Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1999), *13*, 67–78. Printed in the USA.

Temporal representation of state transitions

JIXIN MA, BRIAN KNIGHT, AND EPHRAIM NISSAN

School of Computing and Mathematical Science, University of Greenwich, London SE18 6PF, United Kingdom

(RECEIVED May 28, 1998; REVISED October 19, 1998; ACCEPTED November 4, 1998)

Abstract

This paper describes a knowledge-based temporal representation of state transitions for industrial real-time systems. To allow expression of uncertainty, we shall define fluents as disjuncts of positive/negative time-varying properties. A state of the world is represented as a collection of fluents, which is usually incomplete in the sense that neither the positive form nor the negative form of some properties can be implied from it. The world under consideration is assumed to persist in a given state until an action(s) takes place to effect a transition of it into another state, where actions may either be instantaneous or durative. High-level causal laws are characterized in terms of relationships between actions and the involved world states. An effect completion axiom is imposed on each causal law to guarantee that all the fluents that can be affected by the performance of the corresponding action are governed. This completion requirement is practical for most industrial real-time applications and in fact provides a simple and effective treatment to the so-called frame problem.

Keywords: Temporal Representation; State Transitions; Industrial Real-Time Systems

1. INTRODUCTION

The notion of state transitions is fundamental for many real-time applications. In particular, in monitoring highly complex industrial systems, it is often necessary to reason about varying states, as in the case of processes that evolve with time. During the last three decades, various approaches have been proposed, to the representation of, and reasoning about, both the static and dynamic aspects of the world. The static images of the world under consideration are usually denoted by states that can be described in terms of facts about the worlds, while the dynamic images are characterized by actions whose performance may cause state transitions. An early example of such a treatment of states and actions is the framework of the *situation calculus*, developed by McCarthy and Hayes (1969). In the situation calculus, the state of the world at a given instant is called a situation, while actions constitute transition functions between situations that change the values of fluents (i.e., time-varying properties). Another influential formalism for dealing with actions (events) and state changes is the so-called *event calculus* introduced by Kowalski and Sergot (1986). As being the primitives of the event calculus, events, which are associated with structureless "points" in time, initiate and/or terminate *processes* (i.e., something happening over periods of time) during which states are maintained.

As an example, consider the operating cycle of an automatic tea maker, which can be represented as a finite-state system. The static images of such a system may be denoted by the following 10 states:

State S_1, where:
the tea maker is clean; the switch is at "off" position; and there is neither water nor tea bag inside.

State S_2, where:
the switch is at "off" position; water is not boiling; and there is no tea bag inside.

State S_3, where:
the switch is at "on" position; water is not boiling; and there is no tea bag inside.

State S_4, where:
the switch is at "on" position; water is boiling; and there is no tea bag inside.

State S_5, where:
the switch is at "on" position; water is boiling with tea bag inside.

Reprint requests to: Jixin Ma, School of Computing and Mathematical Science, University of Greenwich, London SE18 6PF, United Kingdom.

State S_6, where:
the switch is at "off" position; water is boiling with tea bag inside.

State S_7, where:
the switch is at "off" position; tea is ready with tea bag inside.

State S_8, where:
the switch is at "off" position; tea is too cold with tea bag inside.

State S_9, where:
the switch is at "on" position; water is not boiling but with tea bag inside.

State S_{10}, where:
the tea maker is dirty; the switch is at "off" position; there is tea bag but no water inside.

State transitions are effected by the performance of some certain types of actions. For instance, starting from state S_3, the performance of the action type "Heating" will effect the transition of world into state S_4.

In the framework of situation calculus, such a state-transition cycle would be expressed as:

$Result(\text{AddWater}, S_1) = S_2$, $Result(\text{SwitchingOn}, S_2) = S_3$, $Result(\text{Heating}, S_3) = S_4$,

$Result(\text{AddTeaBag}, S_4) = S_5$, $Result(\text{SwitchingOff}, S_5) = S_6$, $Result(\text{Cooling}, S_6) = S_7$,

$Result(\text{Pouring}, S_7) = S_{10}$, $Result(\text{Cooling}, S_7) = S_8$, $Result(\text{SwitchingOn}, S_8) = S_9$,

$Result(\text{Heating}, S_9) = S_5$, $Result(\text{Cleaning}, S_{10}) = S_1$,

where *Result* is the function that maps a pair of an action and a situation (state) into another situation (state). Similar expression can be reached by using the event calculus.

However, as noted by many researchers, the temporal ontology in the original versions of both the situation calculus and the event calculus is quite weak. For instance, in the situation calculus, the time points to which situations are referred are not explicitly expressed. In fact, being treated just as a kind of context or time labels, a situation is associated with the set of facts that is used to describe the corresponding state of the world. Therefore, on the one hand, as in most versions of the conventional situation calculus (Genesereth & Nilsson, 1987), if we take the set of labels, $\{s_i\}$, to refer to the set of states of the world, then we cannot carry out any temporal reasoning because there is no temporal knowledge at all. On the other hand, if we take $\{s_i\}$ as the set of situations by means of ordering their subscripts and hence to associate them with different times, then the commonsense constraint, that is, "effects never precede their causes," will restrict the expression that a certain state of the world may appear repeatedly at different times.

To enrich the expressive power of the situation calculus and the event calculus, various revisions have been proposed to characterize richer temporal features (Sandewall & Ronquist, 1986; Gelford et al., 1991; Lin & Shoham, 1991; Pinto & Reiter, 1995; Miller & Shanahan, 1994). However, these approaches have not gone as far as one would like for dealing with general temporal issues because there are still some problematic issues that have not been satisfactorily solved. Specially, in most of the existing revised frameworks:

(a) The fundamental time structure is neither formally characterized nor explicitly expressed. This may lead to difficulties in modelling issues such as continuous changes, point-critical phenomenon, and so on.

(b) They are not powerful enough in either expressing the persistence of a state of the world with respect to some certain time intervals, or expressing the duration knowledge of the life span of actions/events. In fact, for general treatments, the notion of both points and intervals are needed for temporal references of instantaneous and durative phenomenon, respectively. For instance, in Figure 1, by common sense, action "SwitchingOn" should be associated with a time point, while action "Cleaning" should be associated with a time interval.

(c) Formal high-level causal laws (Gooday & Galton, 1996) in terms of relations between actions and their effects are expected in representing common sense knowledge about causation. For example, "Whenev-

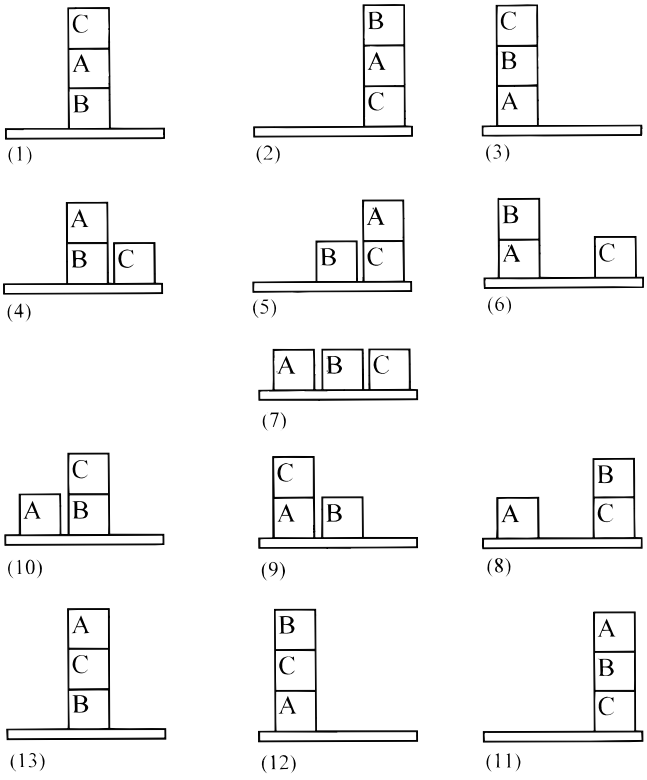


Fig. 1. Thirteen possible complete states that correspond to different configurations of blocks.

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

er over the time line, starting from a given state, the performance of some specified type of action will change the world into a certain state."

(d) The so-called frame problem (i.e., the problem of specifying everything that will not be affected by certain actions) need special treatments.

The objective of this paper is to propose a knowledge-based temporal representation of state transitions that we believe is useful for many engineering applications. In Section 2, we shall present the logical preliminaries of the language. The language will be described as a first-order reified logic with various sorts of objects, including nontemporal elements (individuals), time elements, properties, fluents (i.e., disjuncts of positive/negative time-varying properties), states and action types, etc. To support not only instantaneous but also durative temporal references, intervals and points are taken as primitive in the time ontology. Also, to express some uncertain knowledge about the state of the world, the definition of fluents is extended to allow disjunctions of positive/negative properties. A state of the world is represented as a collection of fluents, which is usually incomplete in the sense that some properties (or their negations) cannot be implied from such a collection of fluents. Following the standard approach, the world under consideration is assumed to persist in a given state until an action(s) takes place over some specified time to effect a transition of it into another state, where actions may either be instantaneous or durative. High-level causal laws are characterized in terms of relations between actions and their effects. Specially in this paper, an assumption is made in the form of a constraint imposed on each causal law, which states that all the fluents that may be possibly affected by a certain action (type) are always governed by any causal law involved. Although such an assumption may seem too strong from the theoretical point of view, it is practical for most industrial real-time applications and in fact provides a simple and effective treatment to the corresponding frame problem. Application examples of the proposed formalism are given in Section 3. Finally, Section 4 recapitulates and draws conclusions.

2. THE LANGUAGE

The language, denoted as L, is a many-sorted first-order reified logic with equality, consisting of a tuple (**X**, **T**, *Meets*, **P**, **F**, **S**, **E)**:

- **X**—a sort of individuals denoting the objects of the world under consideration;
- **T**—a sort of *time elements*;
- *Meets*—an immediately predecessor relation over *time elements*;
- **P**—a sort of properties;
- **F**—a sort of *fluents*;
- **S**—a sort of *states*; and
- **A**—a sort of *actions* (or more strictly, *event types)*.

In particular, we shall denote the members of **T**, **P**, **F**, **S**, and **A** by (possibly scripted) letters t, p, f, s, and a, respectively, while the notation for the members of **X** is application-oriented. In addition, for representing temporal duration, we shall simply adopt the conventional theory of real numbers.

2.1. The temporal basis

In this paper, we shall take the following time theory proposed previously by Ma and Knight (1994) as the temporal basis for the formalism. A time element may either be a time interval or a time point. The distinction between time intervals and time points is characterized by means of a duration assignment function, *Dur*, from the set of time elements to non-negative real numbers. For example, $Dur(t) = 0$, $Dur(t') = 0.23$, etc. A time element t is called an (time) interval if $Dur(t) > 0$; otherwise, t is called a (time) point.

The primitive relation over time elements, *Meets*, is axiomatized by:

(T1) $Meets(t_1, t_2) \wedge Meets(t_1, t_3) \wedge Meets(t_4, t_2) \Rightarrow Meets(t_4, t_3)$
that is, the "place" where two time elements meet is unique.

(T2) $\exists t', t''(Meets(t', t) \wedge Meets(t, t''))$
that is, every time element has at least one neighboring time element preceding it, and another succeeding.

(T3) $\exists t', t''(Meets(t', t_1) \wedge Meets(t_1, t'') \wedge Meets(t', t_2) \wedge Meets(t_2, t'')) \Rightarrow t_1 = t_2$
that is, the time element connecting two meeting places is unique.

(T4) $Meets(t_1, t_2) \Rightarrow \exists t \forall t', t''(Meets(t', t_1) \wedge Meets(t_2, t'') \Rightarrow Meets(t', t) \wedge Meets(t, t''))$
that is, if two meeting places are separated by a sequence of time elements, then there is a time element that connects these two meeting places. Hence, by recalling axiom (T3), for any two adjacent time elements t_1 and t_2, we may denote the *ordered union* of t_1 and t_2 as a time interval, $t = t_1 \oplus t_2$, where $t_1 \oplus t_2$ always implies that $Meets(t_1, t_2)$.

(T5) $Meets(t_1, t_2) \Rightarrow Dur(t_1) > 0 \vee Dur(t_2) > 0$
that is, points cannot meet other points.

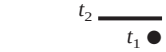
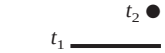
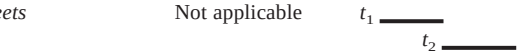
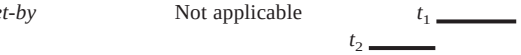
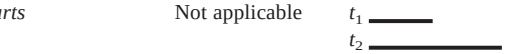
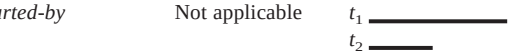
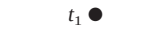
(T6) $Meets(t_1, t_2) \Rightarrow Dur(t_1 \oplus t_2) = Dur(t_1) + Dur(t_2)$
that is, the ordered union operation "$\oplus$" over time elements is in agreement with the conventional addition "$+$" over real numbers.

(T7) $Dur(t) > 0 \Rightarrow \exists t_1, t_2(t = t_1 \oplus t_2)$
that is, each interval can be decomposed into two time elements that meet each other. In fact, axiom (T7) axiomatizes the density of the time structure, which is needed in modelling some continuous physical processes.

Other temporal relationships over time elements, analogous to those introduced by Allen (1984) for intervals, can be classified as in Table 1.

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

Table 1. *Classification of temporal relations*

Relation Relating	Point t_1 to point t_2	Interval t_1 to interval t_2	Point t_1 to interval t_2	Interval t_1 to point t_2
Equal			Not applicable	Not applicable
Before				
After				
Meets	Not applicable			
Met-by	Not applicable			
Overlaps	Not applicable		Not applicable	Not applicable
Overlapped-by	Not applicable		Not applicable	Not applicable
Starts	Not applicable			Not applicable
Started-by	Not applicable		Not applicable	
During	Not applicable			Not applicable
Contains	Not applicable		Not applicable	
Finishes	Not applicable			Not applicable
Finished-by	Not applicable		Not applicable	

It is important to note that the distinction between the assertion that "point t_1 *Meets* interval t_2" and the assertion that "point t_1 *Starts* interval t_2" is critical: while $Starts(t_1, t_2)$ states that point t_1 is the starting part of interval t_2, $Meets(t_1, t_2)$ implies that point t_1 is not a part of interval t_2 at all.

The definition of the derived temporal relations in terms of the *Meets* relation is straightforward. For example, *Before* can be defined as:

$$Before(t_1, t_2) \Leftrightarrow \exists t(Meets(t_1, t) \wedge Meets(t, t_2)).$$

The above definition for the predicate $Before(t_1, t_2)$ accommodates the case where more than one intermediate time elements, $t_1, \ldots, t_m$, stand between t_1 and t_2, because by axioms (T3) and (T4), we can write $t_1 \oplus \ldots \oplus t_m = t$.

For the convenience of expression, we introduce two more derived temporal relations:

$$In(t_1, t_2) \Leftrightarrow Starts(t_1, t_2) \vee During(t_1, t_2) \vee Finishes(t_1, t_2)$$

$$Sub(t_1, t_2) \Leftrightarrow Equal(t_1, t_2) \vee Starts(t_1, t_2) \vee During(t_1, t_2) \vee Finishes(t_1, t_2).$$

Therefore, $In(t_1, t_2)$ denotes that time t_1 is a proper part of time t_2, while $Sub(t_1, t_2)$ denotes that time t_1 is either a proper part of time t_2 or is t_2 itself.

2.2. Properties, fluents, and states

A *property* is a statement (or proposition) that is either true or false. In language L, the truth value of each property is dependent on times, and the sort of fluents, **F**, is defined as the minimal set closed under the following rules:

1. Any property p in sort **P** is a fluent, that is, $p \in \mathbf{F}$;
2. If $f_1, f_2 \in \mathbf{F}$ then $f_1 \vee f_2 \in \mathbf{F}$;
3. If $f \in \mathbf{F}$ then not $(f) \in \mathbf{F}$.

To associate a fluent with a time element, a reified-predicate (Ma & Knight, 1996), *Holds*, is used here to substitute the formula $Holds(f, t)$ for each pair of a fluent f and a time element t, denoting that fluent f holds true over time t.

Remembering that *Holds* is a reified predicate, a certain interpretation of its negation is expected. Following Galton's (1990) notation, we shall use the operator "$\neg$" for reified-predicate negation, distincted from fluent negation that is symbolized by "not."

(F1) $Holds(f_1 \vee f_2, t) \Leftrightarrow Holds(f_1, t) \vee Holds(f_1, t)$
that is, fluent f_1 or fluent f_2 holds true over time t if and only if one of them holds true over time t.

With respect to time points, the relationship between negation of reified-predicate and the corresponding fluent negation is simply characterized by:

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

(F2) $Dur(t) = 0 \Rightarrow \neg Holds(f, t) \Leftrightarrow Holds(\text{not}(f), t)$
that is, a fluent does not hold true at a time point if and only if its negation holds true at that point.

However, with respect to decomposable intervals, the interpretation of negation becomes complicated. In fact, by allowing intervals as arguments of the reified-predicate *Holds*, we will face the possibility that a fluent *f* might holds neither true nor false throughout some interval *t*. That is, it may be the case that fluent *f* holds true over some parts of *t* but also holds false over some other parts. As pointed out by Shoham (1987), and also by Allen and Ferguson (1994), there are two ways we might interpret the negative reified-formula $\neg Holds(f, t)$. In the weak interpretation, $\neg Holds(f, t)$ is true if and only if it is not the case that *f* holds true throughout *t*, and hence $\neg Holds(f, t)$ is true if *f* changes its truth value over time *t*. In the strong interpretation, $\neg Holds(f, t)$ is true if and only if *f* holds false throughout *t*, so neither $Holds(f, t)$ nor $\neg Holds(f, t)$ would be true in the case that fluent *f* holds true over some parts of *t* and also holds false some other parts.

In this paper, we shall take the weak interpretation because it seems appropriate for the standard definition of implication and preserves a simple two-valued logic (Allen & Ferguson, 1994). We use the following axiom to characterize the relation between the truth of a fluent *f* over a decomposable interval *t* and its truth over any proper parts of *t*:

(F3) $t = t_1 \oplus t_2 \Rightarrow Holds(f, t) \Leftrightarrow \forall\, t'(In(t', t) \Rightarrow Holds(f, t'))$

In addition, we impose that:

(F4) $Holds(f, t_1) \wedge Holds(f, t_2) \wedge Meets(t_1, t_2) \Rightarrow Holds(f, t_1 \oplus t_2)$
that is, if a fluent holds true over two adjacent time elements respectively, then it also holds true over their ordered union.

The state of the world can be described in terms of a collection of fluents. We shall use $Belongs(f, s)$ to denote fluent *f* belongs to state *s*. The predicate *Belongs* is formally characterized by:

(F6) $\exists s \forall f(\neg Belongs(f, s))$
that is, there exists a state which is an empty set.

(F7) $\exists s \forall f(\neg Belongs(f, s) \vee \neg Belongs(\text{not}(f), s))$
that is, any state cannot contain both a fluent and its negation.

(F8) $\forall s_1, f_1(\neg Belongs(\text{not}(f), s) \Rightarrow \exists s_2 \forall f_2(Belongs(f_2, s_2) \Leftrightarrow (Belongs(f_2, s_1) \vee f_1 = f_2)))$
that is, any fluent can be added to an existing state to form a new state, as long as the state does not contain the negation of the fluent.

(F9) $s_1 = s_2 \Leftrightarrow \forall f(Belongs(f, s_1) \Leftrightarrow Belongs(f, s_2))$
that is, two states are equal if and only if they contain the same fluents.

In what follows, for the reason of simple expression, if $f_1, \ldots, f_n$ are all the fluents that belong to state *s*, we shall represent *s* as $\langle f_1, \ldots, f_n \rangle$; and also, without confusion, we shall use $Holds(s, t)$ to denote that *s* is the state of the world with respect to time *t*, provided that:

(F10) $Holds(s, t) \Leftrightarrow \forall f(Belongs(f, s) \Rightarrow Holds(f, t))$

For the convenience of expression, we shall use $Substate(s_1, s_2)$ to denote that each fluent belonging to states s_1 also belongs to state s_2:

(F11) $Substate(s_1, s_2) \Leftrightarrow \forall f(Belongs(f, s_1) \Rightarrow Belongs(f, s_2))$

By (F10) and (F11), it is straightforward to infer that:

(Th1) $Substate(s_1, s_2) \wedge Holds(s_2, t) \Rightarrow Holds(s_1, t)$

In addition, we introduce two binary functions, *Union* and *Difference*, over states, so that $Union(s_1, s_2)$ and $Difference(s_1, s_2)$ denote the *union* of state s_1 and state s_2, and the *difference* of state s_1 and state s_2, respectively:

(F12) $Belongs(f, Union(s_1, s_2)) \Leftrightarrow Belongs(f, s_1) \vee Belongs(f, s_2)$

(F13) $Belongs(f, Difference(s_1, s_2)) \Leftrightarrow Belongs(f, s_1) \wedge \neg Belongs(f, s_2)$

In fact, in terms of the terminology of set theory, we may simply take a state as a subset of **F**.

A state *s* is called a *complete state* if and only if, under the Domain Constraints, for any property *p*, either *p* itself, or the negation of *p* (i.e., not(*p*)), can be implied by the fluents belonging to *s*, that is:

$$s \models_{\mathbf{D}} p \quad \text{or} \quad s \models_{\mathbf{D}} \text{not}(p),$$

where D denotes the Domain Constraints that specify the world under consideration. Otherwise, *s* is called an *incomplete state*.

A complete state *s* is called a *minimal complete state* if and only if for any fluent *f* belonging to *s*, $Difference(s_1, \langle f \rangle)$ is not a completed state.

An incomplete state gives a partial description of the world. That is, information about some properties is absent. In fact, for most applications, because the world under consideration (e.g., the universe) is too large for complete description (McCarthy & Hayes, 1969), we shall usually just have some partial knowledge about it. Therefore, in many cases, the world will be described in terms of incomplete states with respect to various times.

2.4. Actions and state transitions

An action (type) is in fact some certain phenomenon whose performance over certain time elements may effect state transitions. In this paper, we assume that the world under con-

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

sideration persists in a given state until an action(s) takes place over some specified time to effect a transition of it into another state. An action may either be instantaneous or durative, depending on the temporal duration it lasts. Instantaneous actions take place at time points, while durative actions take place over time intervals. Similar to Allen's approach (Allen, 1984), we shall use $TakesPlace(a, t)$ to represent that action a (type) takes place at/over time t, and impose the following axiom which states that if an action takes place over time t, then it does not take place over/at any proper part of t:

(E1) $TakesPlace(a, t) \Rightarrow \forall t'(In(t', t) \Rightarrow \neg TakesPlace(a, t'))$.

It is easy to infer from axiom (E1) that, if action a takes place over time t, then there is no proper super-interval of t over which a takes place. However, axiom (E1) dose not forbid action a to take place over various times as long as these time elements have no overlapping parts. In other words, the same action type may take place over some time, and then take place again at some later times.

For the sake of keeping expression simpler, we shall use the abbreviated notation $TakesPlaceOnly(a, t)$ to represent that only action a (rather than any other action types) takes place over time t:

(E2) $TakesPlaceOnly(e, t)$

$$\Leftrightarrow TakesPlace(e, t) \wedge \forall t' \forall e'(Sub(t', t) \wedge e' \neq e$$

$$\Rightarrow \neg TakesPlace(e', t')).$$

To represent high-level causal relationships between actions and their effects in the form of state transitions, we introduce formula $Changes(a, s_1, s_2)$, denoting that, starting from state s_1, the performance of action a will effect the transition of the world into state s_2, provided that there is no other action that takes place meanwhile:

(E3) $Changes(a, s_1, s_2) \Rightarrow \forall t_1, t(Holds(s_1, t_1) \wedge TakesPlaceOnly(a, t) \wedge Meets(t_1, t) \Rightarrow \exists t_2(Meets(t, t_2) \wedge Holds(s_2, t_2)))$.

In what follows, we shall call each formula $Changes(a, s_1, s_2)$ a *causal law*, and call s_1 and s_2 the *initial state* and the *result state* of action a, respectively.

It has been noted that there are a number of difficulties in attempts to formalize the static and the dynamic aspects of the world. In particular, there are three classical problems that have remained obstacles to the description of the effects of actions in the framework of situation calculus, as well as in other similar formalisms such as the event calculus.

The first, that is, the *frame problem* (McCarthy & Hayes, 1969), is the problem of indicating and inferring all those things that do not change when actions are performed and time passes. In other words, the frame problem is the problem of specifying everything that is not affected by certain type of actions. The second, that is, the *ramification problem* (Finger, 1987), is closely related to the frame problem. It is the problem of specifying explicitly everything that is affected by certain actions. Finally, the *qualification problem* (McCarthy, 1977) is the problem of specifying all the preconditions for each action.

Several approaches have been proposed to deal with these problems ever since they were first pointed out. For instance, McCarthy (1980, 1986) introduced the technique of *circumscription* to handle the qualification problem and the frame problem. Generally speaking, solutions to these problems require some certain capability of nonmonotonic reasoning, which consists of being able to reason upon assumptions, that is, making inferences based not only on known facts, but also on assumed facts (Vila, 1994).

In this paper, we shall not really tackle the qualification problems. We simply assume that, whenever $TakesPlace(a, t)$ is asserted as true, all the preconditions of action a are satisfied over time t. Because the preconditions of an action a, denoted as $PreCond(a)$, can be in fact expressed as a collection of fluents, which, without confusion, may be taken as a special state of the world, we impose the following constraint as for the performance of action a:

(E4) $TakesPlace(a, t) \Rightarrow Holds(PreCond(a), t)$.

In what follows, we provide a simple treatment to the corresponding *frame problem*. Without presuming to solve these two problems in their full generality, this treatment is practical and effective for most industrial real-time applications, although it may be too strong from the view of theoretical point and actually lead to the *ramification problem*. The following axiom, (E5), which we shall call the *effect completion axiom*, guarantees that all fluents affected by an action are governed by the correspondingly "minimal" causal law; in other worlds, each causal law completely characterizes the effects of the performing the action being addressed.

(E5) $Changes(a, s_1, s_2) \wedge Substate(s_1, s_1') \Rightarrow Changes(a, s_1', Union(Difference(s_1', s_1), s_2)))$.

Here, what we mean by "a causal law, say $Changes(a, s_1, s_2)$, is the minimal" is: with respect to event a, s_1 is the smallest initial state under the *Substate* relation.

3. EXAMPLES

As discussed in the introduction, modelling the static and the dynamic aspects of the world usually involves representing and reasoning about states, actions, and changes, where the notion of time plays an important role. The formalism proposed in this paper combines the advantages of

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

temporal logics [e.g., McDermott (1982) and Allen (1984)] by the way of allowing explicit time expression and the benefit of state-based approached such as the situation calculus that can easily represent the world remains static except when the agent acts, and where nothing important happens while actions are being executed (Allen & Ferguson, 1994). In what follows, we shall demonstrate two examples as the applications of the new formalism.

3.1. The blocks world

The blocks world is a typical example, especially in artificial intelligence, which has been used to illustrate the application of various formalisms. Consider the following version of the blocks world in which there are only three blocks, A, B, and C. Each block can be either on the table or immediately on the top of exactly one of the other blocks. Also, each block can have at most one of the other blocks immediately on top of it. We may use the following three kinds of fluents to describe the state of these three-block world:

$Clear(x)$: there is no blocks on the top of block x;

$OnTable(x)$: block x is on the table;

$On(x, y)$: block x is on the top of block y;

and therefore the Domain Constraints for the three-block world are:

(3.1.1) $Clear(x) \Rightarrow x = \mathrm{A} \vee x = \mathrm{B} \vee x = \mathrm{C}$
(3.1.2) $OnTable(x) \Rightarrow x = \mathrm{A} \vee x = \mathrm{B} \vee x = \mathrm{C}$
(3.1.3) $On(x, y) \Rightarrow (x = \mathrm{A} \vee x = \mathrm{B} \vee x = \mathrm{C}) \wedge (y = \mathrm{A} \vee y = \mathrm{B} \vee y = \mathrm{C})$
(3.1.4) $OnTable(x)\ \nabla\ \exists y(On(x, y))$
(3.1.5) $On(x, y) \Rightarrow not(x = y) \wedge not(Clear(y))$
(3.1.6) $On(x, y) \wedge On(x, z) \Rightarrow y = z$
(3.1.7) $On(x, y) \wedge On(z, y) \Rightarrow x = z$,

where ∇ denotes "exclusive or."

In the above, (3.1.1)–(3.1.3) preserve that there are only three blocks, A, B, and C. (3.1.4) guarantees that each block is either on the table or on the top of one of the blocks, but cannot on both. (3.1.5) states that a block cannot be on itself, and a block has another block on its top, then its top is not clear. Finally, (3.1.6) and (3.1.7) guarantee that each block can be immediately on the top of at most one of the other blocks, and each block can have at most one of the other blocks immediately on top of it, respectively. There are in total 13 possible **complete states**, which correspond to different configuration of blocks as shown in Figure 1.

These 13 complete states may be described as the following collections of fluents:

$S_1 = \langle Clear(\mathrm{C}), OnTable(\mathrm{B}), On(\mathrm{A}, \mathrm{B}), On(\mathrm{C}, \mathrm{A})\rangle$
$S_2 = \langle Clear(\mathrm{B}), OnTable(\mathrm{C}), On(\mathrm{A}, \mathrm{C}), On(\mathrm{B}, \mathrm{A})\rangle$
$S_3 = \langle Clear(\mathrm{C}), OnTable(\mathrm{A}), On(\mathrm{B}, \mathrm{A}), On(\mathrm{C}, \mathrm{B})\rangle$
$S_4 = \langle Clear(\mathrm{A}), Clear(\mathrm{C}), OnTable(\mathrm{B}), OnTable(\mathrm{C}), On(\mathrm{A}, \mathrm{B})\rangle$
$S_5 = \langle Clear(\mathrm{A}), Clear(\mathrm{B}), OnTable(\mathrm{B}), OnTable(\mathrm{C}), On(\mathrm{A}, \mathrm{C})\rangle$
$S_6 = \langle Clear(\mathrm{B}), Clear(\mathrm{C}), OnTable(\mathrm{A}), OnTable(\mathrm{C}), On(\mathrm{B}, \mathrm{A})\rangle$
$S_7 = \langle Clear(\mathrm{A}), Clear(\mathrm{B}), Clear(\mathrm{C}), OnTable(\mathrm{A}), OnTable(\mathrm{B}), OnTable(\mathrm{C})\rangle$
$S_8 = \langle Clear(\mathrm{A}), Clear(\mathrm{B}), OnTable(\mathrm{A}), OnTable(\mathrm{C}), On(\mathrm{B}, \mathrm{C})\rangle$
$S_9 = \langle Clear(\mathrm{C}), Clear(\mathrm{B}), OnTable(\mathrm{A}), OnTable(\mathrm{B}), On(\mathrm{C}, \mathrm{A})\rangle$
$S_{10} = \langle Clear(\mathrm{A}), Clear(\mathrm{C}), OnTable(\mathrm{A}), OnTable(\mathrm{B}), On(\mathrm{C}, \mathrm{B})\rangle$
$S_{11} = \langle Clear(\mathrm{C}), OnTable(\mathrm{C}), On(\mathrm{B}, \mathrm{C}), On(\mathrm{A}, \mathrm{B})\rangle$
$S_{12} = \langle Clear(\mathrm{B}), OnTable(\mathrm{A}), On(\mathrm{C}, \mathrm{A}), On(\mathrm{B}, \mathrm{C})\rangle$
$S_{13} = \langle Clear(\mathrm{A}), OnTable(\mathrm{B}), On(\mathrm{C}, \mathrm{B}), On(\mathrm{A}, \mathrm{C})\rangle$.

To check if a state is a minimal one under the domain constraints is quite complicated. However, if we add some extra constraints, we may reduce the size of the above complete states. For instance, if $\forall y(\mathrm{not}(On(y, x))) \Rightarrow Clear(x)$ were added, we could get rid of $Clear(\mathrm{C})$ from S_1, and get rid of both $Clear(\mathrm{A})$ and $Clear(\mathrm{C})$ from S_4, and so on, where the reduced states would be still complete.

The three action types that may take place with respect to the three-block world are:

$M(x, y, z)$: moving block x from the top of y to the top of z.

$U(x, y)$: unstacking block x from the top of y and placing it on the table.

$S(x, y)$: picking up block x from the table and stacking it on the top of block y.

Normally, there are some preconditions for the performances of these actions. For instance, to ensure the action of type $M(x, y, z)$ to take place, fluents $Clear(x)$, $Clear(z)$, and $On(x, y)$ must hold. For most applications, the preconditions for some action types are usually included within the initial states of the corresponding actions with respect to involved causal laws. Specially, as for three-block world, we have the following causal laws:

$Changes(M(x, y, z), \langle Clear(x), Clear(z), On(x, y)\rangle, \langle Clear(x), Clear(y), On(x, z)\rangle)$,
$Changes(U(x, y), \langle Clear(x), On(x, y)\rangle, \langle Clear(x), Clear(y), OnTable(x)\rangle)$,
$Changes(S(x, y), \langle Clear(x), Clear(y), OnTable(x)\rangle, \langle Clear(x), On(x, y)\rangle)$.

By the *effect completion axiom* (E5), we can apply these causal laws among the above 13 states $S_1, S_2, \ldots, S_{13}$, and get:

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

Changes(S(A, B), S_7, S_4), *Changes*(S(A, B), S_8, S_{11}), *Changes*(S(A, C), S_7, S_5),

Changes(S(A, C), S_{10}, S_{13}), *Changes*(S(B, A), S_7, S_6), *Changes*(S(B, A), S_5, S_2),

Changes(S(B, C), S_7, S_8), *Changes*(S(B, C), S_9, S_{12}), *Changes*(S(C, A), S_7, S_9),

Changes(S(C, A), S_4, S_1), *Changes*(S(C, B), S_7, S_{10}), *Changes*(S(C, B), S_6, S_3),

Changes(U(A, B), S_4, S_7), *Changes*(U(A, B), S_{11}, S_8), *Changes*(U(A, C), S_5, S_7),

Changes(U(A, C), S_{13}, S_{10}), *Changes*(U(B, A), S_6, S_7), *Changes*(U(B, A), S_2, S_5),

Changes(U(B, C), S_8, S_7), *Changes*(U(B, C), S_{12}, S_9), *Changes*(U(C, A), S_9, S_7),

Changes(U(C, A), S_1, S_4), *Changes*(U(C, B), S_{10}, S_7), *Changes*(U(C, B), S_3, S_6),

Changes(M(A,B,C), S_4, S_5), *Changes*(M(A,C,B), S_5, S_4), *Changes*(M(B,A,C), S_6, S_8),

Changes(M(B,C,A), S_8, S_6), *Changes*(M(C,A,B), S_9, S_{10}), *Changes*(M(C,B,A), S_{10}, S_9).

These state transitions are shown in Figure 2.

In the above, the causal laws only represent high-level state transitions within the blocks world that do not involve explicit temporal references. However, the formalism proposed in Section 2 allows low-level temporal representation and reasoning. In fact, suppose that the world is initially in state S_3 over time T_0, that is,

(3.1a) *Holds* (S_3, T_0)

The objective is to find a plan of actions to change the world into state S_{12}. For the reasoning of simple demonstration, we assume that each action takes the same temporal duration to take place. It is easy to see that the first action may take place is U(C, B):

(3.1b) *TakesPlace*(U(C, B), T_1) $\wedge$ *Meets*(T_0, T_1)

Therefore, by *Changes*(U(C, B), S_3, S_6) and axiom (E3), there will be a time element T_2, such that

(3.1c) *Holds*(S_6, T_2) $\wedge$ *Meets*(T_1, T_2).

From state S_6, there are two actions that can be performed: U(B, A) and M(B, A, C). The more efficient one, however, is U(C, B):

(3.1d) *TakesPlace*(U(B, A), T_3) $\wedge$ *Meets* (T_2, T_3)

Again, by *Changes*(U(B, A), S_6, S_7) and axiom (E3), we get:

(3.1e) *Holds*(S_7, T_4) $\wedge$ *Meets*(T_3, T_4).

Now, from state S_7, there are six actions that can be performed: S(A, B), S(A, C), S(B, A), S(B, C), S(C, A), and S(C, B). With respect to the goal, that is state S_{12}, the most appropriate one is S(C, A):

(3.1f) *TakesPlace*(S(C, A), T_5) $\wedge$ *Meets* (T_4, T_5).

This time, by *Changes*(S(C, A), S_7, S_9) and axiom (E3), we get:

(3.1g) *Holds*(S_9, T_6) $\wedge$ *Meets*(T_5, T_6).

Finally, from state S_9, there are two actions that can be performed: S(B, C) and M(C, A, B), and the suitable one is S(B, C):

(3.1h) *TakesPlace*(S(B, C), T_7) $\wedge$ *Meets*(T_6, T_7)

and therefore, by *Changes*(S(B, C), S_9, S_{12}) and axiom (E3), we reach the goal:

(3.1j) *Holds*(S_{12}, T_8) $\wedge$ *Meets*(T_7, T_8).

It is important to note that, the formalism proposed here allows a certain state of the world to hold repeatedly over/at different times. For instance, suppose the objective is to change the world into state S_{12} in the first place, and then change it into state S_{10}. In this case, from state S_{12}, after the performance of action U(B, C), the world will be changed into state S_9 again:

(3.1k) *TakesPlace*(U(B, C), T_9) $\wedge$ *Meets*(T_8, T_9).

(3.1l) *Holds*(S_9, T_{10}) $\wedge$ *Meets*(T_9, T_{10}).

Therefore, by (3.1g) and (3.1l), we can see that state S_9 holds repeatedly over time elements T_6 and T_{10}.

3.2. The tea maker system

Now, consider the example of the automatic tea maker as described in the introduction. We shall use the following 7 fluents to describe various states of the system:

F_1: the tea maker is clean;

F_2: the switch is at “on” position;

F_3: there is some water inside the tea maker;

F_4: the water is boiling (i.e., the temperature of the water is higher than or equal to 100°C);

F_5: there is a tea bag(s) inside the tea maker;

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

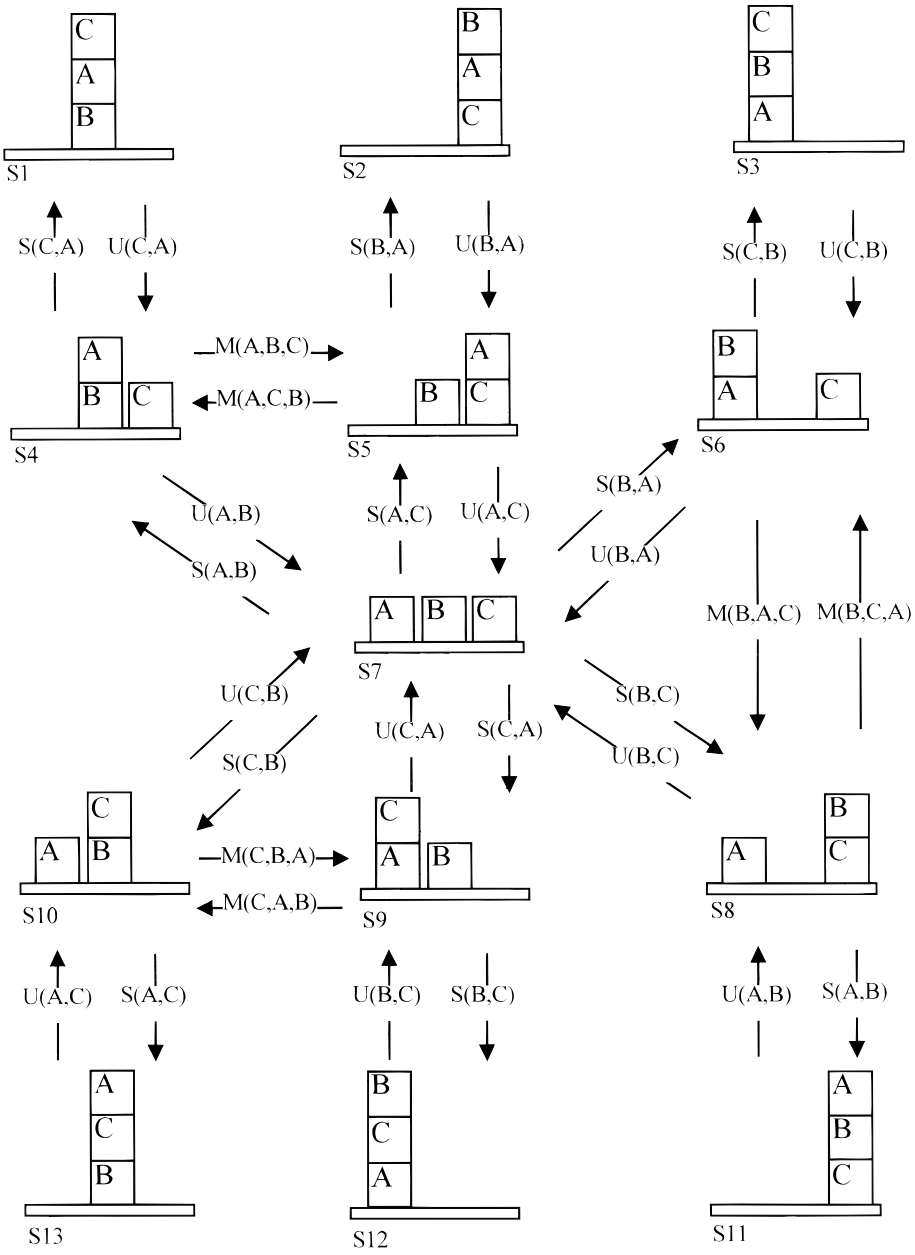


Fig. 2. State transitions.

F_6: the tea is ready (e.g., the temperature of the tea is lower than 100°C but higher than or equal to 60°C); and

F_7: the tea is too cold (e.g., the temperature of the tea is lower than 60°C).

Therefore, those 10 static states of the system as described in Section 1 can be denoted as:

$S_1 = \{F_1, \text{not}(F_2), \text{not}(F_3), \text{not}(F_5)\}$, or namely, the "Clean_Off_NoWater_NoTeaBag" state;

$S_2 = \{\text{not}(F_2), \text{not}(F_4), \text{not}(F_5)\}$, or namely, the "Off_WaterNotBoiling_NoTeaBag" state;

$S_3 = \{F_2, \text{not}(F_4), \text{not}(F_5)\}$, or namely, the "On_WaterNotBoiling_NoTeaBag" state;

$S_4 = \{F_2, F_4, \text{not}(F_5)\}$, or namely, the "On_WaterBoiling_NoTeaBag" state;

$S_5 = \{F_2, F_4, F_5\}$, or namely, the "On_WaterBoiling_TeaBag" state;

$S_6 = \{\text{not}(F_2), F_4, F_5\}$, or namely, the "Off_WaterBoiling_TeaBag" state;

$S_7 = \{\text{not}(F_2), F_6, F_5\}$, or namely, the "Off_TeaReady_TeaBag" state;

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

S_8 = {not(F_2), F_7, F_5}, or namely, the "Off_TeaCold_TeaBag" state;

S_9 = {F_2, F_7, F_5}, or namely, the "On_TeaCold_TeaBag" state;

S_{10} = {not(F_1), not(F_2), not(F_3), F_5}, or namely, the "Dirty_Off_NoWater_TeaBag" state.

Because there are no domain constraints, all the above states are incomplete with respect to the collection of fluents, that is, F_1, F_2, …, F_7.

There are in total 10 action types that may take place with respect to the tea maker system. That is, AddingWater, SwitchingOn, Heating1, Heating2, AddingTeaBag, SwitchingOff, Cooling1, Cooling2, Pouring, and Cleaning. The corresponding causal laws are as below:

Changes(AddingWater, ⟨F_1, not(F_3)⟩, ⟨not(F_4)⟩),
Changes(SwitchingOn, ⟨not(F_2)⟩, ⟨F_2⟩),
Changes(Heating1, ⟨not(F_4)⟩, ⟨F_4⟩),
Changes(Heating2, ⟨F_7⟩, ⟨F_4⟩),
Changes(AddingTeaBag, ⟨not(F_5⟩, ⟨F_5⟩),
Changes(SwitchingOff, ⟨F_2⟩, ⟨not(F_2)⟩),
Changes(Cooling1, ⟨F_4⟩, ⟨F_6⟩),
Changes(Cooling2, ⟨F_6⟩, ⟨F_7⟩),
Changes(Pouring, ⟨F_6⟩, ⟨not(F_1), not(F_3)⟩),
Changes(Cleaning, ⟨not(F_1), F_5⟩, ⟨F_1, not(F_5)⟩).

By the *effect completion axiom* (E5), we can apply these causal laws to states S_1, S_2, …, S_{10}, and get:

Changes(AddingWater, S_1, S_2), *Changes*(SwitchingOn, S_2, S_3),
Changes(Heating1, S_3, S_4), *Changes*(AddingTeaBag, S_4, S_5),
Changes(Heating2, S_9, S_5), *Changes*(SwitchingOff, S_5, S_6),
Changes(Cooling1, S_6, S_7), *Changes*(Cooling2, S_7, S_8),
Changes(Pouring, S_7, S_{10}), *Changes*(Cleaning, S_{10}, S_1)

Both the static and dynamic images of such a tea maker system can be represented as the state-transition cycle shown in Figure 3.

4. CONCLUSION

Temporal logic is by no means the only formalism for handling time that has been around in computer science. Petri nets and temporal constraint-satisfaction networks spring to one's mind in this connection. However, temporal logic is a powerful formalism in that it allows to incorporate inference mechanisms about temporal relations. Such expressive power comes at a cost, though: representation requires one to master a logic formalism rather than just a language. Nevertheless, every tool requires learning how to use it. In this paper, we have been guiding the reader step by step through the rudiments of this tool. We resorted to rather simple examples: the blocks-world universe, and a fairly simple schema representing the operating cycle of an automatic tea maker.

Arguably, temporal logic deserves a wider spread in engineering applications of artificial intelligence. Temporal logic captures more. Whereas developing the representation may be laborious, even for fairly simple applications, nevertheless the reasoning capabilities afforded are an advantage that oftentimes may be worth the effort, with respect to data structure conventions that leave out inference. In terms of costs/benefits, some complex applications admittedly are more straightforwardly served by less ambi-

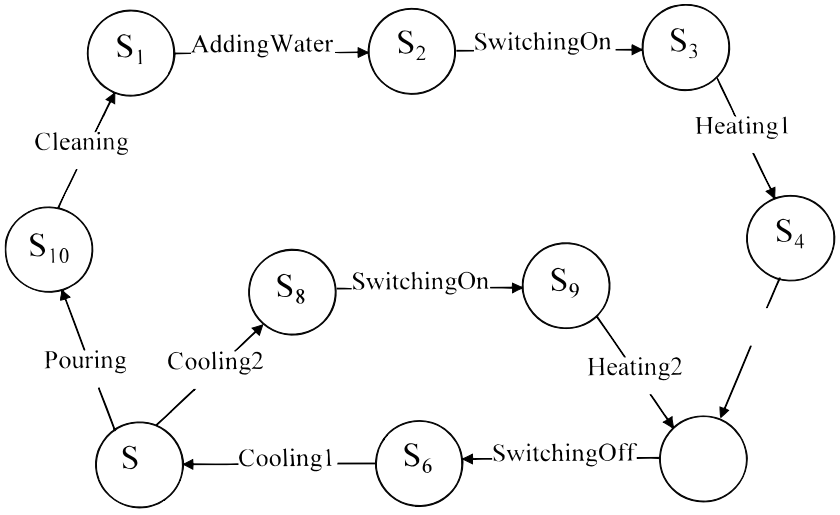


Fig. 3. Static and dynamic images of a tea-maker system represented as the state-transition cycle.

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

tious representations than temporal logic; on the other hand, requirements of flexibility and integration make the incorporation of temporal reasoning capabilities attractive. Within the broad category of logic formalisms specifically devised for handling time and events, there are variants. Advantages have been, as usual, claimed for each. Our own approach is handy in that it allows to reason on both time points and time intervals.

On need, temporal logic could be augmented with further capabilities. For example, either modal operators (in modal logic representation) or predicates (in ontological representations of modals) have been investigated by logicists in computer science for dealing with agency, or, then, with deontic necessity and possibility (i.e., obligation vs. optionality, and prohibition vs. permissibility); in recent years, there has been a trend to merge deontic and agency logic research [e.g., Brown and Carmo (1996)]. Yet, one thing at a time; we cannot deal with the interface to that kind of formalisms within the compass of the present paper. Let it suffice to signal to the reader the possible bonus accruing from the adoption of logic representations, in that a variety of common sense inferential mechanisms can be accommodated under the logician's umbrella. Several of these mechanisms are of interest for application in engineering; none, however, is of as immediate use as temporal reasoning.

REFERENCES

Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence 23*, 123–154.

Allen, J., & Ferguson, G. (1994). Actions and events in interval temporal logic. *Journal of Logic and Computation 4(5)*, 531–579.

Finger, J. (1987). *Exploiting constraints in design synthesis*. PhD thesis, Stanford University, Stanford, CA.

Galton, A. (1990). A Critical examination of Allen's Theory of action and time. *Artificial Intelligence 42*, 159–188.

Gelford, M., Lifschitz, V., & Rabinov, A. (1991). What are the limitations of the situation calculus? *Working Notes of AAAI Spring Symposium Series. Symposium: Logical Formalization of Commonsense Reasoning*, 59–69.

Genesereth, M., & Nilsson, N. (1987). *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers, San Francisco, CA.

Gooday, J., & Galton, A. (1996). The transition calculus: A high-level formalism for reasoning about action and change. *Journal of Experimental and Theoretical Artificial Intelligence 9*, 67–95.

Kowalski, R., & Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing 4*, 67–95.

Lin, F., & Shoham, Y. (1991). Provably correct theories of action. *Proc. Ninth National Conference on Artificial Intelligence (AAAI-91)*, 349–354.

Ma, J., & Knight, B. (1994). A general temporal theory. *The Computer Journal 37(2)*, 114–123.

Ma, J. & Knight, B. (1996). A reified temporal logic. *The Computer Journal 39(9)*, 800–807.

McDermott, D. (1982). Temporal logic for reasoning about processes and plans. *Cognitive Science 6*, 101–155.

McCarthy, J. (1977). Epistemological problems of artificial intelligence. *Proc. Fifth International Joint Conference on Artificial Intelligence*, 1038–1044.

McCarthy, J. (1980). Circumscription: A form of non-monotonic reasoning. *Artificial Intelligence 13*, 27–39.

McCarthy, J. (1986). Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence 28*, 27–39.

McCarthy, J., & Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence* (Meltzer, B., & Michie, D., Eds.), Vol. 4, pp. 463–502, Edinburgh University Press, Scotland.

Miller, R.S., & Shanahan, M.P. (1994). Narratives in the situation calculus. *The Journal of Logic and Computation 4(5)*, 513–530.

Pinto, J., & Reiter, R. (1995). Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, Special Festschrift Issue in honour of Jack Minker *14(2–4)*, 251–268.

Sandewall, E., & Ronnquist, R. (1986). A representation of action structures. *Proc. Fifth National Conference on Artificial Intelligence (AAAI-86)*, 89–97.

Shoham, Y. (1987). Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence 33*, 89–104.

Vila, L. (1994). A survey on temporal reasoning in artificial intelligence. *AI Communication 7(1)*, 4–28.

Brian Knight leads the artificial intelligence team, heads the Computer Science group, and is a co-director of the Centre for Numerical Modelling and Process Analysis at the School of Computing and Mathematical Sciences of the University of Greenwich, in London. Born in London, Prof. Knight was educated at ChristChurch, Oxford, where he was mathematics scholar. He graduated in mathematics at Oxford, and completed a doctoral degree at King's College, London for wok in mathematical physics. He has worked as a project engineer and consultant in several research and development projects in the computing industry. He was employed by IBM as a systems engineer, and as a consultant for the Ministry of Defence, where he designed the central algorithms for a large-scale scientific project. He has also worked as a consultant in the process control industry, where he designed and built network software for process control applications and designed software for North sea oil platform alarm systems. The author of about 100 research papers, mainly in artificial intelligence, Prof. Knight's academic work has centered on scientific software engineering and the applications of AI in the industrial sphere. He has led many research projects developing AI in the engineering field (aluminium furnaces, fire safety, minerals processing, metal foundries, among these). Once appointed as reader in scientific computing at the University of Greenwich, he was sponsored by the Water Research Centre for several years, and conducted research work in the applications of expert systems to water engineering. His current research interests include software engineering, temporal databases, and the theory and application of temporal reasoning, as well as the application of AI techniques.

Jixin Ma is a senior lecturer at the School of Computing and Information Systems of the University of Greenwich, where he earned his Ph.D. in 1994, under the supervision of Prof. Brian Knight. Dr. Ma's main areas of research are temporal logic and temporal databases. His research interests are in artificial intelligence, as well as in mathematical logic, group theory, and linear algebra. A theoretician, his projects and publications are concerned with basic research as well as with its application into working systems, that is, temporal manage-

https://doi.org/10.1017/S0890060499132025 Published online by Cambridge University Press

ment for process control. Both directions are pursued thoroughly in his works, on which he has published extensively. He holds an M.Sc. degree in mathematics (1987) and a degree in applied mathematics (1982) from Zheng Zhou University and Zheng Zhou Institute of Technology, China, respectively. Before his current appointment at the University of Greenwich, he has been affiliated there as a lecturer in software engineering, and in mathematics for computer science. In 1988–1991 Dr. Jixin Ma had also been a lecturer in mathematics in China for 6 years and a half before he came to the United Kingdom. Problems of temporal logic he is currently investigating include: the integration of the point-based, absolute-time approach and interval-based relative-time temporal models; the inference mechanism for retrieval of temporal information, with deep reasoning on relative temporal data in query evaluation; integration with other kinds of reasoning (e.g., reasoning on actions).

Ephraim Nissan is a research fellow at the School of Computing and Mathematical Sciences of the University of Greenwich, in London. He holds the Laurea in electronics engineering from the Milan Institute of Technology. His second thesis discussed there won a nationwide contest in Italy. Afterwards, he earned a Ph.D. degree in computer science from the Ben-Gurion University of the Negev, and won an IPA Award in Computer Science for the doctoral project, on an expert system for computational linguistics (cf. a 50,000-word entry on word-formation now in press in the *Encyclopedia of Computer Science and Technology)*. Listed in the 4th edition of *Who's Who in Science and Engineering*, he was formerly affiliated with three schools of Bar-Ilan University (Israel), and also was a visiting professor (1993) at the Instituto Metodologico Economico Statistico of the University of Urbino (Italy), where he is an overseas associate. He published over 100 papers, has been or is a guest editor for several international journals, and co-founed one. He co-edited a paper collection in humanities computing, *From Information to Knowledge*, with K.M. Schmidt ; Oxford: Intellect, 1995). Later on, he coordinated the "1998 Forum" of *The New Review of Applied Expert Systems*, on refuelling in nuclear engineering; for the same journal; he is also co-coordinator of the "1999 Forum." Moreover, he edited a special issue on "Intelligent Technologies for Electric and Nuclear Power Systems" in *Computers and Artificial Intelligence*, Vol. 17, No. 2-3, 1998, and an issue on educational hypermedia in the *Journal of Educational Computing Research*, Vol. 17, No. 3, 1997. Along with A.A. Martino, he edited thematic issues, "Formal Models of Legal Time" for the legal computing journals: *Law, Computers and Artificial Intelligence* and *Information and Communications Technology Law*, Vol. 7, No. 3, 1988, and, in an attempt to establish a new subdiscipline within AI & Law research, he edited a multiple special issue, "Formal Approaches to Legal Evidence" in Vol. 8 of *Artificial Intelligence and Law* (to appear).