Handbook of Knowledge Representation
Edited by F. van Harmelen, V. Lifschitz and B. Porter

DOI: 10.1016/S1574-6526(07)03012-X

Chapter 12

Temporal Representation and Reasoning

Michael Fisher

This book is about representing *knowledge* in all its various forms. Yet, whatever phenomenon we aim to represent, be it natural, computational, or abstract, it is unlikely to be static. The natural world is always decaying or evolving. Thus, computational processes, by their nature, are dynamic, and most abstract notions, if they are to be useful, are likely to incorporate change. Consequently, the notion of representations *changing through time* is vital. And so, we need a clear way of representing both our temporal basis, and the way in which entities change over time. This is exactly what this chapter is about.

We aim to provide the reader with an overview of many of the ways temporal phenomena can be *modelled*, *described*, *reasoned about*, and *applied*. In this, we will often overlap with other chapters in this collection. Some of these topics we will refer to very little, as they will be covered directly by other chapters, for example, *temporal action logic* [84], *situation calculus* [185], *event calculus* [209], *spatio-temporal reasoning* [74], *temporal constraint satisfaction* [291], *temporal planning* [84, 271], and *qualitative temporal reasoning* [102]. Other topics will be described in this chapter, but overlap with descriptions in other chapters, in particular:

- *automated reasoning*, in Section 12.3.2 and in [290];
- *description logics*, in Section 12.4.6 and in [154]; and
- *natural language*, in Section 12.4.1 and in [250].

The topics in several other chapters, such as *reasoning about knowledge and belief* [203], *query answering* [34] and *multi-agent systems* [277], will only be referred to very briefly.

Although this chapter is not intended to be a comprehensive survey of *all* approaches to temporal representation and reasoning, it does outline many of the most prominent ones, though necessarily at a high-level. If more detail is required, many references are provided. Indeed, the first volume of the *Foundations of Artificial Intel-*

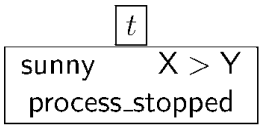


Figure 12.1: State at time t.

ligence series, in which this collection appears, contains much more detail on the use of temporal reasoning in Artificial Intelligence [100] while [112, 56, 129, 114, 148] all provide an alternative logic-based view of temporal logics. In addition, there are many, more detailed, survey papers which we refer to throughout.

The structure of this chapter is as follows. We begin, in Section 12.1, by considering structures for modelling different aspects of time, aiming at providing an overview of many alternatives. In Section 12.2, we discuss languages for talking about such temporal representations and their properties. Typically, these languages are forms of *temporal logic*. Section 12.3 addresses the problem of reasoning about descriptions given in these temporal languages and highlights a number of significant techniques. In order to provide further context for this discussion, Section 12.4 outlines a selection of application areas for temporal representation and reasoning. Finally, in Section 12.5, concluding remarks are provided.

12.1 Temporal Structures

While we will not enter into a philosophical discussion about the nature of time itself (see, for example, [287, 119]), we will examine a variety of different structures that underlie representations of time. Where possible, we will provide mathematical descriptions in order to make the discussions more formal.

We are only able to describe temporal concepts if we are able to refer to a particular time and so relate different times to this. Without prejudicing later decisions, we will describe such times as *states* and will refer to each one via an unique index. Thus, at a particular time, say t, we can describe facts such as "it is sunny", "the process is stopped", and "X is bigger than Y". For example, in Fig. 12.1 we have one such state, t.

Now, as soon as we go beyond this simple view, we face a number of choices, all of which can significantly affect the complexity and applicability of the temporal representation.

12.1.1 Instants and Durations

It may seem as though the index t described above naturally represents an instant in time. Indeed, by describing t as a *state*, we have already implied this. While this is a popular view, it is not the only one. Another approach is to consider t as ranging over a set of temporal *intervals*. An interval is a sequence of time with duration. Thus, if t now refers to an interval, for example, an hour, then Fig. 12.1 represents properties true during that hour: "it is sunny throughout that hour", "the process is stopped in that hour", and "X is bigger than Y for an hour". It is important to note that the language we use to describe properties is vital. Thus, we have just used "throughout", "in", and

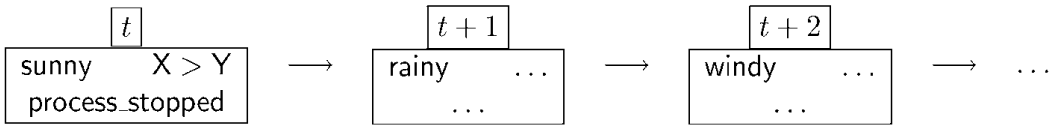


Figure 12.2: Organising states as $\mathbb{N}$.

"for" in describing properties holding over intervals. The differences that such choices make will be considered in more detail in Section 12.2.5. We have also referred to *explicit* times, such as one hour; again, the possibility of talking directly about real values of time will be explored in Section 12.2.6.

Related to the question of whether points or intervals should be used as the basis for temporal representation is the question of whether temporal elements should be *discrete*. If we consider points as the basis for a temporal representation, then it is important to describe the relationship *between* points. An obvious approach is to have each point representing a discrete moment in time, i.e., distinguishably separate from other points. This corresponds to our intuition of 'ticks' of a clock and is so appealing that the most popular propositional temporal logic is based upon this view. This logic, called *Propositional Temporal Logic* (PTL) [113, 223], views time as being isomorphic to the Natural Numbers, with:

- an identifiable start point, characterised by '0';
- discrete time points, characterised by '0', '1', '2', etc.;
- an infinite future; and
- a simple operation for moving from one point ('i') to the *next* (characterised by '$i + 1$').

There are a number of variations of the above properties that we will discuss soon, but let us consider a model for PTL as simply $\langle \mathbb{N}, \pi \rangle$ with π being a function mapping each element of the Natural Numbers, $\mathbb{N}$, to the set of propositions *true* at that moment. We will see later that this is used for the semantics of PTL. We can visualise this as in Fig. 12.2, where π captures the elements inside each temporal element (i.e. all the *true* propositions; those not mentioned are, by default, *false*).

12.1.2 From Discreteness to Density

We next consider some variations on the basic type of model given above. In Section 12.1.4, we re-examine the above assumptions of having an identifiable start state and linearity. For the moment, however, we only review the decision to have a set of *discrete* time points between which we can move via a simple function. Although this corresponds to the Natural Numbers (or Integers), what if we take the Rational Numbers as a basis? Or the Real Numbers? Or, indeed, what if we take a structure that has no analogue in Number Theory?

In general, the model for point-based temporal logic is $\langle S, R, \pi \rangle$, where S is the set of time points, π again maps each point to those propositions true at that point, and R is an earlier–later relation between points in S. In the case of discrete temporal

logics, we can replace the general accessibility relation, R, by a relation between adjacent points, N. This *next-time* relation applies over the set of all discrete moments in time (S). Thus, for all s_1 and s_2 in S, $N(s_1, s_2)$ is true if s_2 is the *next* discrete moment after s_1.

If we go further and use a standard arithmetical structure, we can replace the combination of N and S (or R and S) by the structure itself, e.g., $\mathbb{N}$ with the associated ordering.

Now, if we consider non-discrete structures, such as $\mathbb{R}$, there is no clear notion of the *next* point in time. $\mathbb{R}$ is *dense*, and so if a temporal relation, R, is based on this domain, then if two time points are related, there is always another point that occurs *between* them:

$$\forall i \in S.\, \forall k \in S.\, R(i,k) \quad \Rightarrow \quad [\exists j \in S.\, R(i,j) \wedge R(j,k)].$$

Consequently, the concept of a *next* point in time makes little sense in this context and so logics based on dense models typically use specific operators relating to intervals over the underlying domain; see Section 12.2.4. And so we have almost come full circle: dense temporal logics, such as those based on $\mathbb{R}$, require interval-like operators in their language. (By *interval-like*, we mean operators that refer to particular subsequences of points.)

There is a further aspect that we want to mention and that will become important later once we consider representing point-based temporal logics within classical first-order logic (see Section 12.3.2). As we have seen, some constraints on the accessibility relation (for example, density, above) can be defined using a first-order language over such relations. However, there are some restrictions (for example, finiteness) that cannot be defined in this way [161, 274, 112].

There is much more work in this area, covering a wide variety of base domains for temporal logics. However, we will just mention one further aspect of underlying models of time, namely *granularity*, before moving on to more general organisation within the temporal structure (in Section 12.1.4).

12.1.3 Granularity Hierarchies

The models of time we have seen so far are relatively simple. In mentioning the possibility of an underlying dense domain above we can begin to see some of the complexity; between any two time points there are an infinite number of other time points. Thus, time can be described at arbitrary *granularities*. However, it is often the case that a description is needed at a particular granularity, and only later do we need to consider finer time distinctions. A simple example from practical reasoning concerns a discussion between participants who agree to organise a meeting *every month*. They must agree to either a date, e.g., the 25th, or to a particular day, e.g., the last Tuesday in the month. Later, they will consider times within that day. Then they might possibly consider more detailed times within the meeting itself, and so on. In the first case, the participants wish to represent the possibilities without having to deal with minutes, or even hours. Later, hours, minutes and seconds may be needed. In practical terms such requirements have led to systems such as *calendar logic* [213]. More generally, significant work has been carried out on hierarchies of differing granularities, for example, in [202, 105, 59, 232], with a comprehensive descriptions being given

in [93, 46]. Finally, the work on interval temporal logics described later has also led to alternative views of granularity and projection [206, 130, 58, 131].

12.1.4 Temporal Organisation

In general, the accessibility relation between temporal points is an arbitrary relation. However, as we have seen above, many domains provide additional constraints on this. Typically, the accessibility relation is *irreflexive* and *transitive*. In addition, the use of arithmetical domains, such as $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$, ensures that the temporal structure is both linear and infinite in the future. While a linear model of time is adopted within the most popular approaches [223], there is significant use of the branching (in to the future) model [91, 281], particularly in model checking (see Section 12.4.4). Yet there are many other ways of organising the flow[1] of time, including a *circular* view [239], a *partial-order*, or trace-based, view [163, 218, 139, 268], or an *alternating* view [68, 17]. These last two varieties have been found to be very useful in specific applications, particularly partial-order temporal logics for partial/trace-based requirements specifications, such as Message Sequence Charts or concurrent systems, and alternating-time temporal logics for both the logic of games and the verification of multi-process (and multi-agent, see [277]) systems [18, 14, 200].

All these considerations are closely related to finite automata over infinite strings (ω-automata). There has been a considerable amount of research developing the link between forms of ω-automata (such as Büchi automata) and both temporal and modal logics [254, 279, 280]. It is beyond the scope of this article to delve much into this, yet it is important to recognise that much of the development of (point-based) temporal representation and reasoning is closely related to automata-theoretic counterparts.

12.1.5 Moving in Real Time

So far we have considered the *relative* movement through time, where time is represented by abstract entities organised in structures such as trees or sequences. Even in discrete temporal models, the idea of the *next* moment in time is an abstract one. Each step does not directly correspond to explicit elements of time, such as seconds, days or years. In this section, we will outline the addition of such *real-time* aspects. These allow us to compare times, not just in terms of before/after or earlier/later relations, but also in *quantitative* terms.

Since there are many useful articles on structures for representing real-time temporal properties, such as the influential [12, 13], together with overviews of the work (particularly on timed automata) [15, 19, 44], we will simply give an outline of the *timed automata* approach on discrete, linear models. (Note that a collection of early, but influential, papers can be found in [79].)

Recall that discrete, linear models of time correspond to sequences of 'moments'. These, in turn, can be recognised as infinite words in specific finite automata over infinite strings called Büchi automata. The only relationship between such moments is that each subsequent one is considered as the *next* moment in time. In order to develop a *real-time* version of this approach, we can consider such sequences, but with timing

[1] However, describing time as *flowing* might even be an assumption too far! Several authors have considered time with *gaps* in it [112, 28].

Figure 12.3: Model with timing constraints.

statements referring to particular clocks (in the case in Fig. 12.3, the clock is t) added between each consecutive moment. See Fig. 12.3 for an example of a timed model (here $t < 1$ is a constraint stating that the time, t, is less than 1 on this transition, while the time t is at least 8 on the $t \geqslant 8$ transition).

Where only a finite number of different states exist, Büchi automata can also be extended to recognise these *timed sequences* [12, 13]. In practical applications of such models (see Section 12.4.4) various automata-theoretic operations, such as emptiness checking, are used. These tend to be complex [19], but vary greatly depending on the type of clocks and constraints used.

As well as being developed further, for example, with *clocked transition systems* [165], and extended into *hybrid automata* [11], timed automata have led to many useful and practical verification tools, particularly UPPAAL (see Section 12.4.4).

12.1.6 Intervals

As mentioned above, an *interval* captures some duration of time over which certain properties hold. As in the case of point-based approaches described earlier, there are many different possibilities concerning how intervals are defined. Given a linear model of time, then questions such as whether the 'moments' within this linear order are represented as points or not, whether the order is infinite in either (or both) future or past, etc., must still be decided upon. Additionally, we now have the notion of an interval. Simply, this represents the period of time between two 'moments'. But, of course, there are *many* possibilities here [275]. Does the interval include the end points? Can we have intervals where the start point and end point are the same? Can we have zero length intervals? And so on.

Assuming we have decided on the basic structure of intervals, then the key questions concerned with reasoning in such models are those relating points to intervals, and relating intervals to other intervals. For example, imagine that we have the simple model of time based on $\mathbb{N}$, as described above. Then, let us denote the interval between two time points a and b by $[a, b]$. Now, we might ask:

- does a particular time point c occur within the interval $[a, b]$?
- is a particular time point d adjacent to (i.e., immediately before or immediately after) the interval $[a, b]$ (and what interval do we get if we add d to $[a, b]$)?
- does another interval, $[e, f]$, overlap $[a, b]$?
- is the interval $[h, i]$ a strict *sub-interval* of $[a, b]$?
- what interval represents the overlap of the intervals $[j, k]$ and $[a, b]$?

And so on. As we can see, there are *many* questions that can be formulated. Indeed, we have not even addressed the question of whether intervals are *open* or *closed*. This

question really becomes relevant we consider underlying sets such as the Rational or Real Numbers. Informally, an element x in the temporal domain are within the *open* interval (a, b) if $a < x$ and $x < b$, and is within the *closed* interval $[a, b]$ if $a \leqslant x$ and $x \leqslant b$.

Yet, that is not all. In the temporal models described earlier, we defined temporal properties. Such properties, usually represented by propositions, were satisfied at particular times. Thus, with intervals, we not only have these aspects, but can also ask questions such as:

- does the proposition φ hold *throughout* the interval $[a, b]$?
- does the proposition φ hold anywhere *within* the interval $[a, b]$?
- does the proposition φ hold by the *end* of interval $[a, b]$?
- does the proposition φ hold immediately *before* the interval $[a, b]$?

And so on. Various connectives allow us to express even more:

- given an interval $[a, b]$ where φ holds, is there another interval, $[l, m]$, occurring in the future (i.e., strictly *after* $[a, b]$), on which φ also holds?
- can we split up an interval $[a, b]$ into two sub-intervals, $[a, c_1]$ and $[c_2, b]$ such that φ holds continuously throughout $[a, c_1]$ but not at c_2 (and where joining $[a, c_1]$ and $[c_2, b]$ back together gives $[a, b]$)?

In general, there are *many* questions that can be asked, even when only considering the underlying interval representations. As we will see in Section 12.2.5, once we add specific languages to reason about intervals, then the variation in linguistic constructs brings an even greater set of possibilities.

In a historical context, although work in Philosophy, Linguistics and Logic had earlier considered time periods, for example, [65], interval temporal representations came to prominence in Computer Science and Artificial Intelligence via two important routes:

1. the development, in the early 1980s, of interval temporal logics for the description of computer systems, typically hardware and protocols [135, 204, 208, 252]; and
2. the development, by Allen, of interval representations within Artificial Intelligence, primarily for use in planning systems [6, 9, 7].

We will consider the languages used to describe such phenomena in Section 12.2.5 and will outline some to the applications of interval representations later.

Finally, in this section, we note that there are a number of excellent articles covering much more than we can here: introductory articles, such as [287, 190]; surveys of interval problems in Artificial Intelligence, such as [85, 121]; and the comprehensive survey of interval and duration calculi by Goranko, Montanari, and Sciavicco [127].

12.2 Temporal Language

Just as there are many models for representing temporal situations, there is an abundance of languages for describing temporal properties. Again, many of these languages have evolved from earlier work on modal [181, 61] or tense logics [107, 66]. Yet, with each new type of phenomenon, a different logical approach is often introduced. Thus, there are so many different temporal logics, that we are only able to introduce a few of the more common ones in the following.

12.2.1 Modal Temporal Logic

We will begin with a common language for describing temporal properties, often termed *modal temporal logic* due to its obvious links with modal and tense logics [229, 238, 53, 37]. This is the type of language originally applied by Pnueli [222] and is now widely used in Computer Science. Based on modal notions of *necessity* and *possibility*, the basic (modal) temporal operators are

$\Box\varphi$ — "φ is *always* true in the future"

$\Diamond\varphi$ — "φ is true at *some time* in the future"

These *always* and *sometime* operators form the basis for many logics operating over linear models of time. Yet there are temporal aspects that are impossible to represent simply using '$\Diamond$' and '$\Box$' [161, 292, 53]. Thus, the *until* operator ('$\mathcal{U}$') together with its counterpart, the *unless* operator ('$\mathcal{W}$'), are often imported from tense logic [161, 64]:

$\varphi\mathcal{U}\psi$ — "there exists a moment when ψ holds and φ will continuously hold from now *until* this moment"

$\varphi\mathcal{W}\psi$ — "φ will continuously hold from now on unless ψ occurs, in which case φ will cease"

(Note that there are several variations on the semantics of these operators, for example, differing on whether φ must be satisfied at the current moment.) The similarities between the above connectives means that the *unless* operator is often termed *weak until*. This is generally enough to handle common situations, as both *sometime* and *always* can be defined using *until*. However, in the case of a discrete model of time, it is often convenient to add the *next time* operator, '$\bigcirc$':

$\bigcirc\varphi$ — "φ is true at the *next* moment in time"

The formal semantics for such temporal operators can be given, in the discrete case, using the *next-time* relation introduced earlier. Over models $M = \langle S, N, \pi \rangle$, example semantics can be given as follows.

$$\langle M, s\rangle \models \bigcirc\varphi \quad \text{if, and only if,} \quad \forall t \in S.\ \text{if } N(s,t) \text{ then } \langle M, t\rangle \models \varphi$$

Note that, depending on the semantics of the '$\mathcal{U}$' operator, the '$\bigcirc$' operator may be able to be defined directly using '$\mathcal{U}$' [87].

12.2.2 Back to the Future

Work on tense logics typically incorporated a notion of *past-time* connectives, such as *since* [161, 64]. Though such past-time connectives were omitted from the early temporal logics used in Computer Science, researchers have found it convenient to re-introduce past-time into temporal logics [38, 182].

Thus, temporal logics can contain operators that are the past-time counterparts of $\Box$, $\Diamond$, etc. Discrete temporal logics also incorporate the *previous* operator, '$\bullet$', which is the past-time dual of the "next" operator.

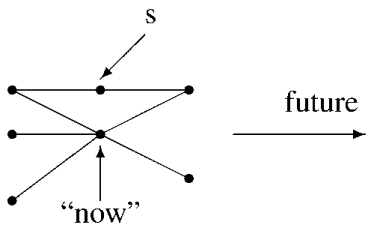
$\bullet\varphi$ — "φ is true at the *previous* moment in time"

In order to indicate some of the interesting interactions between these two operators, we provide more general definitions that depend only on the discreteness of the underlying model, not on its linearity. For this purpose, we again the *next-time* relation introduced earlier and define the semantics for $\bullet$ (over models $M = \langle S, N, \pi \rangle$) as follows.

$$\begin{array}{lll} \langle M, s \rangle \models \bigcirc\varphi & \text{if, and only if,} & \forall t \in S.\ \text{if } N(s,t) \text{ then } \langle M, t \rangle \models \varphi, \\ \langle M, t \rangle \models \bullet\varphi & \text{if, and only if,} & \forall s \in S.\ \text{if } N(s,t) \text{ then } \langle M, s \rangle \models \varphi. \end{array}$$

It is important to note the duality between the semantics of '$\bullet$' and '$\bigcirc$' given earlier. This duality allows us to describe some interesting properties. First of all, note that $\bullet$**false** (or $\bigcirc$**false**) is only satisfiable at the first (or last) moments in the temporal model. Examining the definition above, the only way that $\bullet$**false** can be satisfied is if there are *no* previous moments in time. If there were any previous ones, then **false** would have had to be satisfied at them! Similarly, $\bigcirc$**false** corresponds closely to the ITL operator `fin` describing the end of finite intervals (see Section 12.2.5).

An interesting aspect of the past/future combination is given by the possible interactions between the previous and next operators. For example, the axiom $\varphi \Leftrightarrow \bullet\bigcirc\varphi$ implies that, in models such as that described below, either the state s is disallowed, or if it is allowed, it is indistinguishable from the "now" state by any temporal formula.



As we can see, there is much scope for interesting combinations even with just the *next* and *previous* operators. A large range of interactions can be explored with the *sometime in the future* and the *sometime in the past* operators, or with *until* and *since* [240, 112, 267]. In addition, questions of whether both past and future operators are needed can also been considered [179].

12.2.3 Temporal Arguments and Reified Temporal Logics

While variations of modal temporal logics are widely used in Computer Science, there are alternative approaches that have been developed within Artificial Intelligence. An

obvious alternative to the modal-temporal approach is to essentially use first-order logic statements, treating one of the arguments to each predicate as a reference to time. To see this, let us give the semantics of PTL in classical logic by representing temporal propositions as classical predicates parameterised by the moment in time being considered. Below we look at several temporal formulae and, assuming they are to be evaluated at the moment i, show how these formulae can be represented in classical logic.

$$\begin{array}{lcl} p \wedge \bigcirc q & \rightarrow & p(i) \wedge q(i+1). \\ \Diamond r & \rightarrow & \exists j.\ (j \geqslant i) \wedge r(j). \\ \Box s & \rightarrow & \forall k.\ (k \geqslant i) \Rightarrow s(k). \end{array}$$

This is often termed the *temporal arguments* approach, because the temporal propositions are defined as predicates taking times as arguments.

A further approach that became popular in Artificial Intelligence research is the *reification* approach. Here, the idea is to have predicates such as *holds* and *occurs* applied to properties (often called *fluents*) and times (points or intervals) over which the properties hold (or occur).

Since Allen's Interval Algebra, considered in Section 12.2.5, is of this form, we will not mention these possibilities further. However, there are a great many publications in this area, beginning with initial work on reified approaches, such as McDermott's logic of plans [197], Allen's Interval Algebra [7] (and Section 12.2.5), Situation Calculus [237, 185] and the Event Calculus [169]. In addition, there are numerous surveys and overviews concerning these approaches, including [117, 189, 236, 35].

12.2.4 Operators over Non-discrete Models

As we outlined in Section 12.2.2, various temporal operators have been devised, beginning with *until* and *since* or, alternatively with *sometime in the future* and *sometime in the past*. Indeed, these operators are useful for general linear orders, not just discrete ones [161]. Consequently, if we move away from discrete temporal models towards dense (and, generally, non-discrete) models, these temporal operators form the basis of languages used to describe temporal properties.

Sometime in the future and *sometime in the past* (often referred to as F and P) have been used to analyse a variety of non-discrete logics, for example, those based on $\mathbb{R}$ [111, 112, 114]. Past and future operators, such as *until* and *since* have been productively used in transforming arbitrary formulae into more useful normal forms, for example, separating past-time from future-time [108, 36, 97, 147].

Finally, it is informative to consider the approach taken in *TLR* [39, 164]. Here, the temporal model is based on $\mathbb{R}$ and *until* is taken as the basic temporal operator (only the future time fragment is considered). However, the difficulty of dealing with properties over $\mathbb{R}$ meant that the authors introduced an additional constraint, termed *finite variability*. Here, any property may only change value a *finite* number of times between any two points in time. This avoids the problem of a temporal property, say p, varying between **true** and **false** infinitely over a finite period of time, for example, between 1 and 2 on the Real Number line. (This aspect has also been explored in [77, 118].)

12.2.5 Intervals

As mentioned earlier, the two strong influences for the use of interval temporal representations were from Allen, in Artificial Intelligence, and Moszkowski et al., in Computer Science. We will give a brief flavour of the two different approaches, before mentioning some more recent work.

Allen's interval algebra

Allen was concerned with developing an appropriate formal representation for temporal aspects which could be used in a variety of systems, particularly planning systems. He developed a formal model of intervals, or time periods, and provided syntax to describe the relationships between such intervals [6, 7]. Thus, I_1 *overlaps* I_2 is true if the intervals I_1 and I_2 overlap, I_3 *during* I_4 is true if the interval I_3 is completely contained within I_4, while I_5 *before* I_6 is true if I_5 occurs before I_6. This led on to 13 such binary relations between intervals, giving the Allen Interval Algebra.

Further work on the formalisation and checking of Allen's interval relations can be found in [8, 175, 183, 136, 184, 176] with the algebraic aspects being explored further in [144, 145]. In addition, the basic interval algebra has been extended and improved in many different ways; see [121] for some of these aspects and [85] for a thorough analysis of the computational problems associated with such interval reasoning. These last two references also bring in the work on representing such problems as temporal constraint networks [80, 251] and solving them via constraint satisfaction techniques [291].

Moszkowski's ITL

The interval logic developed by Moszkowski et al. in the early 1980s was much closer in spirit to the propositional (discrete) temporal logics being developed at that time [113]. Moszkowski's logic is called ITL and was originally developed in order to model digital circuits [135, 204]. Although the basic temporal model is similar to that of PTL given earlier, ITL formulae are interpreted in a sub-sequence (defined by $\sigma_b, \ldots, \sigma_e$) of, rather than at a point within, the model σ. Thus, basic propositions (such as P) are evaluated at the *start* of an interval:

$$\langle \sigma_b, \ldots, \sigma_e \rangle \models P \quad \text{if, and only if,} \quad P \in \sigma_b.$$

Now, the semantics of two common PTL operators can be given as follows.

$$\begin{aligned} \langle \sigma_b, \ldots, \sigma_e \rangle \models \Box\varphi \quad &\text{if, and only if,} \quad \text{for all } i, \text{ if } b \leqslant i \leqslant e \\ &\qquad\qquad\qquad\qquad \text{then } \langle \sigma_i, \ldots, \sigma_e \rangle \models \varphi, \\ \langle \sigma_b, \ldots, \sigma_e \rangle \models \bigcirc\varphi \quad &\text{if, and only if,} \quad e > b \text{ and } \langle \sigma_{b+1}, \ldots, \sigma_e \rangle \models \varphi. \end{aligned}$$

A key aspect of ITL is that it contains the basic temporal operators of PTL, together with the *chop* operator, ';', which is used to fuse intervals together (see also [245, 283]). Thus:

$$\langle \sigma_b, \ldots, \sigma_e \rangle \models \varphi; \psi \quad \text{if, and only if,} \quad \text{there exists } i \text{ such that } b \leqslant i \leqslant e$$
$$\text{and both } \langle \sigma_b, \ldots, \sigma_i \rangle \models \varphi \text{ and } \langle \sigma_i, \ldots, \sigma_e \rangle \models \psi.$$

This powerful operator is both useful and problematic (in that the operator ensures a high complexity logic). Useful in that it allows intervals to be split based on their properties; for example, '$\Diamond$' can be derived in terms of ';', i.e.

$$\Diamond\varphi \equiv \textbf{true}; \varphi$$

meaning that there is some (finite) sub-interval in which **true** is satisfied that is followed (immediately) by a sub-interval in which φ is satisfied.

To explain further, simple examples of formulae in ITL are given below, together with English explanations.

- p persists through the current interval: $\Box p$
- The following defines steps within an interval:

  $$up \wedge \bigcirc down \wedge \bigcirc\bigcirc up \wedge \bigcirc\bigcirc\bigcirc down.$$

- The following allows sequences of intervals to be constructed:

  $$\Box\, january;\ \bigcirc\Box\, february;\ \bigcirc\Box\, march; \ldots.$$

- p enjoys a period of being **false** followed by a period of being **true**, i.e., it becomes positive:

  $$\Box\neg p; \bigcirc\Box p.$$

As mentioned earlier, there has also been work on granularity within ITL, particularly via the *temporal projection* operation [206, 130, 58, 131].

In [136], Halpern and Shoham provide a powerful logic (HS) over intervals (not just of linear orders). This logic has been very influential as it subsumes Allen's algebra. Indeed, the HS language with unary modal operators captures entirely Allen's algebra; binary operators are needed to capture the '*chop*' operator within ITL [127], reflecting its additional complexity.

Finally, we note that, there are natural extensions of the above interval approaches. One is to consider intervals, not just over linear orders, but also over arbitrary relations. This moves towards spatial and spatio-temporal logics, see [115] or [74]. Another extension is to bring real-time aspects into interval temporal logics. This has been developed within the work on *duration calculi* [296, 69]. Pointers to such applications of interval temporal logics are provided in Section 12.4. Finally, an interesting extension to interval temporal logic is to add operators that allow endpoints to be moved, thus giving *compass logic* [193].

12.2.6 Real-Time and Hybrid Temporal Languages

In describing real-time aspects, a number of languages can be developed [15]. For instance, standard modal-temporal logic can be extended with annotations expressing real-time constraints [170]. Thus, "I will finish reading this section within 8 time units" might be represented by:

$$\Diamond_{\leqslant 8} finish.$$

Another approach is to use *freeze quantification*. This is similar to the approach taken with hybrid logics (see Section 12.2.8) where a moment in time can be recorded by a variable and then referred to (and used in calculations) later. In addition, there is the possibility of explicitly relating to clocks (and clock variables) within a temporal logic [216]. Consequently, there are a great many different real-time temporal logics (and axiomatisations [249]). There are several excellent surveys of work in this area, including those by Alur and Henzinger [15, 16], Ostroff [217], and Henzinger [140].

In a different direction, the *duration calculus* [78, 69] was introduced in [296], and can be seen as a combination of an interval temporal representation with real-time aspects. It has been applied to many applications in real-time systems, with behaviours mapping on to the dense underlying temporal model.

In developing temporal logics for real-time systems, it became clear that many (hard) practical problems, for example, in complex control systems, required even more expressive power. And so *hybrid systems* were analysed and formalisms for these developed. Hybrid systems combine the standard discrete steps from the automata approach with more complex mathematical techniques related to continuous systems (e.g., differential equations). While we will not delve into this complex area further, we direct the interested reader to the HyTech system [141, 157], the RED system [235] and to work on *hybrid automata* [11].

12.2.7 Quantification

So far we have examined essentially *propositional* languages, most often over discrete, linear models of time. In this section, we will consider the addition of various forms of quantification.[2] Again, we will not provide a comprehensive survey, but will examine a variety of different linguistic extensions that allow us to describe more interesting temporal properties.

Quantification over paths

Although quantification in classical first-order logic is typically used to quantify over a particular data domain, the additional aspect of an underlying temporal structure provides a further possibility in temporal logics, namely the ability to quantify over some aspects of the structure. As we have seen, temporal operators such as '$\Box$' typically quantify over moments of time. Yet, there are other possibilities for quantification, the most common of which is to quantify over possible *paths*. If we consider a linear sequence of time points as a path, then many temporal structures (most obviously, trees) comprise multiple paths [248, 246]. Temporal logics over such branching time structures allow for the possibility of *quantifying* over the paths within the branching structure.

Although branching structures in tense logic were previously studied by Prior (see also [132]), we will exemplify the branching approach by considering two popular temporal logics over branching structures from Computer Science. Computation Tree Logic (CTL) was introduced in [88, 89] and basically used Pnueli's modal temporal logic for describing properties along paths (sequences). However, to deal with the

[2] As one might expect, quantification in temporal logics is related quite closely to quantification in modal logics, though quantified modal logics are not without difficulties [120, 195].

possibility of multiple paths through a tree-like temporal structure, two new logical *path operators* were introduced:

A—'on all future paths starting here'

E—'on some future path starting here'

The CTL approach, however, is to restrict the combinations of temporal/path operators that can occur. Thus, each temporal operator *must* be prefixed by a path operator.

The CTL logic has been popular in specifying properties of reactive systems, for example,

$\mathbf{A}\Box$ *safe* $\quad$ $\mathbf{E}\bigcirc$ *active* $\quad$ $\mathbf{A}\Diamond$ *terminate*

Here, '$\mathbf{A}\Box$' effectively considers all future moments, while '$\mathbf{E}\bigcirc$' must find at least one path such that the required property is true at the next moment in the path, while '$\mathbf{A}\Diamond$' is useful for describing the fact that, whichever future path is considered, the property will hold at some point on that path.

Although restricted in its syntax, CTL has found important uses in verification through *model checking* (see Section 12.4.4) since the complexity of this technique for CTL is relatively low [72].

Just as CTL puts a restriction on the combination of temporal and path quantifiers, the need for more complex temporal formulae, such as '$\Box\Diamond$', over paths in branching structures led to various other branching logics [86, 92, 91, 72], most notably *Full Computation Tree Logic* (CTL*). With CTL* there is no restriction on the combinations of path and temporal operators allowed. Thus, formulae such as

$$\mathbf{A}\Box\Diamond\mathbf{E}\mathbf{A}p$$

can be given. However, there is a price to pay for this increased expressiveness [201], as the decision problem for CTL* is quite complex [92], and so this logic is less often used in practical verification tools.

A further significant development of logics over branching structures was the introduction of *alternating-time temporal logics*. To quote from the abstract of [17]:

"Temporal logic comes in two varieties: linear-time temporal logic assumes implicit universal quantification over all paths that are generated by the execution of a system; branching-time temporal logic allows explicit existential and universal quantification over all paths. We introduce a third, more general variety of temporal logic: alternating-time temporal logic offers selective quantification over those paths that are possible outcomes of games, such as the game in which the system and the environment alternate moves. While linear-time and branching-time logics are natural specification languages for closed systems, alternating-time logics are natural specification languages for open systems. For example, by preceding the temporal operator 'eventually' with a selective path quantifier, we can specify that in the game between the system and the environment, the system has a strategy to reach a certain state. The problems of receptiveness, realisability, and controllability can be formulated as model-checking problems for alternating-time formulae. Depending on whether or not we admit arbitrary nesting of selective path quantifiers and temporal operators, we obtain the two alternating-time temporal logics ATL and ATL*."

Given a set (a *coalition*) of agents, A, ATL allows operators such as $\langle\langle A\rangle\rangle\varphi$, meaning that the set of agents have a collective strategy that will achieve φ. This approach

has been very influential, not only on the specification and verification of open, distributed systems, but also on the modelling of the behaviour of groups of intelligent agents [277, 276].

Finally, we note that the development of the *modal μ-calculus* [171] provided a language that subsumed CTL, CTL*, and many other branching (and linear) logics [76], and there are even timed μ-calculi [142].

Quantification over propositions

In extending from a propositional temporal logic, a small (but significant) step to take is to allow *quantification* over propositions. Thus, the usual first-order quantifier symbols, '$\forall$' and '$\exists$', can be used, but only over Boolean valued variables, namely propositions of the language. Thus, using such a logic, called *quantified propositional temporal logic* (QPTL) [254], it is possible to write formulae such as

$$\exists p.\ p \wedge \bigcirc\bigcirc p \wedge \Diamond\Box\neg p.$$

It is important to note that the particular form of quantification provided here, termed the *substitutional interpretation* [133], can be defined as:

$$\langle M, s\rangle \models \exists p.\ \varphi \quad \text{if, and only if,} \quad \begin{array}{l}\text{there exists a model } M' \text{ such that}\\ \langle M', s\rangle \models \varphi \text{ and } M' \text{ differs from } M\\ \text{in at most the valuation given to } p.\end{array}$$

This style of quantification is used in QPTL and in other extensions of PTL we mention below, such as fixpoint extensions. Note that Haack [133] engages in a thorough discussion of the philosophical arguments between the proponents of the above and the, more standard in classical logic, *objectual interpretation* of quantification:

$$\langle M, s\rangle \models \exists p.\ \varphi \quad \text{if, and only if,} \quad \begin{array}{l}\text{there exists a proposition } q \in \text{PROP}\\ \text{such that } \langle M, s\rangle \models \varphi(p/q)\end{array}$$

where $\varphi(p/q)$ is the formula φ with p replaced by q throughout

QPTL gives an extension of PTL (though still representable using Büchi automata) that allows regular properties to be defined. It was inspired by Wolper's work on extending PTL with grammar operators (termed ETL) [292]. Another approach that followed on from Wolper's work was the development of *fixpoint* extensions [55] of PTL [32, 33, 278, 109], extending PTL with least ('μ') and greatest ('ν') fixpoint operators. In such fixpoint languages, one could write more complex expressions. For a simple example, though, consider:

$$\Box\varphi \equiv \nu\xi.\ \varphi \wedge \bigcirc\xi.$$

Here, $\Box\varphi$ is defined as the maximal (with respect to implication) fixpoint (ξ) of the formula $\xi \Rightarrow (\varphi \wedge \bigcirc\xi)$. Thus, the maximal fixpoint above defines $\Box\varphi$ as the 'infinite' formula

$$\varphi \wedge \bigcirc\varphi \wedge \bigcirc\bigcirc\varphi \wedge \bigcirc\bigcirc\bigcirc\varphi \wedge \cdots.$$

Finally, it is important to note that all these extensions QPTL, ETL, and fixpoint extensions can be shown to be expressively equivalent under certain circumstances [292, 32, 254, 282].

First-order TL

Adding standard first-order (and, in the sense above, objectual) quantification to temporal logic, for example, PTL, is appealing yet fraught with danger. Such a logic is very convenient for describing many scenarios, but is so powerful that we can write down formulae that capture a form of arithmetical induction, from which it is but a short step to being able to represent full arithmetic [262, 263, 1]. Consequently, full first-order temporal logic is incomplete; in other words the set of valid formulae is *not* recursively enumerable (or finitely axiomatisable) when considered over models such as the Natural Numbers.

While some work was carried out on methods for handling, where possible, such specifications [191], first-order temporal logic was generally avoided. Even "small" fragments of first-order temporal logic, such as the *two-variable monadic* fragment, are not recursively enumerable [199, 149].

However, a breakthrough by Hodkinson et al. [149] showed that *monodic* fragments of first-order temporal logics could have complete axiomatisations and even be decidable. A monodic temporal formula is one whose temporal subformulae have, at most, one free variable. Thus, $\forall x.\ p(x) \Rightarrow \bigcirc q(x)$ is monodic, while $\forall x.\forall y.\ p(x, y) \Rightarrow \bigcirc q(x, y)$ is not. Wolter and Zakharyaschev showed that any set of valid *monodic* formulae is finitely axiomatisable [295] over a temporal model based on the Natural Numbers. Intuitively, the monodic fragment restricts the amount of information transferred between temporal states so that, effectively, only individual elements of information are passed between temporal states. This avoids the possibility of describing the evolution through time of more complex items, such as relations, and so retains desirable properties of the logic. In spite of this, the addition of equality or function symbols can again lead to the loss of recursive enumerability from these monodic fragments [295, 82], though recovery of this property is sometimes possible [146].

12.2.8 Hybrid Temporal Logic and the Concept of "now"

The term *hybrid logic* is here used to refer to logical systems comprising a hybrid of modal/temporal and classical aspects [156]. Basically, hybrid modal logics provide a language for referring to specific points in a model. This approach is widely used in *description logics*, with nominals typically referring to individuals [27]. In the case of temporal logics, such a possibility was suggested by Prior [229] in tense logics, but did not become popular until the 1990s, for example, with [52, 54].

The ability to refer to specific time points, for example *now*, has been found to be very useful in a number of applications. Consequently, operators such as '$\downarrow$' are used to bind a variable to the current point [125]. This allows the specifier to describe a temporal situation, record the point at which it occurs, then use a reference to this point in later formulae. This usefulness, has led to work on both reasoning techniques and complexity for such logics [83, 20].

12.3 Temporal Reasoning

Having considered the underlying temporal representations, together with languages that are used to describe such situations, we now take a brief look at a few of the *reasoning methods* developed for these languages.

12.3.1 Proof Systems

There are a wide variety of axiom systems for temporal logics and, consequently, proof methods based upon them. For PTL, the most popular modal-temporal logic, an axiomatisation was provided in [113], and revisited in [243]:

$$\vdash \neg\bigcirc\varphi \Leftrightarrow \bigcirc\neg\varphi$$
$$\vdash \bigcirc(\varphi \Rightarrow \psi) \Rightarrow (\bigcirc\varphi \Rightarrow \bigcirc\psi)$$
$$\vdash \Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$$
$$\vdash \Box\varphi \Rightarrow (\varphi \wedge \bigcirc\Box\varphi)$$
$$\vdash \Box(\varphi \Rightarrow \bigcirc\varphi) \Rightarrow (\varphi \Rightarrow \Box\varphi)$$
$$\vdash (\varphi \mathcal{U} \psi) \Rightarrow \Diamond\psi$$
$$\vdash (\varphi \mathcal{U} \psi) \Leftrightarrow (\psi \vee (\varphi \wedge \bigcirc(\varphi \mathcal{U} \psi)))$$

In addition, all propositional tautologies are theorems and the inference rules used are *modus ponens* together with *temporal generalisation*:

$$\frac{\vdash \varphi}{\vdash \Box\varphi}.$$

However, several other proof systems, even for this logic have been given [172, 191, 87, 260]. Many proof systems for temporal logics are based on their tense logic predecessors, such as those systems developed by van Benthem [275] and Goldblatt [123].

As to other varieties of temporal logic, perhaps the most widely studied are variants of branching-time logics. Thus, there are proof systems for CTL [225] and, recently, CTL* [241, 242].

Concerning quantifier extensions, proof systems have been developed for QPTL [106, 166]. For full first-order temporal logics, an arithmetical axiomatisation has been given in [262]. Recently, complete (*monodic*) fragments of both linear and branching temporal logics have been provided [295, 150] while proof systems have been developed for alternative fragments of first-order temporal logics [221].

12.3.2 Automated Deduction

Given the utility of temporal formalisms, it is not surprising that many computational tools for establishing the truth of temporal statements have been developed. In some approaches, such as model checking (see Section 12.4.4), temporal conditions are often replaced by finite automata over infinite words. The close link between temporal logics and such finite automata [254, 279, 280] means that decisions about the truth of temporal statements can often be reduced to automata-theoretic questions. Rather than discussing this further, we will consider more traditional automated approaches, such as *tableau* and *resolution* systems. However, before doing this, we note that the *temporal arguments* view of temporal representations given earlier points to an obvious way to automate temporal reasoning, namely to translate statements in temporal logic to corresponding statements in classical logic, adding an extra argument. Thus the implication

$$(p \wedge \bigcirc q) \Rightarrow \Box r$$

might become, if we consider the simple Natural Number basis for temporal logic, the following formula

$$\forall t.\ (p(t) \wedge q(t+1)) \quad \Rightarrow \quad (\forall u.\ (u \geqslant t) \Rightarrow r(u)).$$

This is an appealing approach, and has been successfully applied to the translation of modal logics [212]. However, the translation approach has been used relatively little [210], possibly because the fragment of logic translated to often has high complexity; see [143, 124].

Probably the most popular approach to deciding the truth of temporal formulae is the *tableau* method. The basis of the tableau approach is to recursively take the formula apart, until atomic formulae are dealt with, then assess the truth of the formula in light of the truth constraints imposed by these atomic literals [75]. In classical logic, this typically generates a *tree* of subformulae. However, in temporal logics, as in many modal logics [101], either an infinite tree or, more commonly, a graph structure is generated. The main work in this area was carried out by Wolper [292, 293], who developed a tableau system for discrete, propositional, linear temporal logic. Several other tableau approaches have been reported, both for the above logic [128, 253], and for other varieties of temporal logic [90, 194, 126, 220, 168]. However, the structures built using the tableau method are very close to the ω-automata representing the formulae. Thus, particularly in the case of logics such as CTL*, automata theoretic approaches are often used [92].

In recent years, *resolution* based approaches [244, 30] have been developed. These have consisted of both *non-clausal* resolution, where the formulae in question do not have to be translated to a specific clausal form [3, 5], and *clausal* resolution, where such a form is required [67, 284, 95, 99]. Again, resolution techniques have been extended beyond the basic propositional, discrete, linear temporal logics [57, 81, 167], leading to some practical systems (see Section 12.4.3). For further details on such approaches, particularly for discrete temporal logics, the article [243] is recommended.

Automated deduction for interval temporal logics has often been subsumed by work on temporal planning or temporal constraint satisfaction (see Section 12.4.3), though some work has been carried out on SAT-like procedures for interval temporal problems [269] and tableau methods for interval logics [126, 58].

12.4 Applications

In this section we will provide an outline of some of the ways in which the concepts described in the previous sections can be used to describe and reason about different temporal phenomena. This is not intended to be a comprehensive survey and, again, there are a number of excellent publications covering these topics in detail. However, we aim, through the descriptions below, to provide a sense of the breadth of representational capabilities of temporal logics.

12.4.1 Natural Language

The representation of elements of natural language, particularly *tense*, is not only an intuitively appealing use of temporal logic but provides the starting point for much of the work on temporal logics described in this chapter. The main reason for this is the work of Prior [138] on the formal representation of tense [229]. The sentence:

"I am *writing* this section, will *write* the next section later, and eventually will have *written* the whole chapter"

naturally contains the verb "to write" under different tenses. The tenses used depend upon the moment in time referred to, relative to the person describing it. Prior carried out a logical analysis of such uses of tense, developing *tense logic*, and captured a variety of temporal connectives that have subsequently been used in many temporal logics, for example, *until*, *since*, *before*, *after*, and *during*.

Subsequent work by Kamp [161] related tense operators, such as *since* and *until*, to first-order languages of linear order. This work has been very influential, leading to deeper analysis of tense logic [107, 64, 65], and then to work on temporal logics. An excellent summary of such work on tense logic is given in [66].

The representation of natural language using various temporal representations has also moved on, for example, through the work of van Benthem [275], Galton [116], Kamp and Reyle [162], Steedman [257, 258], ter Meulen [265] and Pratt-Hartmann [228]. A more detailed overview of temporal representation in natural language can be found in [266].

Finally, work in this area naturally impacts upon practical applications, such as the use of temporal representation in *legal reasoning* [288].

12.4.2 Reactive System Specification

It is in the description of complex (interacting, concurrent or distributed) systems that temporal representations have been so widely used. While it is clearly impossible to give a thorough survey of all the ways that temporal notations have been used, particularly as formal specifications, we will give some initial pointers to this area below.

Probably the best known style of temporal specification, which has been used in the specification and verification of programs, is that instigated by Pnueli [222, 113, 223] and continued by Manna and Pnueli through a series of books [191, 192] and papers. In such an approach, the expressive power of modal temporal languages is used in order to specify properties such as *safety*:

$$\Box(temperature < 500)$$

ensuring, in this case, that in any current or future situation, the temperature must be less than 500, *liveness*:

$$\Diamond(terminate \wedge successful)$$

where, for example, some process is guaranteed to eventually terminate successfully, and *fairness*:

$$\Box\Diamond request \Rightarrow \Diamond respond$$

guaranteeing that if a request is made often enough ('$\Box\Diamond$' implies "infinitely often") then, eventually, a response will be given.

In parallel with the Manna/Pnueli line of work, Lamport developed a *Temporal Logic of Actions* (TLA) [177]. This has also been successful, leading to a large body of work on temporal specifications of a variety of systems [178]. Finally, it should be noted that descriptions of many real-world applications have been given using other

varieties of temporal language, such as real-time temporal logics (see Section 12.2.6), interval temporal logics (see Section 12.2.5), partial-order temporal logics [268], etc.

Once a system has been specified, for example, using the logical approach above, a number of techniques may be used. These include: *refinement*, in order to develop a modified specification [2]; *execution*, where the specification is treated like a program and executed directly [205, 36, 98]; *deductive verification*, whereby the relationship between two logical specifications is proved (see Sections 12.3.2 and 12.4.3); *algorithmic verification*, where the match between the specification and a finite-state description (for example, a program) is established (see Section 12.4.4); and *synthesis*, whereby such a finite-state description (program) is generated (semi-) automatically from the specification [226].

12.4.3 Theorem-Proving

Several of the reasoning techniques described in Section 12.3 have been developed into powerful proving tools. In the case of modal-temporal logic, the best known is the *Stanford Temporal Prover* (*STeP*) developed over a number of years by Manna and colleagues [259, 51, 50]. STeP supports the "the computer-aided formal verification of reactive, real-time and hybrid systems based on their temporal specification". It incorporates both model checking and proof procedures and is therefore able to tackle more complex, even infinite state, verification problems.

For modal-temporal logics, several other systems have been developed, notably TeMP [155], based on the clausal temporal resolution approach [167], TLPVS [270, 224], built on top of PVS [233], and the Logics Workbench [188, 253].

In terms of interval temporal logics, many of the reasoning techniques and uses of interval algebras have been incorporated in temporal planning [104, 271] and temporal constraint satisfaction systems [85, 291]. These topics are covered in depth elsewhere, but we here just cite some of the relevant work on temporal planning, notably that by Bacchus and Kabanza on using temporal logics to control the planning process [29], by Fox and Long on describing complex temporal domains [103], by Geffner and Vidal on constraint-based temporal planning [286], by Mayer et al. on planning using first-order temporal logics [196], by Gerevini et al. on developing the LPG planning system [187], and by Doherty on planning in temporal action logic [84].

There has been less development and implementation of Moszkowski's ITL, but see [159] for several tools based on this approach. However, the direct execution of ITL statements is the basis for the *Tempura* programming language [205, 134] which is important in the development of compositional reasoning [207]. Just as Tempura is based on forward-chaining execution of ITL statements, the METATEM approach forward-chains through PTL formulae, though in a specific normal form [36, 98]. Alternative approaches to the execution of temporal statements are based on the extension of logic programming to modal temporal logic, giving Templog [4, 40] or Chronolog [215, 186] or the addition of interval constructs to logic programming, giving the temporal event calculus [169, 35, 209, 34]. For introductions to the ideas behind executable temporal logic, see [214, 96].

12.4.4 Model Checking

Undoubtedly the most practical use of temporal logic is in *model checking*. This is simply based on the idea of satisfiability checking. Thus, given a model, M, and a

property, φ, is it the case that φ is true throughout M? If M represents all the possible paths through a hardware design, or all the possible executions of a program, then answering this question corresponds to checking whether all the executions/paths satisfy the property. Consequently, this is used extensively in the formal verification of hardware descriptions, network protocols and complex software [151, 73].

That model checking has become so popular is due mainly to improvements in the *engineering* of model checking algorithms and model checkers. Simply enumerating all the paths through the model M and checking whether φ is satisfied on that path can clearly be slow. However, an automata-theoretic view of the approach helped suggest improvements [254]. Here, the idea is that a Büchi automaton, A_M, can be developed to represent all the paths through M, while another Büchi automaton, $A_{\neg\varphi}$, can be developed to capture all paths that *do not* satisfy φ. Thus, $A_{\neg\varphi}$ represents all the *bad* paths. Now, once we have these two automata, we simply take the product, $A_M \times A_{\neg\varphi}$, which produces a new automaton whose paths are those that satisfy *both* automata. Thus, a path through $A_M \times A_{\neg\varphi}$ would be a path through A_M that also *did not* satisfy φ. Now, the question of whether "all paths through M satisfy φ" can be reduced to the question of whether "the automaton $A_M \times A_{\neg\varphi}$ has *no* accepting runs". This automata theoretic view was very appealing and led to significant theoretical advances [279]. However, a key practical problem is that the space (and time) needed to construct the product of the two automata can be prohibitively large. Thus, mechanisms for reducing this were required before model checking could be widely used.

Two approaches have been developed that have led to widespread use of model checking in system verification. The first is the idea of *on-the-fly model checking* [122, 152]. Here, the product automaton is only constructed as needed (i.e., it is built *on the fly*), avoiding expensive product construction in many cases. This approach has been particularly successful in the `Spin` model checker [153, 256], which checks specifications written in linear temporal logic against systems represented in the `Promela` modelling language [153].

The second approach is to still carry out automata composition, but to find a much better (and more efficient) representation for the structures involved. This is termed *symbolic model checking* [63] and uses Binary Decision Diagrams (BDDs) [60] to represent both the system and property. BDDs are a notation in which Boolean formulae can be represented as a graph structure on which certain logical operations can be very quick. This is dependent on finding a good ordering for the Boolean predicates within the graph structure. The use of varieties of BDDs has led to a significant increase in the size of system that can be verified using model checking, and is particularly successful in the `SMV` [198, 62] and `nu_smv` [71, 211] model checkers, which check branching temporal formulae (in CTL) over finite automata.

Model checking has also been applied to real-time systems [10, 47, 272, 219, 173, 234, 285], most successfully via the UPPAAL system. This has been used to model and verify networks of timed automata, and uses model checking as a key component [180, 42].

Although model checking has been relatively successful, much work still remains. Current work on abstraction techniques (i.e., reducing complex systems to simpler ones amenable to model checking), SAT based and bounded model checking [48, 49, 26, 227], probabilistic model checking [174, 230], and model checking for high-level

languages such as C [31, 255] and Java [289, 160] promise even greater advances in the future.

12.4.5 PSL/Sugar

The success of model checking, particularly in the realm of *hardware* design, has led to the use of temporal techniques in a number of industrial areas. Standards for specifying the functional properties of hardware logic designs are now based upon temporal logics. For example, there is a large consortium developing and applying PSL/Sugar [41, 231]. This, and other approaches such as ForSpec [21] and SystemVerilog Assertions [261], extend temporal logic adding regular expressions and clocks and even allowing more complex combinations of automata and regular expressions [43].

12.4.6 Temporal Description Logics

It is often desirable to combine temporal logic with description logic, to give a *temporal description logic*. While there have been some attempts to consider the general problem of combining such non-classical logics [45], it is only in specific areas that a systematic examination of detailed combinations has been carried out. Temporal description logics are just such an area.

The motivation for studying temporal description logics primarily arose from work on temporal databases [94, 24] and dynamic knowledge/plan representation [247, 22, 23]. A thorough survey of the varieties of combination, and their properties, is provided by Artale and Franconi in [25]. Different logical combinations can be produced, depending on what type of temporal logic is used (e.g., point-based or interval) and how the temporal dimension is incorporated. A simple temporal description logic can be obtained by combining a basic description logic with a standard point-based temporal logic, such as PTL. This combination can be carried out in a number of ways, two of which are termed *external* and *internal* in [25]:

- using an *external* approach, the temporal dimension is used to relate different (static) 'snapshots' of the system, each of which is described by a description logic formula;
- using an *internal* approach, the temporal dimension is effectively embedded within the description logic.

For simplicity, we consider the first view; for example,

$$\texttt{parentof}(\texttt{Michael}, \texttt{Christopher}) \Rightarrow \bigcirc\, \texttt{parentof}(\texttt{Michael}, \texttt{James})$$

Here parentof(Michael, Christopher) is true at present, and within the current description logic theory, while parentof(Michael, James) will be true at the next moment in time. This, relatively simple, approach allows us to add a dynamic element to description logics. Yet, it is also important to be able to carry information between temporal states, for example,

$$\forall \texttt{x}.\, \texttt{parentof}(\texttt{Michael}, \texttt{x}) \Rightarrow \bigcirc \texttt{parentof}(\texttt{Michael}, \texttt{x}) \qquad (12.1)$$

However, just as in first-order temporal logics [263, 1] (see Section 12.2.7), the amount of information transferred between temporal states can drastically affect the properties of the logic. Thus, varieties including (12.1) above, where only individual elements of information are passed between temporal states, correspond to the class of *monodic* first-order temporal logics [149] in which decidability can be retained. Correspondingly, temporal description logics where concepts can evolve over time, but where the temporal evolution of roles is limited, can retain recursive enumerability and, often, decidability [294, 25].

12.5 Concluding Remarks

In this chapter we have provided an overview of a variety of aspects concerning temporal representation and reasoning. Even though this is not meant to be exhaustive, it is clear that not only are there many subtle aspects within the general area of temporal representation, but there is also a vast number of other areas and applications within which temporal approaches are relevant.

Although we have described many aspects of temporal representation and reasoning, others that we have omitted include:

- *temporal data mining*—the extraction of temporal patterns either from large datasets or streams of data [264, 46];
- *temporal databases*—the incorporation in (relational) databases and query languages of various temporal constraints [46, 273, 70]; and
- *probabilistic temporal logics*—the extension of temporal representations with probabilities and uncertainty [137], together with various applications such as probabilistic model checking [230].

As is clear in these areas, as well as in the topics examined within this chapter, research on temporal representation and reasoning continues to expand and progress. New formalisms, techniques and tools are being developed, and all of this points to the increasing relevance of temporal representation and reasoning to knowledge representation, and to Computer Science and Artificial Intelligence in general.

Acknowledgements

The author would like to thank a number of experts in the area who have reviewed this chapter and provided valuable insights and corrections, in particular: Anthony Galton; Valentin Goranko; Ian Hodkinson; Jixin Ma; Angelo Montanari; Ben Moszkowski; Wojciech Penczek; Ian Pratt-Hartmann; Mark Reynolds; Pierre-Yves Schobbens; and Mike Wooldridge.

Bibliography

[1] M. Abadi. The power of temporal proofs. *Theoretical Computer Science*, 65(1):35–83, 1989.

[2] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.
[3] M. Abadi and Z. Manna. Non-clausal temporal deduction. In *Proc. Workshop on Logics of Programs*, *Lecture Notes in Computer Science*, vol. 193, pages 1–15. Springer, June 1985.
[4] M. Abadi and Z. Manna. Temporal logic programming. *Journal of Symbolic Computation*, 8(3):277–295, 1989.
[5] M. Abadi and Z. Manna. Nonclausal deduction in first-order temporal logic. *ACM Journal*, 37(2):279–317, April 1990.
[6] J.F. Allen. Maintaining knowledge about temporal intervals. *ACM Communications*, 26(11):832–843, November 1983.
[7] J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.
[8] J.F. Allen and P.J. Hayes. A common sense theory of time. In *Proc. 9th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 528–531, August 1985.
[9] J.F. Allen and J.A. Koomen. Planning using a temporal world model. In *Proc. 8th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 741–747, August 1983.
[10] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, May 1993.
[11] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.
[12] R. Alur and D.L. Dill. The theory of timed automata. In de Bakker et al. [79], pages 45–73.
[13] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(1):183–235, 1994.
[14] R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. *Theoretical Computer Science*, 331(1):97–114, 2005.
[15] R. Alur and T.A. Henzinger. Logics and models of real time: A survey. In de Bakker et al. [79], pages 74–106.
[16] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.
[17] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *ACM Journal*, 49(5):672–713, 2002.
[18] R. Alur, T.A. Henzinger, F.Y.C. Mang, S. Qadeer, S.K. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proc. 10th International Conference on Computer Aided Verification (CAV)*, *Lecture Notes in Computer Science*, vol. 1427, pages 521–525. Springer, 1998.
[19] R. Alur and P. Madhusudan. Decision problems for timed automata: a survey. In *Formal Methods for the Design of Real-Time Systems (International School on Formal Methods for the Design of Computer, Communication and Software Systems)*, *Lecture Notes in Computer Science*, vol. 3185, pages 1–24. Springer, 2004.
[20] C. Areces, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *The Logic Journal of the IGPL*, 8(5):653–679, 1999.

[21] R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, E. Singerman, A. Tiemeyer, M.Y. Vardi, and Y. Zbar. The ForSpec temporal logic: a new temporal property-specification language. In *Proc. 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, *Lecture Notes in Computer Science*, vol. 2280, pages 296–311. Springer, 2002.
[22] A. Artale and E. Franconi. A temporal description logic for reasoning about actions and plans. *Journal of Artificial Intelligence Research (JAIR)*, 9:463–506, 1998.
[23] A. Artale and E. Franconi. Representing a robotic domain using temporal description logics. *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 13(2):105–117, April 1999.
[24] A. Artale and E. Franconi. Temporal entity-relationship modeling with description logics. In *Proc. International Conference on Conceptual Modelling (ER)*. Springer-Verlag, November 1999.
[25] A. Artale and E. Franconi. Temporal description logics. In Fisher et al. [100], pages 375–388.
[26] G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani. Bounded model checking for timed systems. In *Proc. 22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE)*, *Lecture Notes in Computer Science*, vol. 2529, pages 243–259. Springer, 2002.
[27] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
[28] M. Baaz, A. Leitsch, and R. Zach. Completeness of a first-order temporal logic with time-gaps. *Theoretical Computer Science*, 160(1–2):241–270, 1996.
[29] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1–2):123–191, 2000.
[30] L. Bachmair and H. Ganzinger. Resolution theorem proving. In A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning*, vol. I, pages 19–99. Elsevier Science, 2001 (Chapter 2).
[31] T. Ball and S.K. Rajamani. The SLAM toolkit. In *Proc. 13th International Conference on Computer Aided Verification (CAV)*, *Lecture Notes in Computer Science*, vol. 2102, pages 260–264. Springer, 2001.
[32] B. Banieqbal and H. Barringer. A study of an extended temporal language and a temporal fixed point calculus. Technical Report UMCS-86-10-2, Department of Computer Science, University of Manchester, November 1986.
[33] B. Banieqbal and H. Barringer. Temporal logic with fixed points. In *Proc. Temporal Logic in Specification*, *Lecture Notes in Computer Science*, vol. 398, pages 62–74. Springer, 1987.
[34] C. Baral. Query Answering, 2007. (*In this volume*).
[35] C. Baral and M. Gelfond. Logic programming and reasoning about actions. In Fisher et al. [100], pages 389–428.
[36] H. Barringer, M. Fisher, D. Gabbay, R. Owens, and M. Reynolds, editors. *The Imperative Future: Principles of Executable Temporal Logics*. Research Studies Press, Chichester, United Kingdom, 1996.
[37] H. Barringer and D. Gabbay. Modal varieties of temporal logic. In Fisher et al. [100], pages 119–166.

[38] H. Barringer, R. Kuiper, and A. Pnueli. A compositional temporal approach to a CSP-like language. In *Proc. IFIP Working Conference "The Role of Abstract Models in Information Processing"*, Vienna, 1985.

[39] H. Barringer, R. Kuiper, and A. Pnueli. A really abstract concurrent model and its temporal logic. In *Proc. 13th ACM Symposium on the Principles of Programming Languages (POPL)*, January 1986.

[40] M. Baudinet. On the expressiveness of temporal logic programming. *Information and Computation*, 117(2):157–180, 1995.

[41] I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh. The temporal logic sugar. In *Proc. 13th International Conference on Computer Aided Verification (CAV)*, *Lecture Notes in Computer Science*, vol. 2102, pages 363–367. Springer, 2001.

[42] G. Behrmann, A. David, K.G. Larsen, O. Möller, P. Pettersson, and W. Yi. UPPAAL—present and future. In *Proc. 40th IEEE Conference on Decision and Control (CDC)*. IEEE Computer Society Press, 2001.

[43] S. Ben-David, D. Fisman, and S. Ruah. Embedding finite automata within regular expressions. In *Proc. 1st International Symposium on Leveraging Applications of Formal Methods (ISoLA)*. Springer, 2004.

[44] J. Bengtsson and W. Yi. Timed automata: semantics, algorithms and tools. In *Lecture Notes on Concurrency and Petri Nets*, *Lecture Notes in Computer Science*, vol. 3098. Springer-Verlag, 2004.

[45] B. Bennett, C. Dixon, M. Fisher, E. Franconi, I. Horrocks, and M. de Rijke. Combinations of modal logics. *AI Review*, 17(1):1–20, 2002.

[46] C. Bettini, S. Jajodia, and S. Wang. *Time Granularities in Databases, Data Mining and Temporal Reasoning*. Springer-Verlag, New York, USA, 2000.

[47] D. Beyer, C. Lewerentz, and A. Noack. Rabbit: A tool for BDD-based verification of real-time systems. In *Proc. 15th International Conference on Computer Aided Verification, (CAV)*, *Lecture Notes in Computer Science*, vol. 2725, pages 122–125. Springer, 2003.

[48] A. Biere, A. Cimatti, E.M. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proc. Design Automation Conference (DAC)*, pages 317–320, 1999.

[49] A. Biere, A. Cimatti, E.M. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, *Lecture Notes in Computer Science*, vol. 1579, pages 193–207. Springer, 1999.

[50] N. Bjørner, A. Browne, M. Colón, B. Finkbeiner, Z. Manna, H. Sipma, and T. Uribe. Verifying temporal properties of reactive systems: A STeP tutorial. *Formal Methods in System Design*, 16(3):227–270, 2000.

[51] N. Bjørner, Z. Manna, H. Sipma, and T. Uribe. Deductive verification of real-time systems using STeP. *Theoretical Computer Science*, 253(1):27–60, 2001.

[52] P. Blackburn. Nominal tense logic. *Notre Dame Journal of Formal Logic*, 34(1):56–83, 1993.

[53] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[54] P. Blackburn and M. Tzakova. Hybrid languages and temporal logic. *Logic Journal of the IGPL*, 7(1):27–54, 1999.

[55] A. Blass and Y. Gurevich. Existential fixed-point logic. In *Computation Theory and Logic*, *Lecture Notes in Computer Science*, vol. 270, pages 20–36. Springer, 1987.

[56] L. Bolc and A. Szalas, editors. *Time and Logic: A Computational Approach*. Univ. College London Press, 1995.

[57] A. Bolotov and M. Fisher. A clausal resolution method for CTL branching-time temporal logic. *Journal of Experimental and Theoretical Artificial Intelligence*, 11:77–93, 1999.

[58] H. Bowman and S. Thompson. A decision procedure and complete axiomatization of finite interval temporal logic with projection. *Journal of Logic and Computation*, 13(2):195–239, 2003.

[59] D. Bresolin, A. Montanari, and G. Puppis. Time granularities and ultimately periodic automata. In *Proc. 9th European Conference on Logics in Artificial Intelligence (JELIA)*, *Lecture Notes in Computer Science*, vol. 3229, pages 513–525. Springer, 2004.

[60] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[61] R.A. Bull and K. Segerberg. Basic modal logic. In Gabbay and Guenthner [110], Chapter II.1, pages 1–88.

[62] J.R. Burch, E.M. Clarke, D.E. Long, K.L. McMillan, and D.L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4):401–424, 1994.

[63] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proc. 5th IEEE Symposium on Logic in Computer Science (LICS)*, pages 428–439. IEEE Computer Society Press, 1990.

[64] J.P. Burgess. Axioms for tense logic; 1—'Since' and 'until'. *Notre Dame Journal of Formal Logic*, 23(4):367–374, October 1982.

[65] J.P. Burgess. Axioms for tense logic; 2—Time periods. *Notre Dame Journal of Formal Logic*, 23(4):375–383, October 1982.

[66] J.P. Burgess. Basic tense logic. In Gabbay and Guenthner [110], Chapter II.2, pages 89–134.

[67] A. Cavali and L. Fariñas del Cerro. A decision method for linear temporal logic. In *Proc. 7th International Conference on Automated Deduction (CADE)*, *Lecture Notes in Computer Science*, vol. 170, pages 113–127. Springer, 1984.

[68] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *ACM Journal*, 28(1):114–133, January 1981.

[69] Z. Chaochen and M.R. Hansen. *Duration Calculus—A Formal Approach to Real-Time Systems*. *EATCS Monographs in Theoretical Computer Science*. Springer, 2004.

[70] J. Chomicki and D. Toman. Temporal databases. In Fisher et al. [100], pages 429–468.

[71] A. Cimatti, E.M. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: A new symbolic model verifier. In *Proc. 11th International Conference on Computer Aided Verification (CAV)*, *Lecture Notes in Computer Science*, vol. 1633, pages 495–499. Springer, 1999.

[72] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
[73] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, December 1999.
[74] A. Cohn. Spatial Reasoning, 2007. (*In this volume*).
[75] M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors. *Handbook of Tableau Methods*. Kluwer Academic Press, 1999.
[76] M. Dam. CTL* and ECTL* as fragments of the modal mu-calculus. *Theoretical Computer Science*, 126(1):77–96, 1994.
[77] E. Davis. Infinite loops in finite time: Some observations. In *Proc. International Conference on Knowledge Representation and Reasoning (KR)*, pages 47–58, 1992.
[78] Duration calculus. http://www.iist.unu.edu/dc.
[79] J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors. *Proc. REX Workshop on Real-Time: Theory in Practice*, *Lecture Notes in Computer Science*, vol. 600. Springer, 1991.
[80] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1–3):61–95, 1991.
[81] A. Degtyarev, M. Fisher, and B. Konev. Monodic temporal resolution. *ACM Transactions on Computational Logic*, 7(1):108–150, January 2006.
[82] A. Degtyarev, M. Fisher, and A. Lisitsa. Equality and monodic first-order temporal logic. *Studia Logica*, 72(2):147–156, 2002.
[83] S. Demri and R. Goré. Cut-free display calculi for nominal tense logics. In *Proc. Conference on Tableaux Calculi and Related Methods (TABLEAUX)*, *Lecture Notes in Artificial Intelligence*, vol. 1617, pages 155–170. Springer-Verlag, 1999.
[84] P. Doherty. Temporal Action Logic, 2007. (*In this volume*).
[85] T. Drakengren and P. Jonsson. Computational complexity of temporal constraint problems. In Fisher et al. [100], pages 197–218.
[86] E.A. Emerson. Alternative semantics for temporal logics. *Theoretical Computer Science*, 26:121–130, 1983.
[87] E.A. Emerson. In *Handbook of Theoretical Computer Science*, pages 997–1071. Elsevier Science Publishers B.V., 1990 (chapter *Temporal Modal Logic*).
[88] E.A. Emerson and E.M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proc. 7th International Colloquium on Automata, Languages and Programming (ICALP)*, *Lecture Notes in Computer Science*, vol. 85, pages 169–181. Springer, 1980.
[89] E.A. Emerson and E.M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.
[90] E.A. Emerson and J.Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.
[91] E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not never" revisited: on branching versus linear time temporal logic. *ACM Journal*, 33(1):151–178, January 1986.

[92] E.A. Emerson and A.P. Sistla. Deciding full branching time logic. *Information and Control*, 61:175–201, 1984.
[93] J. Euzenat and A. Montanari. Time granularity. In Fisher et al. [100], pages 59–118.
[94] M. Finger and D. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic, Language, and Information*, 1:203–234, 1992.
[95] M. Fisher. A resolution method for temporal logic. In *Proc. 12th International Joint Conference on Artificial Intelligence (IJCAI)*, Sydney, Australia, 1991. Morgan Kaufman.
[96] M. Fisher. An introduction to executable temporal logics. *Knowledge Engineering Review*, 11(1):43–56, March 1996.
[97] M. Fisher. A normal form for temporal logic and its application in theorem-proving and execution. *Journal of Logic and Computation*, 7(4), July 1997.
[98] M. Fisher. METATEM: the story so far. In *Proc. 3rd International Workshop on Programming Multiagent Systems (ProMAS)*, *Lecture Notes in Artificial Intelligence*, vol. 3862. Springer-Verlag, 2006.
[99] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, January 2001.
[100] M. Fisher, D. Gabbay, and L. Vila, editors. *Handbook of Temporal Reasoning in Artificial Intelligence*, *Foundations of Artificial Intelligence*, vol. 1. Elsevier Press, 2005.
[101] M. Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, 13(2):237–247, April 1972.
[102] K. Forbus. Qualitative Reasoning, 2007. (*In this volume*).
[103] M. Fox and D. Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)*, 20:61–124, 2003.
[104] M. Fox and D. Long. Time in planning. In Fisher et al. [100], pages 497–536.
[105] M. Franceschet. Dividing and conquering the layered land. PhD thesis, Department of Mathematics and Computer Science, University of Udine, Italy, 2001.
[106] T. French and M. Reynolds. A sound and complete proof system for QPTL. In *Proc. International Conference on Advances in Modal Logic (AiML)*, 2002.
[107] D. Gabbay. Model theory for tense logics and decidability results for non-classical logics. *Annals of Mathematical Logic*, 8:185–295, 1975.
[108] D. Gabbay. Expressive functional completeness in tense logic (preliminary report). In U. Monnich, editor. *Aspects of Philosophical Logic*, pages 91–117. Reidel, Dordrecht, 1981.
[109] D. Gabbay. Declarative past and imperative future: executable temporal logic for interactive systems. In *Proc. International Colloquium on Temporal Logic in Specification*, *Lecture Notes in Computer Science*, vol. 398, pages 67–89. Springer-Verlag, 1989.
[110] D. Gabbay and F. Guenthner, editors. *Handbook of Philosophical Logic (II)*, *Synthese Library*, vol. 165. Reidel, 1984.
[111] D. Gabbay and I. Hodkinson. An axiomitization of the temporal logic with until and since over the real numbers. *Journal of Logic and Computation*, 1(2):229–259, 1990.
[112] D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, vol. 1. Clarendon Press, Oxford, 1994.

[113] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. The temporal analysis of fairness. In *Proc. 7th ACM Symposium on the Principles of Programming Languages (POPL)*, pages 163–173, Las Vegas, Nevada, January 1980.
[114] D. Gabbay, M. Reynolds, and M. Finger. *Temporal Logic: Mathematical Foundations and Computational Aspects*, vol. 2. Clarendon Press, Oxford, 2000.
[115] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev. Combining spatial and temporal logics: expressiveness vs. complexity. *Journal of Artificial Intelligence Research (JAIR)*, 23:167–243, 2005.
[116] A.P. Galton. *The Logic of Aspect*. Clarendon Press, Oxford, 1984.
[117] A.P. Galton. A critical examination of Allen's theory of action and time. *Artificial Intelligence*, 42:159–188, 1990.
[118] A.P. Galton. An investigation of 'Non-intermingling' principles in temporal logic. *Journal of Logic and Computation*, 6(2):271–294, 1996.
[119] A.P. Galton. Eventualities. In Fisher et al. [100], pages 25–58.
[120] J. Garson. Quantification in modal logic. In Gabbay and Guenthner [110], Chapter II.6, pages 249–307.
[121] A. Gerevini. Processing qualitative temporal constraints. In Fisher et al. [100], pages 247–278.
[122] R. Gerth, D. Peled, M.Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proc. 15th IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification (PSTV)*, *IFIP Conference Proceedings*, vol. 38, pages 3–18. Chapman & Hall, 1996.
[123] R. Goldblatt. *Logics of Time and Computation*. CSLI Lecture Notes Stanford, CA, 1987.
[124] R. Gómez and H. Bowman. PITL2MONA: implementing a decision procedure for propositional interval temporal logic. *Journal of Applied Non-Classical Logics*, 14(1–2):105–148, 2004.
[125] V. Goranko. Temporal logic with reference pointers. In ICTL'94 [158], pages 133–148.
[126] V. Goranko, A. Montanari, and G. Sciavicco. A general tableau method for propositional interval temporal logics. In *Proc. International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, *Lecture Notes in Computer Science*, vol. 2796, pages 102–116. Springer, 2003.
[127] V. Goranko, A. Montanari, and G. Sciavicco. A road map of interval temporal logics and duration calculi. *Journal of Applied Non-Classical Logics*, 14(1–2):9–54, 2004.
[128] G.D. Gough. Decision procedures for temporal logic. Master's thesis, Department of Computer Science, University of Manchester, UK, October 1984.
[129] P. Gribomont and P. Wolper. In *From Modal Logic to Deductive Databases: Introducing a Logic Based Approach to Artificial Intelligence*, pages 165–234. Wiley, 1989 (chapter *Temporal Logic*).
[130] D. Guelev. A complete proof system for first-order interval temporal logic with projection. *Journal of Logic and Computation*, 14(2):215–249, 2004.
[131] D. Guelev and D. van Hung. A relatively complete axiomatisation of projection onto state in the duration Calculus. *Journal of Applied Non-Classical Logics*, 14(1–2):149–180, 2004.
[132] Y. Gurevich and S. Shelah. The decision problem for branching time logic. *Journal of Symbolic Logic*, 50(3):668–681, 1985.

[133] S. Haack. *Philosophy of Logics*. Cambridge University Press, 1978.
[134] R. Hale and B. Moszkowski. Parallel programming in temporal logic. In *Proc. Parallel Architectures and Languages Europe (PARLE)*, *Lecture Notes in Computer Science*, vol. 259, pages 277–296. Springer, 1987.
[135] J. Halpern, Z. Manna, and B. Moszkowski. A hardware semantics based on temporal intervals. In *Proc. International Colloquium on Automata Languages and Programming (ICALP)*, *Lecture Notes in Computer Science*, vol. 154, pages 278–291. Springer-Verlag, 1983.
[136] J. Halpern and Y. Shoham. A propositional modal logic of time intervals. *ACM Journal*, 38(4):935–962, 1991.
[137] S. Hanks and D. Madigan. Probabilistic temporal reasoning. In Fisher et al. [100], pages 315–342.
[138] P. Hasle and P. Øhrstrøm. Foundations of temporal logic—The WWW-site for prior-studies. http://www.kommunikation.aau.dk/prior.
[139] J.G. Henriksen and P.S. Thiagarajan. Dynamic linear time temporal logic. *Annals of Pure and Applied Logic*, 96(1–3):187–207, 1999.
[140] T. Henzinger. It's about time: real-time logics reviewed. In *Proc. 9th International Conference on Concurrency Theory (CONCUR)*, *Lecture Notes in Computer Science*, vol. 1466, pages 439–454. Springer, 1998.
[141] T. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.
[142] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
[143] B. Hirsch and U. Hustadt. Translating PLTL into WS1S: Application description. In *Proc. Methods for Modalities (M4M) II*, Amsterdam, Netherlands, 2001.
[144] R. Hirsch. From points to intervals. *Journal of Applied Non-Classical Logics*, 4(1):7–27, 1994.
[145] R. Hirsch. Relation algebras of intervals. *Artificial Intelligence*, 83:1–29, 1996.
[146] I. Hodkinson. Monodic packed fragment with equality is decidable. *Studia Logica*, 72:185–197, 2002.
[147] I. Hodkinson and M. Reynolds. Separation—past, present, and future. In *We Will Show Them! Essays in Honour of Dov Gabbay*, vol. 2, pages 117–142. College Publications, 2005.
[148] I. Hodkinson and M. Reynolds. Temporal logic. In P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*. Elsevier, 2006 (Chapter 11).
[149] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 106:85–134, 2000.
[150] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable and undecidable fragments of first-order branching temporal logics. In *Proc. 17th IEEE Symposium on Logic in Computer Science (LICS)*, pages 393–402. IEEE Computer Society, 2002.
[151] G.J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, Englewood Cliffs, NJ, 1991.
[152] G.J. Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997 (Special issue on Formal Methods in Software Practice).

[153] G.J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, November 2003.
[154] I. Horrocks. Description Logics, 2007. (*In this volume*).
[155] U. Hustadt, B. Konev, A. Riazanov, and A. Voronkov. TeMP: A temporal monodic prover. In *Proc. 2nd International Joint Conference on Automated Reasoning (IJCAR)*, *Lecture Notes in Artificial Intelligence*, vol. 3097, pages 326–330. Springer, 2004.
[156] Hybrid logics web page. http://hylo.loria.fr.
[157] HyTech: The HYbrid TECHnology tool. http://embedded.eecs.berkeley.edu/research/hytech.
[158] D. Gabbay and H.-J. Ohlbach, editors. In *Proc. First International Conference on Temporal Logic (ICTL)*. *Lecture Notes in Computer Science*, vol. 827. Springer, 1994.
[159] Interval temporal logic. http://www.cse.dmu.ac.uk/STRL/ITL//itlhomepage.html.
[160] Java PathFinder. http://javapathfinder.sourceforge.net.
[161] J. Kamp. Tense logic and the theory of linear order. PhD thesis, University of California, Los Angeles, May 1968.
[162] J. Kamp and U. Reyle. *From Discourse to Logic*. Kluwer, Dordrecht, 1993.
[163] S. Katz and D. Peled. Interleaving set temporal logic. *Theoretical Computer Science*, 75(3):263–287, 1990.
[164] Y. Kesten, Z. Manna, and A. Pnueli. Temporal verification of simulation and refinement. In *A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium*, *Lecture Notes in Computer Science*, vol. 803. Springer, 1994.
[165] Y. Kesten, Z. Manna, and A. Pnueli. Verification of clocked and hybrid systems. *Acta Informatica*, 36(11):837–912, 2000.
[166] Y. Kesten and A. Pnueli. Complete proof system for QPTL. *Journal of Logic and Computation*, 12(5):701–745, 2002.
[167] B. Konev, A. Degtyarev, C. Dixon, M. Fisher, and U. Hustadt. Mechanising first-order temporal resolution. *Information and Computation*, 199(1–2):55–86, 2005.
[168] R. Kontchakov, C. Lutz, F. Wolter, and M. Zakharyaschev. Temporalizing tableaux. *Studia Logica*, 76(1):91–134, 2004.
[169] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 1(4):67–95, 1986.
[170] R. Koymans. Specifying message passing systems requires extending temporal logic. In *Proc. Temporal Logic in Specification*, *Lecture Notes in Computer Science*, vol. 398, pages 213–223. Springer, 1987.
[171] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
[172] F. Kröger. *Temporal Logic of Programs*. *EATCS Monographs on Theoretical Computer Science*, vol. 8. Springer-Verlag, Berlin, 1987.
[173] Kronos tool. http://www-verimag.imag.fr/TEMPORISE/kronos/.
[174] M.Z. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. In *Proc. 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, *Lecture Notes in Computer Science*, vol. 2280, pages 52–66. Springer, 2002.

[175] P. Ladkin. The completeness of a natural system for reasoning with time intervals. In *Proc. 10th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 462–467, August 1987.
[176] P. Ladkin and R. Maddux. On binary constraint problems. *ACM Journal*, 41:435–469, 1994.
[177] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.
[178] L. Lamport. *Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.
[179] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *Proc. 17th IEEE Symposium on Logic in Computer Science (LICS)*, pages 383–392. IEEE Computer Society, 2002.
[180] K.G. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. In *Proc. Conference on Fundamentals of Computation Theory*, *Lecture Notes in Computer Science*, vol. 965, pages 62–88. August 1995.
[181] E.J. Lemmon and D. Scott. *An Introduction to Modal Logic*. *American Philosophical Quarterly, Monograph Series*, vol. 11. 1977. (Originally written in 1963 as a draft for a book.).
[182] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Proc. Workshop on Logics of Programs*, *Lecture Notes in Computer Science*, vol. 193, pages 196–218. Springer, 1985.
[183] G. Ligozat. Weak representation of interval algebras. In *Proc. 8th American National Conference on Artificial Intelligence (AAAI)*, pages 715–720, 1990.
[184] G. Ligozat. Tractable relations in temporal reasoning: pre-convex relations. In *Proc. ECAI Workshop on Spatial and Temporal Reasoning*, August 1994.
[185] F. Lin. Situation Calculus, 2007. (*In this volume*).
[186] C. Liu and M. Orgun. Dealing with multiple granularity of time in temporal logic programming. *Journal of Symbolic Computation*, 22(5–6):699–720, 1996.
[187] LPG: A fully-automated domain-independent planner for PDDL2.2 domains. http://zeus.ing.unibs.it/lpg.
[188] The logics workbench. http://www.lwb.unibe.ch.
[189] J. Ma and B. Knight. Reified temporal logics: an overview. *Artificial Intelligence Review*, 15(3):189–217, 2001.
[190] J. Ma, B. Knight, and T. Peng. Temporal reasoning about action and change. In K. Anjaneyulu, M. Sasikumar, and S. Ramani, editors. *Knowledge Based Computer Systems—Research and Applications*, pages 193–204. Narosa Publishing House, 1996.
[191] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992.
[192] Z. Manna and A. Pnueli. *The Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.
[193] M. Marx and M. Reynolds. Undecidability of compass logic. *Journal of Logic and Computation*, 9:897–914, 1999.
[194] W. May and P.H. Schmitt. A tableau calculus for first-order branching time logic. In *Proc. International Conference on Formal and Applied Practical Reasoning (FAPR)*, *Lecture Notes in Computer Science*, vol. 1085, pages 399–413. Springer, 1996.

[195] M.C. Mayer and S. Cerrito. Variants of first-order modal logics. In *Proc. International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, *Lecture Notes in Computer Science*, vol. 1847, pages 175–189. Springer, 2000.
[196] M.C. Mayer, A. Orlandini, G. Balestreri, and C. Limongelli. A planner fully based on linear time logic. In *Proc. AI Planning Systems*, pages 347–354, 2000.
[197] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.
[198] K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.
[199] S. Merz. Decidability and incompleteness results for first-order temporal logic of linear time. *Journal of Applied Non-Classical Logics*, 2:139–156, 1992.
[200] Mocha: Exploiting modularity in model checking. http://www.cis.upenn.edu/~mocha.
[201] F. Moller and A. Rabinovich. Counting on CTL*: on the expressive power of monadic path logic. *Information and Computation*, 184(1):147–159, 2003.
[202] A. Montanari. Metric and layered temporal logic for time granularity. PhD thesis, University of Amsterdam, Amsterdam, Netherlands, September 1996. ILLC Dissertation Series 1996-02.
[203] Y. Moses. Reasoning about Knowledge and Belief, 2007. (*In this volume*).
[204] B. Moszkowski. Reasoning about digital circuits. PhD thesis, Computer Science Department, Stanford University, 1983.
[205] B. Moszkowski. *Executing Temporal Logic Programs*. Cambridge University Press, Cambridge, UK, 1986.
[206] B. Moszkowski. Compositional reasoning about projected and infinite time. In *Proc. 1st IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 238–245. IEEE Computer Society Press, 1995.
[207] B. Moszkowski. Compositional reasoning using interval temporal logic and tempura. In *Compositionality: The Significant Difference*, *Lecture Notes in Computer Science*, vol. 1536, pages 439–464. Springer, 1998.
[208] B. Moszkowski and Z. Manna. Reasoning in interval temporal logic. In *Proc. AMC/NSF/ONR Workshop on Logics of Programs*, *Lecture Notes in Computer Science*, vol. 164, pages 371–383. Springer-Verlag, 1984.
[209] E. Mueller. Event Calculus, 2007. (*In this volume*).
[210] A. Nonnengart. Resolution-based calculi for modal and temporal logics. In *Proc. 13th International Conference on Automated Deduction (CADE)*, *Lecture Notes in Artificial Intelligence*, vol. 1104, pages 598–612. Springer, 1996.
[211] NuSMV: A new symbolic model checker. http://nusmv.irst.itc.it.
[212] H.-J. Ohlbach. Translation methods for non-classical logics—an overview. *Bulletin of the Interest Group in Propositional and Predicate Logics (IGPL)*, 1(1):69–90, 1993.
[213] H.-J. Ohlbach and D. Gabbay. Calendar logic. *Journal of Applied Non-Classical Logics*, 8(4):291–324, 1998.
[214] M. Orgun and W. Ma. An overview of temporal and modal logic programming, In ICTL'94 [158], pages 445–479.
[215] M. Orgun and W. Wadge. Theory and practice of temporal logic programming. In L. Fariñas del Cerro and M. Penttonen, editors. *Intensional Logics for Programming*. Oxford University Press, 1992.

[216] J. Ostroff. *Temporal Logic of Real-Time Systems*. Research Studies Press, 1990.
[217] J. Ostroff. Formal methods for the specification and design of real-time safety critical systems. *Journal of Systems and Software*, 18(1):33–60, 1992.
[218] W. Penczek. Axiomatizations of temporal logics on trace systems. *Fundamenta Informaticae*, 25(2):183–200, 1996.
[219] W. Penczek and A. Polrola. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*. *Studies in Computational Intelligence*, vol. 20. Springer, 2006.
[220] R. Pliuskevicius. The saturated tableaux for linear miniscoped Horn-like temporal logic. *Journal of Automated Reasoning*, 13:391–407, 1994.
[221] R. Pliuskevicius. On an *omega*-decidable deductive procedure for non-Horn sequents of a restricted FTL. In *Proc. 1st International Conference on Computational Logic*, *Lecture Notes in Computer Science*, vol. 1861, pages 523–537. Springer, 2000.
[222] A. Pnueli. The temporal logic of programs. In *Proc. 18th Symposium on the Foundations of Computer Science (FOCS)*, Providence, USA, November 1977.
[223] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.
[224] A. Pnueli and T. Arons. TLPVS: A PVS-based LTL verification system. In *Verification: Theory and Practice*, *Lecture Notes in Computer Science*, vol. 2772, pages 598–625. Springer, 2003.
[225] A. Pnueli and Y. Kesten. A deductive proof system for CTL. In *Proc. 13th International Conference on Concurrency Theory (CONCUR)*, *Lecture Notes in Computer Science*, vol. 2421, pages 24–40. Springer, 2002.
[226] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symposium on Principles of Programming Languages (POPL)*, pages 179–190, New York, 1989.
[227] M.R. Prasad, A. Biere, and A. Gupta. A survey of recent advances in SAT-based formal verification. *International Journal on Software Tools for Technology Transfer (STTT)*, 7(2):156–173, 2005.
[228] I. Pratt-Hartmann. Temporal prepositions and their logic. *Artificial Intelligence*, 166(1–2):1–36, 2005.
[229] A. Prior. *Past, Present and Future*. Clarendon Press, Oxford, UK, 1967.
[230] PRISM: Probabilistic symbolic model checker. http://www.cs.bham.ac.uk/~dxp/prism.
[231] PSL/Sugar Consortium Web Page. http://www.pslsugar.org.
[232] G. Puppis. Automata for branching and layered temporal structures. PhD thesis, Department of Mathematics and Computer Science, University of Udine, Italy, 2006.
[233] The PVS specification and verification system. http://pvs.csl.sri.com.
[234] Rabbit timed automata. http://www-sst.informatik.tu-cottbus.de/~db/Rabbit.
[235] The RED (Region Encoding Diagram) system. http://cc.ee.ntu.edu.tw/~farn/red.
[236] H. Reichgelt and L. Vila. Temporal qualification in artificial intelligence. In Fisher et al. [100], pages 167–196.
[237] R. Reiter. *Knowledge in Action*. MIT Press, 2001.
[238] N. Rescher and A. Urquart. *Temporal Logic*. Springer-Verlag, 1971.
[239] M. Reynolds. Axiomatisation and decidability of F and P in cyclical time. *Journal of Philosophical Logic*, 23:197–224, 1994.

[240] M. Reynolds. More past glories. In *Proc. IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Press, 2000.
[241] M. Reynolds. An axiomatization of full computation tree logic. *Journal of Symbolic Logic*, 66(3):1011–1057, 2001.
[242] M. Reynolds. An axiomatization of PCTL*. *Information and Computation*, 201(1):72–119, 2005.
[243] M. Reynolds and C. Dixon. Theorem-proving for discrete temporal logic. In Fisher et al. [100], pages 279–314.
[244] J.A. Robinson. A machine based logic based on the resolution principle. *ACM Journal*, 12(1):23–41, January 1965.
[245] R. Rosner and A. Pnueli. A choppy logic. In *Proc. IEEE Symposium on Logic in Computer Science (LICS)*, pages 306–313. IEEE Computer Society, 1986.
[246] M. Sabbadin and A. Zanardo. Topological aspects of branching-time semantics. *Studia Logica*, 75(3):271–286, 2003.
[247] K.D. Schild. Combining terminological logics with tense logic. In *Progress in Artificial Intelligence—Proc. 6th Portuguese Conference on Artificial Intelligence (EPIA)*, *Lecture Notes in Computer Science*, vol. 727, pages 105–120. Springer, 1993.
[248] B.-H. Schlingloff. Expressive completeness of temporal logic of trees. *Journal of Applied Non-Classical Logics*, 2(2), 1992.
[249] P.-Y. Schobbens, J.-F. Raskin, and T. Henzinger. Axioms for real-time logics. *Theoretical Computer Science*, 274(1–2):151–182, 2002.
[250] L. Schubert. Natural Language Processing, 2007. (*In this volume*).
[251] E. Schwalb and L. Vila. Temporal constraints: a survey. *Constraints*, 3(2–3):129–149, 1998.
[252] R.L. Schwartz, P.M. Melliar-Smith, and F.H. Vogt. An interval-based temporal logic. In *Proc. AMC/NSF/ONR Workshop on Logics of Programs*, *Lecture Notes in Computer Science*, vol. 164, pages 443–457. Springer, June 1984.
[253] S. Schwendimann. A new one-pass tableau calculus for PLTL. In *Proc. Workshop on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, *Lectures Notes in Computer Science*, vol. 1397, pages 277–291. Springer-Verlag, 1998.
[254] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.
[255] The SLAM Project: Debugging system software via static analysis. http://research.microsoft.com/slam.
[256] On-the-fly, LTL model checking with SPIN. http://spinroot.com/spin/whatispin.html.
[257] M. Steedman. Dynamic semantics for tense and aspect. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1292–1298, 1995.
[258] M. Steedman. Temporality. In J.F.A.K. van Benthem and A. Ter Meulen, editors. *Handbook of Logic and Language*. MIT Press, 1997.
[259] The Stanford Temporal Prover. http://www-step.stanford.edu.
[260] C. Stirling. Modal and temporal logics. In S. Abramsky, D. Gabbay, and T. Maibaum, editors. *Handbook of Logic in Computer Science*. Oxford University Press, 1992.

[261] SystemVerilog assertions. http://www.eda-stds.org/sv-ac.
[262] A. Szalas. Arithmetical axiomatisation of first-order temporal logic. *Information Processing Letters*, 26:111–116, November 1987.
[263] A. Szalas and L. Holenderski. Incompleteness of first-order temporal logic with until. *Theoretical Computer Science*, 57:317–325, 1988.
[264] Temporal data mining website. http://www.temporaldatamining.com.
[265] A. ter Meulen. *Representing Time in Natural Language: The Dynamic Interpretation of Tense and Aspect*. The MIT Press, Cambridge, MA, 1995.
[266] A. ter Meulen. Temporal reasoning in natural language. In Fisher et al. [100], pages 559–586.
[267] D. Thérien and T. Wilke. Nesting until and since in linear temporal logic. *Theory of Computing Systems*, 37(1):111–131, 2004.
[268] P.S. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for Mazurkiewicz traces. *Information and Computation*, 179(2):230–249, 2002.
[269] J. Thornton, M. Beaumont, A. Sattar, and M.J. Maher. Applying local search to temporal reasoning. In *Proc. International Symposium on Temporal Representation and Reasoning (TIME)*, pages 94–99, 2002.
[270] The TLPVS WWW page. http://www.wisdom.weizmann.ac.il/~verify/tlpvs/index.shtml.
[271] P. Traverso. Planning, 2007. (*In this volume*).
[272] S. Tripakis, S. Yovine, and A. Bouajjani. Checking timed Büchi automata emptiness efficiently. *Formal Methods in System Design*, 26(3):267–292, 2005.
[273] TSQL2 Temporal Query Language. http://www.cs.arizona.edu/~rts/tsql2.html.
[274] J.F.A.K. van Benthem. Correspondence theory. In Gabbay and Guenthner [110], Chapter II.4, pages 167–248.
[275] J.F.A.K. van Benthem. *The Logic of Time*. 2nd edition. Kluwer Academic Publishers, 1991.
[276] W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.
[277] W. van der Hoek and M. Wooldridge. Multi-Agent Systems, 2007. (*In this volume*).
[278] M.Y. Vardi. A temporal fixpoint calculus. In *Proc. 15th ACM Symposium on Principles of Programming Languages (POPL)*, pages 250–259, 1988.
[279] M.Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency—Structure versus Automata (Proc. 8th Banff Higher Order Workshop)*, *Lecture Notes in Computer Science*, vol. 1043, pages 238–266. Springer, 1996.
[280] M.Y. Vardi. Alternating automata: unifying truth and validity checking for temporal logics. In *Proc. 14th International Conference on Automated Deduction (CADE)*, *Lecture Notes in Computer Science*, vol. 1249, pages 191–206. Springer, 1997.
[281] M.Y. Vardi. Branching vs. linear time: final showdown. In *Proc. 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, *Lecture Notes in Computer Science*, vol. 2031, pages 1–22. Springer-Verlag, 2001.

[282] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.
[283] Y. Venema. A logic with the chop operator. *Journal of Logic and Computation*, 1:453–476, 1991.
[284] G. Venkatesh. A decision method for temporal logic based on resolution. In *Proc. Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, *Lecture Notes in Computer Science*, vol. 206, pages 272–289. Springer-Verlag, Berlin–Heidelberg–New York, 1986.
[285] Verics system. http://www.ipipan.waw.pl/~penczek/abmpw/verics-ang.htm.
[286] V. Vidal and H. Geffner. Branching and pruning: an optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170(3):298–335, 2006.
[287] L. Vila. Formal theories of time and temporal incidence. In Fisher et al. [100], pages 1–24.
[288] L. Vila and H. Yoshino. Time in automated legal reasoning. In Fisher et al. [100], pages 537–558.
[289] W. Visser, K. Havelund, G. Brat, and S. Park. Model checking programs. In *Proc. International Conference on Automated Software Engineering (ASE)*, 2000.
[290] A. Voronkov. First-Order Reasoning, 2007. (*In this volume*).
[291] T. Walsh. Constraint Satisfaction, 2007. (*In this volume*).
[292] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.
[293] P. Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, 110–111:119–136, June–Sept. 1985.
[294] F. Wolter and M. Zakharyaschev. Temporalizing description logics. In *Proc. 2nd International Workshop on Frontiers of Combining Systems (FroCoS)*, Amsterdam, NL, 1998.
[295] F. Wolter and M. Zakharyaschev. Axiomatizing the monodic fragment of first-order temporal logic. *Annals of Pure and Applied Logic*, 118(1–2):133–145, 2002.
[296] C. Zhou, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.