen temporal relations among time intervals. Vilian and Kautz (Vilain & Kautz, 1986) showed that computing the consequences of temporal assertions is computationally intractable in the interval-vased representation. To avoid this NP-hard execution time problem, we will therefore opt for taking time points as primitives.

2.4 Partial Order Time Structure for Nonlinear Planning

There exist different views about time structures. For example, time may be viewed as linear or as branching. Time may branch only in the future or it may branch both in the future and in the past, and time may or may not be allowed to meet again after branching. In AI most formalisms assume linear time. In Allen's interval temporal logic, time is linear and any two time intervals are exclusively related by one of the thirteen temporal relations. In fact, for any two derived time points in this logic, either one precedes the other or they are equal (Theorem 8 in (Allen & Hayes, 1985)). In McDermott's point-based temporal logic, time branches only into the future. McDermott (82) stated that the future is really indeterminate. "The future may be unknown, but there is only one past". In McDermott's temporal model, two branches that diverge into the future are not allowed to meet in the more distant future. This is expressed as an axiom (Axiom 7):

$$\begin{aligned}&(if(and(=<?s1\ S)(=<?s2\ S))\\&\quad(or(=<\ s1\ s2)(=<\ s2\ s1)))\end{aligned}$$

This axiom states that, given a state S, for any two state $?s_1$ and $?s_2$, if they precede S, then either $?s_1$ precedes $?s_2$ or vice versa. In classical planning, there is a unique goal state (or goal time point) in which the given goals hold true. Every state which is involved in the planning must precede the goal state. According to the above axiom, the temporal relation among the time points which are involved in planning is a total order relation. For applications in classical planning, McDermott's future branching temporal model collapses into a linear model.

What we are proposing here is not a strict temporal logic, but a language for nonlinear planning (there already have existed many formal languages suitable for linear planning). We need not restrict ourselves to all the philosophical considerations of time. We opt for a partial order structure of time to model nonlinear planning and nonlinear plans. Of course, in such a partial order structure, time should be allowed to meet again after branching.

We have explained why we interpret assertions over the maximal intervals. However, in a temporal logic with partial order time structure, to capture the persistence assumption, we have to interpret assertions of proposition types over *the necessarily maximal* intervals. That is, if a proposition type is asserted true over an interval in an interpretation, then the interval is the maximal for the property in all *completions* of the interpretation. If an interpretation $\mathcal{M}$ is the tuple $\langle \mathcal{W}, \mathcal{R}, \mathcal{D}, \pi_F, \pi_P \rangle$, then $\mathcal{M}' = \langle \mathcal{W}, \mathcal{C}(\mathcal{R}), \mathcal{D}, \pi_F, \pi_P \rangle$ is a *completion* of $\mathcal{M}$, where $\mathcal{C}(\mathcal{R})$ is a total order relation over the time points in $\mathcal{W}$ such that $\mathcal{R} \subseteq \mathcal{C}(\mathcal{R})$. The requirement that assertions are interpreted over the necessarily maximal intervals is achieved in our temporal logic by restricting interpretations that, for any two intervals over which a proposition type is asserted, either these two intervals are the same or one is strictly before the other. As the result of this constraint, in our

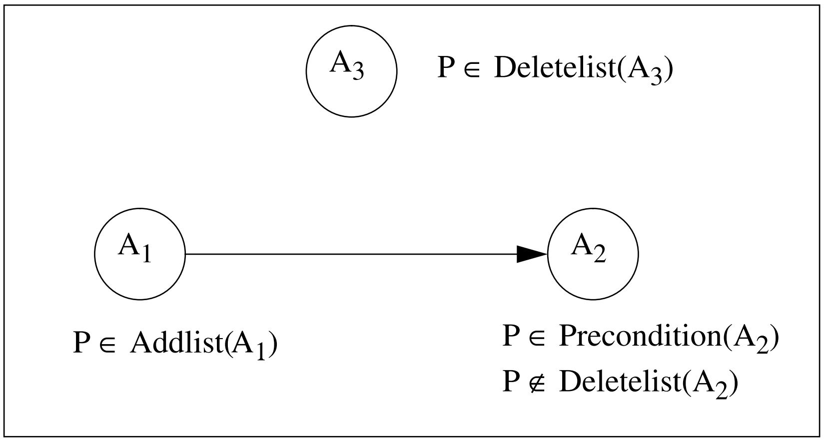


Figure 1: A nonlinear plan with deleted-condition conflict.

temporal logic, time intervals for the same properties or actions collapse into linear order, whereas time intervals of different properties or actions remain in partial order.

2.5 Assertion for Necessary Truth

Figure 1 gives a possible nonlinear plan. For simplicity, in this plan, we assume that all A_1, A_2 and A_3 are point actions. In this example, action A_1 asserts the property P and P is one of the preconditions for A_2. The execution of A_3 denies the property P. Obviously, the precondition of action A_2 may be *possibly* denied in this nonlinear plan and the nonlinear plan contains deleted-condition conflicts.

The deleted-condition conflicts can be easily detected in linear planning. In classical nonlinear planning, to avoid deleted-condition conflict, instead of being asserted to be true, a property has to be asserted to be *necessarily true* (Chapman, 1987) at every step in a nonlinear plan. A nonlinear plan is a partial description of a set of linear plans and each of its linear plans is called a *completion* of the nonlinear plan. A property is said to be necessarily true at a step in a nonlinear plan if it holds true at that step in all the completions of the nonlinear plan. As the counterpart of the necessary truth, a property is said to be *possibly true* at a step in a nonlinear plan if it holds true at that step in some completions of the nonlinear plan. Corresponding to the necessary truth and possibly truth, Christensen (91) defined "plain truth" for proposition types in a nonlinear plan. If the current nonlinear plan is "complete" (linear), then the assertion of the plain truth of a proposition type is defined as usual. Otherwise, the assertion is defined "arbitrarily".

In nonlinear planning, we are only concerned with the necessary truth. Failing to maintain the necessary truth of properties in nonlinear planning leads to deleted-condition conflicts. To maintain the computational advantage of a reified temporal logic, we do not intend to introduce modal operators (such as $\Box$ in (Chapman, 1987)) to distinguish necessary truth assertions from other kinds of truth assertions. Because our interpretation of proposition types are over the necessarily maximal intervals, in our temporal logic, all the truth assertions are necessary truth assertions.

2.6 Weak Interpretation of Negation

In reified temporal logics in AI, interpreting assertions over time intervals leads to two possible interpretations for negation, a *weak interpretation* and a *strong interpretation of negation*. In the weak interpretation, the negation of a proposition type assertion, $\neg Holds(P, t_i, t_j)$, is true if and only if P does not hold true throughout the whole time interval (t_i, t_j). In other words, the formula $\neg Hold(P, t_i, t_j)$ holds true if and only if either the proposition type P changes its truth value during the time (t_i, t_j) or $NOT(P)$ (i.e. $\neg P$) holds true throughout the interval. In the strong interpretation, the negation of a proposition type assertion $\neg Holds(P, t_i, t_j)$ is true if and only if $NOT(P)$ holds throughout the interval (t_i, t_j). A logic with only strong negation has truth gaps, i.e. a proposition may be neither true nor false over a time interval. Researchers in AI have tended to use the weak interpretation of negation to preserve a simple two-valued logic (Allen, 1991a; Shoham, 1987). Allen argues that, in his temporal logic, the strong negation can still be inferred by a formula of the form:

$$\forall t'.In(t', t) \Longrightarrow \neg Holds(P, t')^1$$

This formula states that, for any sub-interval t' of t, P does not hold. In other word, $NOT(P)$ holds throughout the interval t.

In our temporal language, interpretation of negation becomes more complicated because assertions are interpreted exclusively over the necessarily maximal time intervals. For the weak interpretation of negation in our language, the negation of a proposition type assertion, $\neg Holds(P, t_i, t_j)$, is true if and only if it is not the case that P holds true throughout the necessarily maximal time interval (t_i, t_j). In the strong interpretation, the negation of a proposition type assertion, $\neg Holds(P, t_i, t_j)$, is true if and only if $NOT(P)$ holds throughout the necessarily maximal interval (t_i, t_j). Obviously, in our temporal language, using the strong interpretation will lead to even more truth gaps. This leaves us no choice but to adopt a weak interpretation of negation. Following Allen's example, we can still make assertions equivalent to the strong interpretation of negation and various "intermediate ones" between the two extreme ones. We will discuss this issue formally later in this section.

3 A Temporal Logic for Nonlinear Planning

Our formal language is a point-based, reified interval temporal logic. To represent and reason about nonlinear planning, time is modeled by a partial order structure; assertions of proposition types are interpreted over the *necessarily maximal* time intervals[2]; a proposition type is associated with a time interval through a global predicate; time points are the only temporal primitives in the logic and, therefore, time intervals are referred to by their end-points; there are two proposition types, property and action in the logic. To distinguish between these two types, we use two different global predicates. The atomic formula $Hold(P, t_i, t_j)$ expresses that the property P holds true over the necessarily maximal time interval (t_i, t_j) and the atomic formula $Exec(A, t_i, t_j)$ expresses that the action A is executed over the necessarily maximal time interval (t_i, t_j). The basic considerations for such a temporal logic will be discussed in the next section.

[1] This formula is expressed as $\forall t'.In(t', t) \Longrightarrow \neg P(t')$ in (Allen, 1991a)

[2] This will be explained in the next section.

3.1 Syntax

Given

DV: a set of data variables,

TV: a set of temporal variables which is disjoint from DV,

DF: a set of data function symbols,

TF: a set of temporal function symbols which is disjoint from DF,

$\mathcal{A}$: a set of primitive action symbols,

$\mathcal{P}$: the set of primitive property symbols which is disjoint from $\mathcal{A}$,

a set of data terms, T_D, is defined recursively as follows

1. if $t \in DV$, then $t \in T_D$,
2. if $f \in DF$, $arity(f) = n$ and $t_1, \ldots, t_n \in T_D$, then $f(t_1, \ldots, t_n) \in T_D$.

The definition of a set of *temporal terms*, T_T, is similar to that of data terms except that the arguments of temporal functions are not restricted to be temporal terms. A set of temporal terms, T_T, is defined as follows:

1. if $t \in TV$, then $t \in T_T$,
2. if $f \in TF$, $arity(f) = n$, and $t_1, \ldots, t_n \in T_T \bigcup T_D$, then $f(t_1, \ldots, t_n) \in T_T$.

The set of *well-formed formulas(wffs)* is defined inductively as follows:

1. If $dt_1, \ldots, dt_n \in T_D$, $tt_1, tt_2 \in T_T$, $P \in \mathcal{P}$, $arity(P) = n$, then $Hold(P(dt_1, \ldots, dt_n), tt_1, tt_2)$ is a wff,
2. if $dt_1, \ldots, dt_n \in T_D$, $tt_1, tt_2 \in T_T$, $A \in \mathcal{A}$, $arity(A) = n$, then $Exec(A(dt_1, \ldots, dt_n), tt_1, tt_2)$ is a wff,
3. if $tt_1, tt_2 \in T_T$, then $tt_1 \prec tt_2$, $tt_1 \preccurlyeq tt_2$ and $tt_1 = tt_2$ are wffs,
4. if φ and ψ are wffs, then so are $\neg\varphi$, $\varphi \wedge \psi$,
5. if φ is a wff and $v \in DV \bigcup TV$ is a free variable of φ, then $\forall v.\varphi[v]$ is a wff.

We assume the usual definitions of $\vee$, $\Longrightarrow$, $\Longleftrightarrow$ and $\exists$.

3.2 Semantics

Definition 1 *A* partial order time structure *is a pair* $(\mathcal{W}, \mathcal{R})$ *where* $\mathcal{W}$ *is a non-empty universe of time points and* $\mathcal{R}$ *is a* partial order relation *between the time points in* $\mathcal{W}$.

Definition 2 *An* interpretation *of a temporal language is a tuple :* $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{D}, \pi_F, \pi_P \rangle$ *where*

$(\mathcal{W}, \mathcal{R})$ *is a partial order time structure,*

$\mathcal{D}$ *is the domain of* data *for all the time points in* $\mathcal{W}$.

π_F *is a interpretation of the function symbols. If* $f \in DF \bigcup TF$ *and* $arity(f) = n$, *then* $\pi_F(f) \in (\mathcal{D} \bigcup \mathcal{W})^n \longrightarrow \mathcal{D} \bigcup \mathcal{W}$.

π_P *is a interpretation of the proposition types (actions and properties). If* $P \in \mathcal{P} \bigcup \mathcal{A}$ *and* $arity(P) = n$, *then* $\pi_P(P) \in \mathcal{D}^n \longrightarrow 2^{\mathcal{W} \times \mathcal{W}}$. *We require that if* $(t_1, t_2) \in \pi_P(P)(dt_1, \ldots, dt_n)$, *then* $(t_1, t_2) \in \mathcal{R}$. *To characterize the fact that assertions are interpreted over the necessarily maximal intervals,* π_P *has the property that, if* $(t_1, t_2), (t_3, t_4) \in \pi_P(P)(dt_1, \ldots, dt_n)$ *and* (t_1, t_2) *and* (t_3, t_4) *do not refer to the same time interval, then either* $(t_2, t_3) \in \mathcal{R}$ *or* $(t_4, t_1) \in \mathcal{R}$.

A *variable assignment* ν for a given interpretation $\mathcal{M}$, is a mapping from variable symbols to the elements of the domains, $\nu : (DV \bigcup TV) \longrightarrow (\mathcal{D} \bigcup \mathcal{W})$ which satisfies sort restriction, i.e. $\nu(x) \in \mathcal{D}$ if $x \in DV$ and $\nu(x) \in \mathcal{W}$ if $x \in TV$.

For a given interpretation $\mathcal{M}$ and variable assignment ν, a *term assignment* τ is a mapping, $\tau : (T_D \bigcup T_T) \longrightarrow (\mathcal{D} \bigcup \mathcal{W})$ which can be defined as follows.

1. If $t \in DV \bigcup TV$, then $\tau(t) = \nu(t)$,
2. if $f \in DF \bigcup TF$ and $arity(f) = n$, then $\tau(f(t_1, \ldots, t_n)) = \pi_F(f)(\tau(t_1), \ldots, \tau(t_n))$.

The semantics of the wffs is given with respect to an interpretation $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{D}, \pi_F, \pi_P, \rangle$ and a variable assignment ν. The *satisfaction relation* ($\models$)of the well formed formulas under such a pair is defined recursively as follows.

$$
\begin{array}{lcl}
\langle \mathcal{M}, \nu \rangle & \models & Hold(P(dt_1, \ldots, dt_n), tt_1, tt_2) \text{ iff} \\
& & (\tau(tt_1), \tau(tt_2)) \in \pi_P(P)(\tau(t_1), \ldots \tau(t_n)) \\
\langle \mathcal{M}, \nu \rangle & \models & Exec(A(dt_1, \ldots, dt_n), tt_1, tt_2) \\
& & \text{iff } (\tau(tt_1), \tau(tt_2)) \in \pi_P(A)(\tau(t_1), \ldots \tau(t_n)) \\
\langle \mathcal{M}, \nu \rangle & \models & t_1 \prec t_2 \text{ iff } (\tau(t_1), \tau(t_2)) \in \mathcal{R} \\
\langle \mathcal{M}, \nu \rangle & \models & t_1 \preccurlyeq t_2 \text{ iff } (\tau(t_1), \tau(t_2)) \in \mathcal{R} \text{ or } \tau(t_1) = \tau(t_2) \\
\langle \mathcal{M}, \nu \rangle & \models & \neg\varphi \text{ iff } \langle \mathcal{M}, \nu \rangle \models \varphi \text{ is false} \\
\langle \mathcal{M}, \nu \rangle & \models & (\varphi \wedge \psi) \text{ iff } \langle \mathcal{M}, \nu \rangle \models \varphi \text{ and } \langle \mathcal{M}, \nu \rangle \models \psi \\
\langle \mathcal{M}, \nu \rangle & \models & \forall v.\varphi \text{ iff } \langle \mathcal{M}, \nu' \rangle \models \varphi \text{ for all } \nu' \text{ which} \\
& & \text{agree with } \nu \text{ except possibly on } v.
\end{array}
$$

An interpretation $\mathcal{M}$ is a *model* for a wff φ (written as $\mathcal{M} \models \varphi$) if $\langle \mathcal{M}, \nu \rangle \models \varphi$ for all variable assignments ν. A wff is a *closed formula* if it contains no free variables. If a closed formula φ is satisfied by a interpretation $\mathcal{M}$ and a variable assignment, then $\mathcal{M}$ is a model of φ. A wff φ is said to be *satisfiable* if it has a model. A wff φ is said to be *valid* (written as $\models \varphi$) if its negation is not satisfiable.

3.3 Some Properties of the Logic

3.3.1 Necessary Maximal Interval Axiom

To capture the fact that the assertions of proposition types are interpreted over the necessarily maximal time intervals, we have the main axiom schemata, *necessarily maximal interval axiom* schemata, of the temporal logic as follows.

Axiom 1

$$\begin{aligned} &\forall t_i, t_j, t_i', t_j'.(Holds(P, t_i, t_j) \wedge Holds(P, t_i', t_j') \Longrightarrow \\ &\quad (t_i = t_i' \wedge t_j = t_j') \vee t_j \prec t_i' \vee t_j' \prec t_i) \end{aligned}$$

This necessarily maximal interval axiom schemata states that, for any two intervals over which a property holds true, either they are equal or one of them is strictly before the other. This main axiom schemata corresponds to Allen's main proposition constraint (Allen, 1984).

3.3.2 Assertions over Arbitrary Intervals

In our temporal logic, assertions of proposition types are interpreted over the necessarily maximal time intervals. We have already mentioned that this does not prevent us from referring to assertions necessarily over arbitrary time intervals, though such kinds of assertions are not really needed in specification of planning problems.

We are going to use $Hold^*(P, t_i, t_j)$ to express the assertion that the property P is necessarily true over a arbitrary interval (t_i, t_j) and use $Exec^*(A, t_1, t_2)$ to express the assertion that the action A is necessarily executed over a arbitrary interval (t_i, t_j).

A property is a "liquid" proposition type; that is, if it holds true over a time interval (t_i, t_j), it holds true over all sub-intervals of (t_i, t_j). According to this property, $Hold^*(P, t_i, t_j)$ can be defined as an abbreviation.

Definition 3 *For a property P,*

$$\begin{aligned} &Hold^*(P, t_i, t_j) \Longleftrightarrow \\ &\quad (\exists t_i', t_j'.(t_i' \preccurlyeq t_i \preccurlyeq t_j \preccurlyeq t_j') \wedge Hold(P, t_i', t_j')) \end{aligned}$$

From this definition, we have the following axiom to characterize that a property is a liquid proposition type.

Axiom 2

$$\begin{aligned} &Hold(P, t_i, t_j) \Longrightarrow \\ &\quad (\forall t_i', t_j'.t_i \preccurlyeq t_i' \preccurlyeq t_j' \preccurlyeq t_j \Longrightarrow Hold^*(P, t_i', t_j')) \end{aligned}$$

An action is both "gestalt" and "solid" proposition type, that is, if it is executed over a time interval, then the interval is both the maximal interval and the smallest possible interval for the action. According to this property, we can define $Exec^*(A, t_i, t_j)$ as follows.

Definition 4 *For an action A*

$$Exec(A, t_i, t_j) \Longleftrightarrow Exec^*(A, t_i, t_j)$$

This definition also characterizes the ordinary understanding of *event type* in AI. In fact, it seems that there is nothing in this logic which could prevent the action type from being extended to the more general event type.

3.4 Strong Negation

For simplicity, in the following discussion, a proposition type refers to a property or an action and the global predicate $True$ (or $True^*$) refers to $Hold$ ($Hold^*$) if the proposition type concerned is a property and refers to $Exec$ ($Exec^*$) otherwise.

In this temporal logic, we adopt weak interpretation of negation. On some occasions, however, we may need to use a *strong negation.* In Allen's definition of strong negation, for a property P, $NOT(P)$ holds true over a time interval if P does not hold for any of its sub-intervals. In Allen's temporal logic, strong negation can be defined only for properties (only for the global predicate Hold) and not for events and processes (not for Occur and Occurring). Shoham (87) gave his definition of strong negation as follows.

> "$NOT(x)$ holds over an interval if x does not hold for that interval, and if x does not hold over any of its sub-intervals, except perhaps at the end points."

Shoham argued that, in his case, the definition of strong negation can be applied to arbitrary proposition types. For property types, this definition is similar to that of Allen. However, for actions (events), this definition may cause some confusion. Suppose that A is an action and A holds true over a time interval (t_i, t_j). In general, an action type holds neither over two properly overlapping intervals nor over two intervals one of which properly contains the other. This implies that A does not hold true over any proper subinterval of (t_i, t_j). However, according to Shoham's definition, $NOT(A)$ should hold over any proper sub-intervals of (t_i, t_j). This is different from the intuitive understanding of strong negation of an action. Intuitively, we can assert neither $Exec(A, t'_i, t'_j)$ nor $Exec(NOT(A), t'_i, t'_j)$ if the action A is executed over a interval (t_i, t_j) and (t'_i, t'_j) is a proper subinterval of it.

Intuitively, what we can say is that $NOT(A)$ holds over the gaps of the maximal time intervals over which A holds true. In our temporal logic, because the assertions are interpreted exclusively over the necessarily maximal time intervals, it is much easier for us to refer to these gaps. Using the concept "completion of an interpretation" defined before, now, we give our definition for strong negation, which can be applied to both properties and actions. This definition can also be applied to event type in AI.

Definition 5 *In an interpretation $\mathcal{M}$, the negation of a proposition type, $NOT(P)$, holds true over a non-point interval (t_i, t_j) (e.g., $t_i \prec t_j$), expressed as $True^*(NOT(P), t_i, t_j)$, if and only if for any interval (t'_i, t'_j) over which P holds true in $\mathcal{M}$ (e.g., $True^*(P, t'_i, t_j)$), (t_i, t_j) and (t'_i, t'_j) do not properly overlap in all the completions of $\mathcal{M}$.*

In our temporal logic, this definition implies that:

$$True^*(NOT(P), t_i, t_j) \Longleftrightarrow \\ \forall t'_i, t'_j.(True(P, t'_i, t'_j) \Longrightarrow t'_j \preccurlyeq t_i \vee t_j \preccurlyeq t'_i)$$

Definition 6 *In an interpretation $\mathcal{M}$, the negation of a proposition type, $NOT(P)$, holds true over the necessarily maximal non-point interval (t_i, t_j), expressed as $True(NOT(P), t_i, t_j)$, if and only if (t_i, t_j) is the maximal interval such that, for any interval (t'_i, t'_j) over which P holds true in $\mathcal{M}$ ($True^*(P, t'_i, t_j)$), (t_i, t_j) and (t'_i, t'_j) do not properly overlap in all the completions of $\mathcal{M}$.*

This definition implies that:

$$True(NOT(P), t_i, t_j) \Longleftrightarrow \\ True^*(NOT(P), t_i, t_j) \wedge \forall t'_i, t'_j.(True^*(NOT(P), t'_i, t'_j) \Longrightarrow \\ (t_i \preccurlyeq t'_i \preccurlyeq t'_j \preccurlyeq t_j) \vee (t'_j \prec t_i \vee t'_j = t_i \wedge True(P, t_i, t_i)) \vee \\ (t_j \prec t'_i \vee t_j = t'_i \wedge True(P, t_j, t_j)))$$

In our temporal logic, a proposition type is required to hold over a series of disjoint intervals (Axiom 1). This requirement allows a primitive proposition type being true over point intervals which are maximal. Here, given a proposition type P, we define the intervals over which $NOT(P)$ holds true as the gaps of the necessarily maximal intervals over which P holds. Therefore, the negation of a primitive proposition type are required to hold only over non-point intervals. This is a restriction imposed by the logic itself. This restriction does not affect the application of the logic in planning, and does not affect its application to problem solving in almost all cases. In Allen's temporal logic, every proposition type is exclusively interpreted over non-point time intervals[3]. Allen and Hayes (87) claim that time points are "...mere mathematical abstractions and not times at which an event occurs or when some proposition is true". Unlike in Allen's logic, we maintain the ability to represent and reason about both point and interval actions. This is especially useful in planning.

In Allen's temporal logic, NOTs cancel out for property types. In Shoham's temporal logic, NOTs cancel out only for point-point-liquid proposition types[4] ($\updownarrow^p_p$). In this language, we can prove that NOTs cancel out for both properties and actions as long as both the proposition types and their negations hold true only over the maximal non-point interval (refer to (Zhang, 1994)).

Theorem 1 *Let P be a proposition type, (t_i, t_j) is a non-point time interval. Suppose that, for any the necessarily maximal interval (t'_i, t'_j) over which P or $NOT(P)$ holds true, $t'_i \prec t'_j$. Then*

$$True(NOT(NOT(P)), t_i, t_j) \Longleftrightarrow True(P, t_i, t_j)$$

[3] In (Lin, 1991), this is expressed as a lemma (Lemma 3):

$$\forall.P.\exists Start(P), End(P).(P \in Start(P) \wedge End(P) \wedge Start(P) \prec End(P))$$

and a formal proof is given.

[4] According to Shoham's definition, a proposition type is *point-point-liquid* if whenever it holds over an interval it holds at all of its internal points and whenever it holds at all internal points of some non-point interval it holds also over the non-point interval itself. This coincides with our property type.

3.5 Connective-like Functions

To allow properties to name complex logical expressions, besides the strong negation NOT, Allen also introduces the connective-like functions (data type functions) to the properties, AND and OR which correspond to the logical operators $\wedge$ and $\vee$([Allen 84]).

In our framework, like the strong negation NOT, these connective-like functions can be applied to both property types and to action types. As in Allen's temporal logic, the function AND can move through both Hold and Exec freely in our temporal logic:

$$True(AND(p,q),t_i,t_j) \Longleftrightarrow True(p,t_i,t_j) \wedge True(q,t_i,t_j)$$

Such a property, obviously, is not suitable for the function OR. As we already have the definitions of the functions NOT and AND, mirroring the normal definition of disjunction, the function OR can be defined in the same way as in [Allen 85]:

$$True(OR(p,q),t_i,t_j) \Longleftrightarrow True(NOT(AND(NOT(p),NOT(q))),t_i,t_j)$$

Other connective-like functions such as IMPLY and EQUIVALENCE can be defined in a similar way.

4 Application in Nonlinear Planning

In this section, we give a brief description of how the temporal logic is applied in deductive nonlinear planning.

4.1 Specification of Planning Problems

4.1.1 Specification of Actions

In planning, a given problem domain can be described by a set of actions Γ and a set of domain constraints Σ_D. In this temporal language, each action in Γ can be formally specified by a set of formulas in Horn clause form.

Example 1 *In the blocks world, let $Puton(x,y,z)$ be an action that moves block x from y to z, as shown in Figure 2a*[5]*. This action can be specified by the following two clauses.*

$$\begin{aligned}&Hold(On(x,z),t_1,SKF1(t_1,x,y,z)) :-\\&\quad Hold(Clear(x),t_2,t_2'), Hold(Clear(z),t_3,t_1), Hold(On(x,y),t_4,t_1),\\&\quad Exec(Puton(x,y,z),t_1,t_1), t_2 \prec t_1 \prec t_2';\end{aligned}$$

$$\begin{aligned}&Hold(Clear(y),t_1,SKF2(t_1,x,y,z)) :-\\&\quad Hold(Clear(x),t_2,t_2'), Hold(Clear(z),t_3,t_1), Hold(On(x,y),t_4,t_1),\\&\quad Exec(Puton(x,y,z),t_1,t_1), t_2 \prec t_1 \prec t_2';\end{aligned}$$

[5] In this definition, x is not equal to the table. If y is equal to the table, then $Clear(y)$ is not an asserted property. If z is equal to the table, then $Clear(z)$ is not a deleted property.

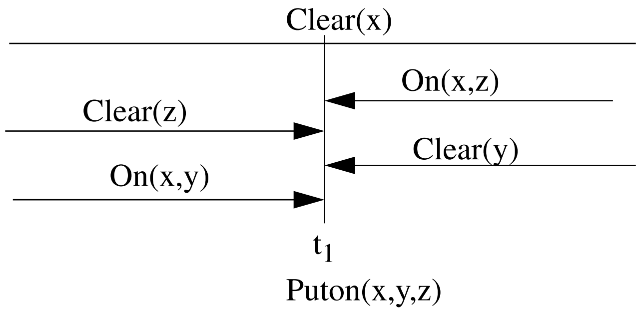
Puton(x,y,z):

Precondtions: {Clear(x), Clear(z), On(x,y)}

Deleted properties: {On(x,y), Clear(z)}

Asserted properties: {On(x,z), Clear(y)}

a. The action Puton(x,y,z)



b.The temporal structure of the point action Puton(x,y,z).

Figure 2: The point action Puton(x,y,z).

The causal relations and the temporal relations of the properties and the action involved are clearly specified in the formulas. For example, the first formula expresses that if the blocks x and z are clear, x is on y and the action $Puton(x, y, z)$ is executed, then x will be on z. The formula also expresses that the temporal intervals over which the deleted properties $Clear(z)$ and $On(x, y)$ hold true are finished at the time point t_1 at which the action is executed, the intervals over which the undeleted precondition $Clear(x)$ holds overrids the time point t_1, and the interval over which the asserted property $On(x, z)$ holds is started from t_1. SKF1 and SKF2 are Skolem functions.

In fact, the full specification of a planning rule should be a temporal formula which contains quantifiers. For example, an instance of the first planning rule in the above example should be of the form:

$$\begin{aligned}
&\forall t_1 \exists t.(\exists t_2, t_2', t_3, t_4.(Exec(Puton(A, B, C), t_1, t_1) \wedge t_2 \prec t_1 \prec t_2' \\
&\quad \wedge Hold(Clear(A), t_2, t_2') \wedge Hold(Clear(C), t_3, t_1) \wedge Hold(On(A, B), t_4, t_1)) \\
&\quad \Longrightarrow Hold(On(A, C), t_1, t))
\end{aligned}$$

In the formula, the temporal variable t is existentially quantified. This implies that, after being asserted, the property $On(A, C)$ will hold true until the time point at which it is denied by other actions or by the environment. However, if t were universally quantified, the formula would imply that, after being asserted, the property $On(A, C)$ would hold true at all the time points which are after t_1. This obviously contradicts the case in the real world.

From the above example, we can see that actions can be concisely, clearly and directly specified in our temporal logic. This is because that, in our temporal logic, assertions of proposition types are interpreted exclusively over the necessarily maximal intervals. The fact that a property p is asserted over an interval (t_i, t_j), $Hold(p, t_i, t_j)$, implies that (t_i, t_j) is the maximal interval for p; i.e. p is asserted at t_i and deleted at t_j. Therefore, causal relations and temporal relations of actions can be easily and directly expressed.

In general, in a given planning problem domain, the set of (point) actions Γ is a set of planning rules of the form

$$Hold(p, t, SKF(t, v_1, \ldots, v_m)) :- B_1, \ldots, B_n, Exec(A, t, t), C$$

where B_i, $i = 1, \ldots n$, the bodies of the planning rule are of the form $Hold(q_i, t_i, t_i')$, A is the action of the rule and C is a formula consists of the temporal constraints. If the above planning rule is in Γ, then p is said to be *an asserted property of* A, p is also said to be a *postcondition of* A. q_i, $i = 1 \ldots n$, are said to be the *preconditions of* A. For any B_i, if it is of the form $Hold(q_i, t_i, t)$, then it is said to be *a deleted property of* A. Otherwise, it is also said to be a postcondition of A. SKF is a Skolem function and $v_1, \ldots, v_m$ are all the universally quantified data variables that appear in p.

Now, we give an example to show how an interval action can be specified as temporal formulas in Horn clausal form.

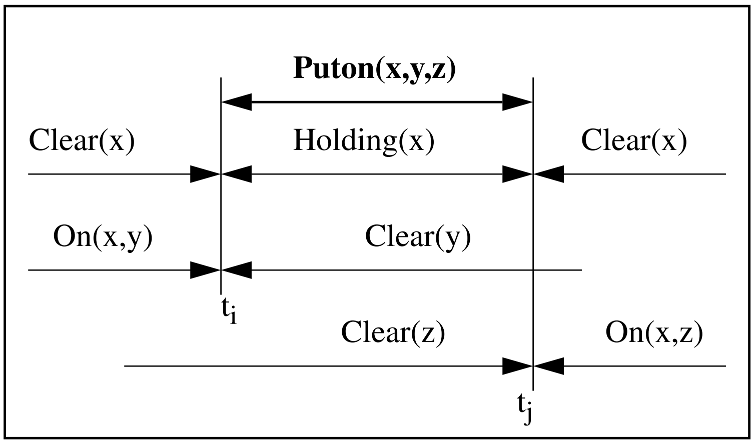


Figure 3: An interval action $Puton(x, y, z)$.

Example 2 *Figure 3 describes an interval action* $Puton(x, y, z)$ *over an interval* (t_i, t_j). *This interval action can be specified by the following temporal formulas in Horn clausal form.*

$$\begin{aligned} & Hold(On(x, z), t_j, SKF1(t_j, x, y, z)) :- \\ & \quad Hold(Clear(x), t_1, t_i), Hold(Clear(z), t_2, t_j), Hold(On(x, y), t_3, t_i), \\ & \quad Exec(Puton(x, y, z), t_i, t_j); \end{aligned}$$

$$Hold(Clear(y), t_i, SKF2(t_i, x, y, z)) :-$$
$$Hold(Clear(x), t_1, t_i), Hold(Clear(z), t_2, t_j), Hold(On(x, y), t_3, t_i),$$
$$Exec(Puton(x, y, z), t_i, t_j);$$

$$Hold(Clear(x), t_j, SKF3(t_j, x, y, z)) :-$$
$$Hold(Clear(x), t_1, t_i), Hold(Clear(z), t_2, t_j), Hold(On(x, y), t_3, t_i),$$
$$Exec(Puton(x, y, z), t_i, t_j);$$

In this specification, the action is executed over the interval (t_i, t_j), the deleted properties of the action $On(x, y)$ and $Clear(z)$ are deleted at different time points t_i and t_j respectively. The asserted properties $Clear(y)$ and $On(x, z)$ are asserted at different points t_i and t_j respectively. The undeleted precondition of the action $Clear(x)$ is deleted at the starting point of the action and is reasserted at the ending point of the action. Comparing this specification with the action in Figure 3, the set of planning rules clearly specifies the temporal relations and the causal relations among the action and the related properties.

4.1.2 Specification of Initial Conditions and Goals

The initial conditions of a planning problem can be expressed as special planning rules, each with a empty body:

$$Hold(p, T_0, SKC(v_1, \ldots, v_m)) :-$$

where SKC is a Skolem function. The formula expresses that the property p holds true from the initial time point T_0 to a time point at which it is deleted.

The specification of a planning problem in a problem domain consists of a set of planning rules Γ (including actions and initial conditions), a set of domain constraints Σ_D and a goal to be achieved. In this planning language, a goal is a definite constraint planning rule with an empty head: $:-G_1, \ldots, G_n | C_0$ where $G_1, \ldots, G_n$ are the goals to be achieved and C_0 describes the temporal relation between the individual goals.

For example, in the blocks world, the goal that block A is on block B and B is on block C should be precisely described as follows:

> There exists a time point t such that the two time intervals over which the properties $On(A, B)$ and $On(B, C)$ hold respectively properly override t.

This goal can be expressed as a definite constraint planning rule:

$$:-Hold(On(A, B), t_1, t_2), Hold(On(B, C), t_3, t_4) | t_1 \prec t \prec t_2 \wedge t_3 \prec t \prec t_4$$

4.1.3 Specification of Domain Constraints

In a state-based language, like situation logics, a domain constraint expresses that two properties which are defined to be mutually exclusive are not allowed to hold true at the same state or time point. In our nonlinear interval temporal logic, if two properties are defined to be mutually exclusive, then the temporal intervals over which they hold true respectively must not *possibly* and properly overlap.

In our temporal logic, a set of domain constraints Σ_D for a problem domain is a set of formulas of the form:

$$\forall t_1, t_2, t_3, t_4.(Hold(p, t_1, t_2) \wedge Hold(q, t_3, t_4) \Longrightarrow t_2 \preccurlyeq t_3 \vee t_4 \preccurlyeq t_1)$$

If Σ_D contains the above formula, the properties p and q are said to be *mutually exclusive* according to Σ_D.

It is not difficult to show that if a nonlinear temporal model satisfies a domain constraint of the above form, then the intervals of the two properties involved will not possibly and properly overlap.

4.2 Constraint Logic Planning

Based upon our temporal logic, we build a deductive nonlinear planning framework, constraint logic planning. Here, we give a brief description of the constraint logic planning.

In our temporal logic, a nonlinear plan can be represented as a temporal formula. For example, in the blocks world, let P be a nonlinear plan consisting of two actions $Exec(Puton(A, T, B), t_1, t_1)$, which moves block A from the table T to the top of block B at the time point t_1, and $Exec(Puton(B, T, C), t_2, t_2)$, which moves block B from the table T to the top of block C at the time point t_2. In P, both time points t_1 and t_2 are after the initial time point T_0 and there is no direct temporal relation between t_1 and t_2. This nonlinear plan can be expressed as a well formed formula in our temporal logic as follows.

$$\begin{aligned} P &= Exec(Puton(A, T, B), t_1, t_1) \wedge T_0 \prec t_1 \wedge \\ &\quad\; Exec(Puton(B, T, C), t_2, t_2) \wedge T_0 \prec t_2 \end{aligned}$$

Syntactically, a nonlinear plan can be expressed as a well formed temporal formula. Semantically, a nonlinear plan describes a nonlinear temporal model. That is, under the *closed world assumption* and the specification of actions, a temporal model can be constructed for a given nonlinear plan. For example, based on the action $Puton(x, y, z)$ given in Figure 2, the temporal model for the above nonlinear plan is shown in Figure 4. In this figure, T_∞ represents the universal ending point. A bi-directed line of a property indicates a time interval over which the property hold true. For example, the property $On(B, T)$ holds true over the interval that is started from T_0 and is ended at t_2.

To provide a formal framework for nonlinear planning, in (Zhang, 1994), we proposed a set of restrictions on actions. We showed that the set of restrictions can prevent actions from being improperly defined. We also argued that actions in most of the problem domains either satisfy the restrictions or can be reasonably modified to satisfy the restrictions. We proved that, under the set of restrictions on actions, a nonlinear plan is conflict free if and only if the corresponding set of domain constraints is satisfied in the temporal model of the nonlinear plan. For example, the nonlinear plan given above contains conflict. The action $Puton(A, T, B)$ possibly deletes one of the preconditions $Clear(B)$ of the action $Puton(B, T, C)$. This conflict in P can be detected by the fact that the domain constraint

$$Hold(On(A, B), t_1, T_\infty) \wedge Hold(Clear(B), T_0, T_\infty) \Longrightarrow T_\infty \preccurlyeq t_1 \vee T_\infty \preccurlyeq T_0$$

is not satisfied in the temporal model of P, shown in Figure 4.

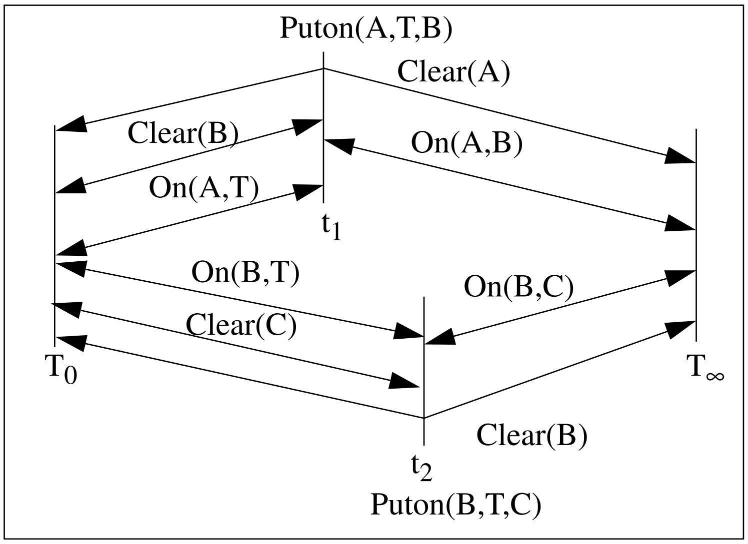


Figure 4: The temporal model of a nonlinear plan which contains conflict.

In (Zhang, 1994), we showed that, from a set of domain constraints, we can obtained a set of temporal formulas, which we termed action constraints. For example, one such action constraint in the blocks world expresses that the actions $Puton(x, y, z)$ and $Puton(z, y', x)$ should not possibly and properly overlap. We proved that nonlinear plans that satisfy a certain property are conflict free if and only if they satisfy the corresponding set of action constraints (we showed that all nonlinear plans constructed from our constraint logic planning satisfy this property and we also proved that all the nonlinear plans that do not satisfy this property contain conflicts). Therefore, to verify the conflict freeness of a nonlinear plan, we need not check domain constraints over the temporal model of the nonlinear plan, we need only check the set of action constraints on the nonlinear plan itself. We termed this conflict detecting method action constraint maintenance.

Planning in our framework, we simply continuously introduce actions with the corresponding temporal relations to achieve goals by resolving the goals with the corresponding planning rules and check if the plan under construction is consistent with the corresponding set of action constraints. The soundness and completeness of the action constraint maintenance guarantee that the constructed plans are conflict free. Also, we proved that if a nonlinear plan is not consistent with an action constraint, all the possible expansions of the plan will not be consistent with the constraint. Therefore, in our constraint logic planning, backtracking from an inconsistency will not lose possible solutions. This lays down the fundation for the completeness of constraint logic planning.

5 Conclusion

Most research areas within AI involve reasoning about time. The most popular temporal logics adopted in AI are those of Allen (85) and McDermott (81). As Shoham (87) pointed out, however, neither the temporal logic of Allen nor that of McDermott has completely clear semantics. The temporal logic proposed by Shoham, which borrows heavily from the intuitions and formulations of Allen and MeDermott, has precise syntax and semantics. However, Shoham did not discuss how his temporal logic can be applied, for example, to planning.

Our temporal logic proposed in this report bears a strong resemblance to that of Shoham and McDermott. However, our temporal logic is proposed especially as a formal language for nonlinear planning and has precise syntax and semantics. Compared with other reified temporal logics in AI, the most specific characteristic of this temporal logic is that assertions of proposition types are interpreted over the necessarily maximal intervals. Because of this characteristic, when applying the temporal logic in planning, we can deal with the frame problem efficiently. This characteristic also unifies the assertions of actions and properties and unifies strong negations of actions and properties. Also according to this characteristic, the fact that a property p holds true over an interval (t_i, t_j), $Hold(p, t_i, t_j)$, implies that p is asserted by some action or the environment at t_i and p remains true until t_j at which p is deleted by some other action or the environment. As a result, in specification of a planning problem, the causal relations and temporal relations between an action and its properties (i.e. its preconditions and postconditions) can be directly, concisely and clearly described using this temporal logic.

Directly upon the temporal logic, we build our deductive nonlinear planning framework, constraint logic planning. In (Zhang, 1994), we proved that our constraint logic planning is sound, complete and is efficient.

6 Acknowledgements

We would like to thank Behnam Bani-Eqbal, David Carlisle, Clare Dixon, Rajeev Goré and Ian Pratt who read an early version of this report and gave valuable comments.

This paper is a modified version of part of Zhang's Ph.D. thesis (Zhang, 1994). The work was supported by the British Council and the State Education Commission of China.

References

Allen, James F. 1990. Formal Models of Planning. *Pages 50–54 of:* Allen, J., Hendler, J., & Tate, A. (eds), *Readings in Planning.* Morgan Kaufmann.

Allen, James F. 1991a. Planning as Temporal Reasoning. *Pages 3–14 of: Proceedings of the 12th International Joint Conference on Artificial Intelligence.*

Allen, James F. 1991b. Temporal Reasoning and Planning. *In:* Allen, James F., Kautz, H., Pelavin, R., & Tenenberg, J. (eds), *Reasoning about Plans.* San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Allen, James F., & Koomen, J. A. 1983. Planning Using a Temporal World Model. *In: Proceedings of the 8th International Joint Conference on Artificial Intelligence.*

Allen, J.F. 1984. Towards a General Theory of Action and Time. *Artificial Intelligence*, **23**, 123–154.

Allen, J.F., & Hayes, P.J. 1985. A Common-Sense Theory of Time. *Pages 528–531 of: Proceedings of the 9th International Joint Conference on Artificial Intelligence.*

Biundo, S., Dengler, D., & Koehler, J. 1992. Deductive Planning and Plan Reuse in a Command Language Environment. *Pages 628–632 of: Proceedings of the 10th European Conference on Artificial Intelligence.*

Chapman, David. 1987. Planning for Conjunctive Goals. *Artificial Intelligence*, **32**, 333–377.

Christensen, Jens. 1990. A Hierarchical Planner that Generates its Own Hierarchies. *Pages 1004–1009 of: Proceedings of AAAI-90.*

Christensen, Jens, & Grove, A. 1991. A formal Model for Classical Planning. *Pages 246–251 of: Poceedings of the 12th International Joint Conference on Artificial Intelligence.*

Cranefield, Stephen J. S. 1990. *Planning for Behaviour-Based Robotic Assembly: A Logical Framework.* Ph.D. thesis, University of Edinburgh.

Currie, Ken, & Tate, Austin. 1989 (May). *O-Plan: the Open Planning Architecture.* Tech. rept. AIAI-TR-67. Artificial Intelligence Applications Institute, University of Edinburgh, Edinburgh.

Dean, T. 1985. Temporal Reasoning Involving Counterfactuals and Disjunctions. *In: Proceedings of the 9th International Joint Conference on Artificial Intelligence.*

Fariñas del Cerro, Luis. 1982. A simple Deduction Method for Modal Logic. *Information Processing Letters*, **14**(2).

Fariñas del Cerro, Luis, & Herzig, A. 1988. *Quantified Modal Logic and Unification Theory.* Tech. rept. Rapport LSI n° 293. IRIT, Université Paul Sabatier.

Fariñas del Cerro, Luis, & Herzig, A. 1989. Deterministic Modal Logics for Automated Deduction. *In:* Brazdil, P., & Konolige, K. (eds), *Machine Learning, Meta Reasoning and Logics.* Kluwer Academic Publishers.

Fariñas del Cerro, Luis, & Herzig, A. 1990. Automated Quantified Modal Logic. *In: Proc. ECAI 90.*

Fikes, R. E., & Nilsson, N. J. 1971. STRIPS: a New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, **2**, 189–208.

Frisch, A. M., & Scherl, R. B. 1990 (Sep. 10 -14). A Constraint Logic Approach to Modal Deduction. *Pages 234–250 of: Proc. European Workshop JELIA 90, LNCS 478.*

Green, Cordell. 1969. Application of Theorem Proving to Problem Solving. *Pages 741–747 of: Proceedings of the 1st International Joint Conference on Artificial Intelligence.*

Halpern, J.K., & Shoham, Y. 1986. A Propositional Modal Logic of Time Intervals. *In: Proceedings Symposium on Logic in Computer Science.*

Halphern, J. Y., Manna, Z., & Moszkowski, B. 1983. A High-Level Semantics Based on Interval Logic. *Pages 278–291 of: Proceedings of ICALP-83, Lecture Notes in Computer Science 154*. Springer Verlag.

Kautz, H. A. 1982. Planning Within First-Order Dynamic Logic. *Pages 19–26 of: Proceedings of the CSCSI/SCEIO.*

Kowalski, R. 1979. *Logic for Problem Solving.* New-York: Elsevier North-Holland, Inc.

Lin, Yunqing. 1991. Two Theories of Time. *Journal of Applied Non-Classical Logics*, **1**(1), 37–63.

Manna, Z., & Waldinger, R. 1980. A Deductive Approach to Program Synthesis. *ACM Transactions on Programming Languages and Systems*, **2**(1), 90–121.

Manna, Z., & Waldinger, R. 1986. A Theory of Plans. *Pages 11–45 of:* Georgeff, M., & Lansky, A. (eds), *Proceeding of the 1986 Workshop on Reasoning About Actions and Plans.*

McCarthy, J., & Hayes, P.J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, **4**, 463–502.

McDermott, D.V. 1982. A Temporal Logic for Reasoning About Processes and Plans. *Cognitive Science*, **6**, 101–155.

Miller, D., Firby, R. J., & Dean, T. 1985. Deadlines, Travel Time and Robot Problem Solving. *Pages 1052–1054 of: Proceedings of the 9th International Joint Conference on Artificial Intelligence.*

Moszkowski, B. 1986. *Executing Temporal Logic Programs.* Cambridge University Press.

Ohlbach, Hans J. 1988. A Resolution Calculus for Modal Logics. *Pages 500–516 of: Proceedings of the 9th International Conference on Automated Deduction, LNCS 310.* Springer Verlag.

Pednault, E. P. D. 1991. Generalizing Nonlinear Planning to Handle Complex Goals and Actions with Context-Dependent Effects. *In: Proceedings 12th International Joint Conference on Artificial Intelligence.*

Reichgelt, H. 1987. Semantics for Reified Temporal Logic. *Pages 49–61 of:* Hallam, J., & Mellish, C. (eds), *Advances in Artificial Intelligence (Proc. AISB conf., University of Edinburgh).* Wiley.

Rosenschein, S. J. 1981. Plan Synthesis: A Logical Perspective. *In: Proceedings of the 7th International Joint Conference on Artificial Intelligence.*

Sacerdoti, Earl D. 1975. The Non-linear Nature of Plans. *In: Proceedings of the 4th International Joint Conference on Artificial Intelligence.*

Shoham, Yoav. 1987. Temporal Logics in AI: Semantical and Ontological Considerations. *Artificial Intelligence*, **33**, 89 –104.

Tate, Austin. 1986. Goals Structures, Holding Periods and "Clouds". *In:* Georgeff, M., & Lansky, A. (eds), *Proceedings of the Workshop Reasoning about Actions and Plans.*

Tsang, E. P. K. 1987 (April). TLP - A Temporal Planner. *In: Proc. AISB-87.*

Vere, S. A. 1983. Planning in Time: Windows and Durations for Activities and Goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-5**(3), 246–267.

Vere, S. A. 1985a. Splicing Plans to Achieve Misordered Goals. *Pages 1016–1021 of: Proceedings of the 9th International Joint Conference on Artificial Intelligence.*

Vere, S. A. 1985b. Temporal Scope of Assertions and Window Cutoff. *Pages 1055–1059 of: Proceedings of the 9th International Joint Conference on Artificial Intelligence.*

Vilain, Marc, & Kautz, Henry. 1986. Constraint Propagation Algorithms for Temporal Reasoning. *Pages 377–382 of: Proceedings of AAAI-86.*

Waldinger, R. 1977. Achieve Several Goals Simultaneously. *Pages 94 –137 of:* Elcock, E. W., & Michie, D. (eds), *Machine Intelligence 8.* Ellis Horwood Limited and John Wiley.

Wallen, L. A. 1987. Matrix Proof Methods for Modal Logic. *In: Proceedings of the 10th International Joint Conference on Artificial Intelligence.*

Zhang, Ying. 1994 (May). *Constraint Logic Planning: A Formal Framework for Nonlinear Planning.* Ph.D. thesis, Manchester Univ., Computer Science Dept.