Real-time and Probabilistic Temporal Logics: An Overview*

Savas Konur

Department of Computer Science
University of Liverpool
November 2008

* This research was supported by EPSRC under the grant EP/F033567/1.

Contents

Abstract

In this paper we analyse various temporal formalisms, including propositional/first-order linear temporal logics, branching temporal logics, partial-order temporal logics, interval temporal logics, real-time temporal logics and probabilistic temporal logics. We extrapolate the notions of decidability, axiomatizability, expressiveness, model checking, etc. for each logic analysed. We also provide a comparison of features of the temporal logics discussed.

1 Introduction

The number of formalisms that facilitate modelling, specifying, and proving timing properties of real-time systems has exploded over the last decade. Specification of real-time systems must be supported by formal, mathematically founded methods in order to be satisfactory and reliable. Temporal logics have been used for this purpose for several years.

Temporal logics allow the specification of system behaviour in terms of logical formulas, including temporal constraints, events, and the relationships between the two. In most cases, temporal logics have been defined to satisfy specific needs. In recent years the structure and capabilities of temporal logics have grown (although not all are suitable for specifying real-time systems). In some cases, simple temporal logics are preferable to more complex and powerful ones, since the former are more satisfactorily adopted in certain applications.

In this survey we review a number of well-known temporal logics designed for the specification of real-time systems. All these logics are different in terms of expressiveness, order, presence of a metric for time, the type of temporal operators, the fundamental time entity and the structure of time. They also have different capabilities for the specification, validation, and verification of real-time systems.

As well as the basic temporal framework, we also survey two important areas of extension: 'real-time' and 'probability'. Real-time aspect of temporal logics is quite crucial for the specification and verification of *functional requirements*[1] for software embedded hard real-time systems.

The need for reasoning about probability arises in many areas of research. In computer science we analyse randomised or probabilistic programs, reason about the behaviour of a program under probabilistic assumptions about the input, reason about uncertain information in an expert system or deal with *dependability requirements*[2]. The underlying mathematical foundation for reasoning about probability is the theory of probability and stochastic processes (e.g. Markov processes). In order to carry out formal reasoning about probability, it is helpful to have a logic for reasoning about probability with a well-defined syntax and semantics. Having such a logic might also clarify the role of probability in the analysis.

[1]The *functional requirements* express what a system must be able to do and what it must not to do.

[2]The *dependability requirements* express that the probability for undesirable but unavidable behaviour of a system must be below a certain limit.

2 Classical Temporal Logic

Temporal logics focus on statements whose truth values depend on time. Temporal statements typically contain some reference to time conditions, while classical logic deals with timeless entities. While classical logic formulas can characterise static states and properties, temporal logic formulas can describe dynamic state changes and properties of behaviours, and, hence, can span a wide range of problems in various fields with a richer notation.

A temporal logic basically results from an extension of a classical propositional or predicate logic with temporal quantifiers introducing temporal modalities. Due to its temporal quantifiers, temporal logic is a convenient and appropriate means to reason with time-related statements. Indeed, classical logic can also handle temporal properties, but the formulas tend to be complicated since points of time have to be explicitly represented, as a separate sort in the underlying universe.

In modal logics, statements are evaluated with respect to certain *worlds*. Temporal logics are particular modal logics where the set of worlds $\mathcal{W}$ is interpreted as the set of all possible instants $\mathcal{T}$ of a temporal domain. Temporal logics are usually built as extensions of classical logic by adding a set of new operators that hide quantification over the temporal domain. Temporal logics in the literature are principally obtained by extending propositional or first-order logic.

As in modal logic, where the world in which the formula is evaluated is referenced, in temporal logics the evaluation instant of a formula is used. The value of a formula is a dynamic concept. Therefore, the concept of formula satisfiability must be modified to consider both the interpretation of a formula and the instant of evaluation.

A particular system of temporal logic called *Tense Logic* introduced in [Pri67] adds four new unary operators to classical logics:

- P: "It has at some time been the case."
- F: "It will at some time be the case."
- H: "It has always been the case."
- G: "It will always be the case."

P and F are known as the *weak tense operators*, while H and G are known as the *strong tense operators*. G and H are duals of F and P, respectively. Therefore, $F\phi \equiv \neg G \neg \phi$ and $P\phi \equiv \neg H \neg \phi$, where ϕ is a formula.

Soon after the introduction of Tense Logic, its basic "$PFGH$" syntax was extended in various ways. Some important examples are the following:

The binary temporal operators S and U ("since" and "until"). These were introduced by Kamp [Kam68]. The intended meanings are:

- Spq: "q has been true since a time when p was true"
- Upq: "q will be true until a time when p is true"

The importance of the S and U operators is that they are expressively complete with respect to first-order temporal properties on continuous, strictly linear temporal orders (which is not true for the one-place operators on their own) [Kam68]. The

introduction of operators S and U is relevant because these operators can express concepts that cannot be expressed with the operators G, H, F and P. On the contrary, these operators can be defined in terms of S and U:

- $Pp \equiv Sp(p \vee \neg p)$
- $Fp \equiv Up(p \vee \neg p)$

Other common operators are *next* and *prev*. These operators are unary and can be defined in terms of the S and U operators. These two operators assume different meanings depending on the time structure, for example, discrete or continuous, or whether the logic is event-based.

The presence of distinct operators for the past and future simplifies the specification model. On the other hand, this distinction is only a convention, since in most temporal logics formulas can easily be shifted to the past or to the future.

Similar to the extension of propositional logic with the temporal operators S and U one can extend predicate logic with these operators to get a first-order temporal logic. In first-order temporal logics, problems arise because of the interplay between quantification and time [Koy90]. One of these problems is the possibility that the quantified variables (and possibly even their value domains) change over time. This problem is avoided by only allowing quantification over variables that do not change over time. Even in this restricted case most first-order temporal logics are incomplete. One important exception is the monadic fragment of first-order temporal logic, which is shown to be complete [WZ01].

3 Classification of Temporal Logics

Temporal logic systems can be classified along various dimensions: propositional versus first-order, point-based versus interval-based, linear versus branching, discrete versus continuous, etc [Eme95, Ven98, BMN00]. In this section we provide a taxonomy and evaluation criteria to classify and compare temporal logics. Below we discuss the most important features of temporal logics and the criteria used to identify their general characteristics and properties.

3.1 Fundamental Entities of the Logic

Choosing a mathematical model of time has been a primary concern in philosophy. A basic way to characterise temporal logics is whether *points (instants)* or *intervals* are used to model time.

Time, based on the point-based paradigm, can be formally represented as a structure $\mathcal{T} = \langle T, < \rangle$ such that T is a non-empty set of time points, and $<$ is a binary relation on T. Time durations are expressed by using quantification over time. Logics based on time points [MP83] specify system behaviour with respect to certain reference points in time; points are determined by a specific state of the system and by the occurrence of events marking state transition. In order to describe temporal relationships, some modal operators are adopted. In point-based logics it is more difficult to express relationships between intervals in which certain events are verified. The

difficulties associated with modelling the refinement of a system specification using a point-based temporal logic are widely recognised as an important problem [FM94].

Interval-based temporal logics (interval logics) are more expressive, since they are capable of describing events in time intervals. Usually, interval-based logics permit one to write formulas with a greater level of abstraction, and so are more concise and easy to understand than point-based temporal logics. In the case of time intervals [SMS82, SMSV83, Mos83, Lad87, MS87, RG89, HS91] formulas specify the temporal relationships among facts, events, and intervals, thus allowing a higher level of abstraction for system specification. Interval-based logics usually present specific operators to express the relationships between intervals (*meets*, *before*, *after, etc.* [All83]), operators to combine intervals (e.g., the *chop* operator [RP86]), or operators to specify the interval boundaries on the basis of the truth of predicates [MS87].

There are two approaches to translate the interval notion into a precise mathematical formalism: *(i)* either intervals are primitive objects of the model and studied on their own, without referring to their internal structure, or *(ii)* they are built up from a traditional point-based temporal domain.

The first approach has been followed in [vB91], where a 'period structure' has been studied and analysed. Interval structures are formally represented by $\langle \mathcal{I}, \subseteq, \prec \rangle$, where $\mathcal{I}$ is a non-empty set of atomic objects called 'intervals', $\subseteq$ is a sub-interval relation, and $\prec$ is a precedence relation ("it is entirely before"). [MSV02, Vit05] consider the notion of interval, similarly. While this approach seems cleaner from a "philosophical" point of view, it turns out that the basic principles needed to directly define an interval structure are more involved than the ones for point-based structures. Furthermore, many usual properties of flows of time, like *linearity*, *density*, and *discreteness*, which are easily defined in terms of points do not directly transfer to intervals.

The latter approach is more common in the interval logics literature [GMS04, HS91, Ven91]. In this view, an underlying flow of time is modelled as a strict partial ordering of time points, while intervals are defined as sets of time points satisfying some particular constraints. By doing so, all the usual properties of strict orderings (like *linearity*, *density*, *discreteness*, *unboundedness*, . . .) can be defined and transferred to interval structures. Given a strict partial ordering $\mathcal{T} = \langle T, < \rangle$ (T is a non-empty set of time points, and $<$ is a binary relation on T), an interval structure can be formally represented by $\langle \mathcal{T}, \mathcal{I}(\mathcal{T}) \rangle$, where $\mathcal{I}(\mathcal{T})$ is a set of intervals.

We conclude this section with the historical development of interval-based approach. The concept of time intervals was first studied by Walker [Wal47]. Walker considered a non-empty set of intervals, which is ordered by a partial ordering relation. However, his work does not cover aspects of temporal logic in a general sense. In [Ham71] interval ontology was analysed philosophically. In [Hum79] an interval tense logic which is based on sub-interval relations was introduced. Dowty emphasised that human language and reasoning have an interval-based semantics rather than point-based one, and he worked on interval-based temporal languages [Dow79]. Similar works in natural languages, such as axiomatic systems for interval-based temporal logics, persistency, homogeneity, were done by [Kam79, Rop80, Bur82, vB83]. In philosophical logic Simons and Galton argued that temporal constructions of natural language required interval-based approach rather than point-based approach [Gal84, Sim87].

Interval-based temporal logics have played an important role in reasoning in artificial intelligence. Some important research has been done within this field by Allen [All83, All84, AH89, AF94].

This work mainly includes thirteen interval relations, known as Allen's relations, axiomatisation and representation of interval structures, and interval-based theory of actions and events. Ladkin worked on completeness theorem and satisfiability algorithms for Allen's logic [Lad87]. Galton pursued a further study on Allen's works [Gal90]. Interval based-logics have been applied to other fields in computer science. In [Par78, Pra79, HPS83] some work on process logic can be found. In process logic intervals represent pieces of information. Another important work was the development of *interval temporal logic (ITL)*, and its application to design of hardware components [Mos83, HMM83]. ITL had an important impact in temporal logic studies. Various variations have been proposed so far. In particular, Duration Calculus is an extension of interval temporal logic with a calculus to specify and reason about properties of state durations [CHR91].

3.2 Temporal Domain Structure

In a modal logic, interpreted over some Kripke structure $\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$, the main properties are related to the properties of the relation $\mathcal{R}$. Similarly, the structure of temporal domains is derived from properties of relationship $\mathcal{R}$. For temporal logics, the relation $\mathcal{R}$ is called a *precedence relation*, and is denoted by $<$. Some of the important properties of the temporal domain structure are given below:

In a temporal logic the structure of time is *linear* if any two points are comparable. Mathematically, a strict partial ordering is called linear if any two distinct points satisfy the condition: $\forall x, y : x < y \vee x = y \vee x > y$. In linear temporal logics there is only one possible successor for each time instant. A temporal logic is called *linear time logic* if the structure of time is linear.

A particular class of structures is the *branching-time structures*. The underlying temporal structure of branching-time temporal logics has a branching-like nature where each time point may have many successor points. The structure of time thus corresponds to an infinite tree. A *tree* is a set of time points T ordered by a binary relation $<$ which satisfies the following requirements [GHR94]:

- $\langle T, < \rangle$ is irreflexive;
- $\langle T, < \rangle$ is transitive;
- $\forall t, u, v \in T$ $u < t$ and $v < t \rightarrow u < v, u = v$ or $u > v$ (i.e. the past of any point is linear);
- $\forall x, y \in T, \exists z \in T$ such that $z < x$ and $z < y$ (i.e. $\langle T, < \rangle$ is connected).

A temporal logic is called a *branching time logic* if it is interpreted over a branching Kripke structure. In a linear time logic temporal modalities describe events along a single time line. In contrast, in branching time logic systems modalities allow quantification over possible future paths.

A temporal domain is *discrete* with respect to the relation $<$ if each non-final point is followed by a next point or an immediate successor. This can be formulated in first-order logic: $\forall x, y\ (x < y \rightarrow \exists z\ (x < z \wedge \neg\exists w(x < w \wedge w < z)))$. Discrete (linear) time is a model isomorphic to a discrete series of natural numbers, such that the instants x, y, z are positive integers. In most temporal logics used for program reasoning, time is discrete where the present instant corresponds to the program's current state and the next instant corresponds to the program's immediate successor state. Thus, the temporal structure corresponding to a sequence of states of a program execution is the nonnegative integers.

A temporal domain is *dense* if, between any two distinct points, we can find another different point. This can be mathematically formulated as $\forall x, y(x < y \rightarrow \exists z(x < z < y))$. Rational or the real numbers are quite convenient to represent the flow of dense time, therefore to model arbitrary small steps in time. This is useful, for example, in capturing the notion of movement.

It is noteworthy to mention that there is a distinction between *density* and *continuity*. A model of dense time is isomorphic to a dense series of rational numbers, meaning that there is always a rational number between any two rational numbers; whereas a model of *continuous* time is isomorphic to a continuous series of real numbers. Arithmetical continuity defines a real number as the limit of an infinite series of rational numbers, such that the infinite set of limits is a continuous set of real numbers. Suppose that the set of rational numbers is cut into a left and a right half, of numbers smaller and bigger than $\sqrt{2}$, respectively. Such a cut, without a proper point on either edge, is called a gap, and a flow of time is called continuous if it has no gaps. $\mathbb{Q}$ thus forms the standard counterexample, whereas $\mathbb{R}$ and $\mathbb{Z}$ are continuous [Ven98].

A temporal domain is *bounded above* (*bounded below*) if the temporal domain is bounded in the future (past); i.e. $\exists x \neg\exists y.x < y$ ($\exists x \neg\exists y.y < x$). Similarly, a temporal domain is *unbounded above* (*unbounded below*) if every point has a successor (predecessor); i.e. $\forall x \exists y.x < y$ ($\forall x \exists y.y < x$). Domains that are bounded below correspond to many useful system specifications.

A temporal domain *is Dedekind complete* if every non-empty and bounded above set of points has a least upper bound.

The temporal domain may be limited in the past and/or in the future or be unlimited, and it may be dense or discrete. Thus, the temporal structure may be linear or branched in the past and/or in the future. These properties have implications for the decidability of the logic, its executability, and the style used to write formulas.

4 Propositional Temporal Logics

An important success in temporal logic study was the introduction of the temporal operators into linear-time temporal logic by Kamp [Kam68]. In [Pnu77] Pnueli introduced a very unfluental *Linear Temporal Logic (LTL)*. LTL can express properties of *future* paths in a computation tree. In particular, properties such as "for some state on the path" or "for every two consecutive states" can be expressed. In [SC85] Sistla and Clarke proved that the problem of determining whether a Kripke structure fulfill-

ing an LTL formula is PSPACE-complete. They also showed that model checking for LTL is PSPACE-complete. It is also known that the fragment using only the "sometimes in the future" modality, denoted F, as well as the fragment using only the "next" modality, denoted X, have NP-complete satisfiability problems[SC85]. Nevertheless, the fragment when both F and X are allowed is PSPACE-complete.

In [GPSS80] the logic *Propositional Temporal Logic* was introduced (over discrete time models with "next" (X) and "until" (U) temporal operators. The models are infinite sequences of states with a first state, but no last state. A sound and complete axiomatic system for propositional temporal logic was provided in [GPSS80]. It was also shown that the logic is decidable and complete. In [LPZ85] PTL was extended with the past operators, and a complete proof system for both future and past operators was presented (A detailed discussion can be found in [Sza95, LP00].) PTL has a rich language. In fact, the following temporal operators can be semantically defined in PTL: *eventually* $(\overrightarrow{\Diamond}\phi)$, *once (at some time in the past)* $(\overleftarrow{\Diamond}\phi)$, *henceforth* $(\overrightarrow{\Box}\phi)$, *so far* $(\overleftarrow{\Box}\phi)$, *always* $(\overline{\Box}\phi)$, *sometimes* $(\overline{\Diamond}\phi)$, *until* $(\phi U\psi)$, *since* $(\phi S\psi)$, *immediately after* $(\overrightarrow{\bigcirc}\phi)$, *immediately before* $(\overleftarrow{\bigcirc}\phi)$, etc.

In the literature several examples of properties of programs expressible by means of temporal logics can be found [Kro87, MP81b, MP81a]. Some important properties are expressed in PTL as follows:

- $p \rightarrow \overrightarrow{\Box} q$ *(safety property)*: All states reached by a program after the state satisfying p do satisfy q.
- $\overrightarrow{\Box}\left((\neg q) \vee (\neg p)\right)$ *(safety property)*: The program cannot enter critical regions p and q simultaneously.
- $p \rightarrow \overrightarrow{\Diamond} q$: *(liveness property)*: There is a state reached by a program after the state satisfying p does satisfy q.
- $\overrightarrow{\Box}\overrightarrow{\Diamond} p \rightarrow \overrightarrow{\Diamond} q$: *(liveness property)*: If a request p is repeated, a response q is received.
- $\overrightarrow{\Box} p \rightarrow \overrightarrow{\Diamond} q$ *(liveness property)*: If a request p is hold permanently, a response q is received.

PTL is a decidable logic. An automata theoretic technique of obtaining satisfiability can be found in [SVW85]. In [SC85] it was found that the satisfiability problem for PTL is PSPACE-complete. In [LP00] an exponential time tableau algorithm, which has an overall time boundof $2^{O(|\phi|)}$, is presented for the validity problem for PTL, where ϕ is a PTL formula. [SC85] also provides a complete deductive system for proving the validity of PTL formulas. Among the proof systems existing in literature are Hilbert-style proof system [Lad87], a Gentzen-style proof system [Sza95] and a clausal resolution approach [Fis91, FDP01]. These proof systems are all sound and complete.

5 First-Order Temporal Logics

First-order temporal logics are extension of propositional temporal logics. Besides all features of propositional ones these logics also allow arbitrary data structures and quantifiers over individuals. First-order temporal logic systems have found numerous applications in both computer science and artificial intelligence: A typical application

is their usage in specification and verification of reactive systems. First-order logics provide more expressive and powerful tools for formalising the behaviour of executable temporal logics. They allow the extension of techniques for reasoning about knowledge to more dynamic and powerful classes [FHMV96, HWZ00]. First-order logic systems are also quite useful in information systems in the sense that query languages for temporal databases are often based on variants of FOTL [CT98].

Despite the usefulness in various areas, first-order temporal logics are very expressive languages with a very high computational complexity. Although some axiomatisations of first-order temporal logics were studied [Rey96], many varieties of first-order logics are not even recursively enumerable [Aba89, ANS79, GHR94, Mer92], and so do not admit finite proof methods at all. For that reason, most of the works in this field have mainly dealt with developing PTL-based tools. There are very few examples that recursively enumerable or decidable fragments of first-order temporal logics have been found. However, these variations were just small extensions to the propositional case. Some examples of these extensions are weaker versions of validity [Aba89], minimal interaction between quantifiers and temporal operators [Cho95] and very restricted first-order features [Mer92, Pli97].

One important development was done by Hodkinson, Wolter, and Zakharyaschev, who introduced a new natural *monodic* fragment of first-order temporal logic, and showed that it is quite expressive and has much better computational behaviour [HWZ00]. In monodic formulas, the scope of temporal operators is restricted only to subformulas with at most one free variable. The whole monodic fragment can be represented as a finite axiomatic system [WZ02], and so can be supported by tableau or resolution-type reasoning mechanism. Moreover, by restricting the first-order part to certain decidable fragments [ANvB98, Grä99], decidable monodic fragments of first-order temporal logic can be obtained over various flows of time.

The first-order temporal logic is represented by QTL, which is constructed from the following alphabet, which does not comprise equality and function symbols: *predicate symbols*: $P_0, P_1...$, *variables*: $x_0, x_1, ...$, *constants*: $c_0, c_1, ...$, *boolean connectives*: $\wedge$, $\neg$, *universal quantifier*: $\forall$, and *temporal operators*: $S(since)$ and $U(until)$. Let F be the underlying time structures assumed for QTL constitutes strict linear orders. Then, $QTL(F)$ denotes the first-order temporal logic of F, and $QTL_{fin}(F)$ denotes the logic of F with *finite domains*.

5.1 Undecidable Fragments of QTL

In the literature, it has been known that both the monadic and two-variable fragments of classical first-order logic are decidable [BGG97]. However, the computational complexities of their temporal counterparts are different. Let QTL^2 denote the *two - variable fragment* of QTL, and QTL^{mo} denote the *monadic fragment* (not monodic) of QTL, which respectively means that every formula in QTL^2 contains at most two distinct individual variables, and the set of formulas that contain only unary predicates and propositional variables. The theorems below show the complexities of these two fragments of QTL.

Let $\mathbb{T}$ be either $\{\langle \mathbb{N}, < \rangle\}$ or $\{\langle \mathbb{Z}, < \rangle\}$. Then $QTL^2 \cap QTL^{mo} \cap QTL(\mathbb{T})$ is not

recursively enumerable (see [HWZ00]). Let F be either $\{\langle \mathbb{N}, < \rangle\}$ or $\{\langle \mathbb{Z}, < \rangle\}$. Then $QTL^2 \cap QTL^{mo} \cap QTL_{fin}(F)$ is not recursively enumerable (see [HWZ00]).

5.2 Decidable Fragments of QTL

In the theorems given above there is a quantification in three 'dimensions', one temporal and two domain, since the linear time operator U can be applied to formulas with two free variables. This causes a problem that these fragments of QTL are undecidable. It is known that the three-variable fragment of classical first-order logic is undecidable [BGG97].

In order to avoid this problem corresponding fragment of QTL, which is QTL_1, contains all QTL-formulas φ such that any subformula of φ of the form $\psi_1 U \psi_2$ and $\psi_1 S \psi_2$ has at most one free variable. These formulas are *monodic* (not monadic) formulas, allowing quantification into temporal contexts only with one free variable. The monodic fragments of $QTL(\langle \mathbb{N}, < \rangle)$ and $QTL(\langle \mathbb{Z}, < \rangle)$ are recursively enumerable.

Let F be any of the following classes of flows of time: $\{\langle \mathbb{N}, < \rangle\}$, $\{\langle \mathbb{Z}, < \rangle\}$, $\{\langle \mathbb{Q}, < \rangle\}$ the class of all finite strict linear orders, any first-order-definable class of strict linear orders, and F^+ be F and $\{\langle \mathbb{R}, < \rangle\}$. Then, the following fragments are decidable : $QTL(F) \cap QTL^1$, $QTL(F) \cap QTL_1^2$, $QTL(F) \cap QTL_1^{mo}$, $QTL_{fin}(F^+) \cap QTL^1$, $QTL_{fin}(F^+) \cap QTL_1^2$, $QTL_{fin}(F^+) \cap QTL_1^{mo}$ (see [HWZ00]).

In [GKWZ02] it was shown that $QTL(\langle \mathbb{N}, < \rangle) \cap QTL^1$ is EXPSPACE-hard. It also follows that the satisfiability problem for QTL_1^{mo}-formulas in models based on $\langle \mathbb{N}, < \rangle$ is EXPSPACE-complete.

It has been assumed in this section that QTL and its fragments do not include *equality* and *function symbols*. It can be shown that undecidability is a major problem with the logic extended with function symbols [WZ02]. For example, the set of one-variable formulas with one function symbol that are valid in models based on $\langle \mathbb{N}, < \rangle$ is not recursively enumerable. Moreover, the set of monodic QTL formulas with equality that are valid in all temporal models based on $\langle \mathbb{N}, < \rangle$ is not recursively enumerable. [DFL02] proves that an even simpler fragment consisting of monodic monadic two-variable formulae is not recursively enumerable. In [WZ02] a finite *Hilbert-style axiomatisation* of monodic fragment of first-order temporal logic was constructed. It was also proved that the monodic fragment with equality is not recursively axiomatisable.

The decidability results can be extended to temporalties description logics. The resulting temporalties description logics are suitable for temporal conceptual modelling. These recent research results have showed that relatively expressive subsets of first-order temporal logic could be found. In [WZ99] certain similarities between monodic first-order temporal logic and effective multi-dimensional knowledge representation formalisms are described and it has been suggested that the monodic first-order temporal logic systems can be considerably extended. In [HWZ00, WZ02] there is a scope for enriching the expressive power of the monodic fragment. For example, applications of temporal operators, such as next-time, can be allowed to formulas with two or more free variables. Formulas of this form are particularly useful in temporal databases, and cover the decidable fragments of first-order temporal logic developed by Pliuskevicius [Pli97]. Some recent works present tableau-based satisfiability checking algorithms for

description logics with temporal and epistemic operators [LSWZ02]. Similarities between such logics and monodic temporal logics suggest that tableau-based reasoning systems can also be constructed for decidable monodic fragments. This can be carried out by combining existing tableau systems for PTL and the classical first-order components. It is also worth to mention that [DFKL08] introduces some resolution systems for monodic first-order temporal logics.

6 Branching Time and Partial-Order Temporal Logics

6.1 Branching Time Temporal Logics

A temporal logic system is called *Branching Time logic* if the underlying semantics of the structure of time is branching. The underlying structure of time in branching time is a tree-like structure. That is, every time instant may have several immediate successors which correspond to different futures.

Temporal logics with underlying branching time is fundamental to both computer science and artificial intelligence. Particularly, it has been widely used in AI applications. In planning systems agents formulate different plans and action strategies according to different future world states. Branching time temporal logics are very useful to model the reasoning of agents about the universe of possibilities in which branches represent choices of actions or plans [McD82, RG93]. Another important application of these logics is formalising specifications and behaviour of systems.

Branching time logics are usually used to verify finite state systems by model checking because of the efficiency of the corresponding model checking algorithms. Linear time logics usually come with a deductive proof system for dealing with the infinite state systems.

The first ideas about branching time logics appeared in [Abr80]. Later, The unified branching time system UB was defined [BAMP81]. A simple branching time logic, CTL, was introduced in [CE82]. Thereafter, CTL* was introduced in [EH86]. CTL* is an extension over CTL by adding the properties of linear time temporal logic. CTL*[P], an extension over CTL*, was introduced in [LS95]. UB, CTL and CTL* include only future time temporal connectives. In contrast, CTL*[P] contains both past and future time temporal connectives.

6.1.1 Computational Tree Logic (CTL)

CTL is an extension to the logic UB by adding a new path modality U (until). The fundamental temporal entity is the point, and presents specific operators for reasoning about the system behaviour in terms of several futures, called sequences. CTL does not provide an explicit metric for time. For verifying CTL specification a model checking approach is typically used since the specification can be modelled as a state machine. The language of CTL contains states formulas only.

Some examples of CTL formulas of CTL formulas are given below:

- EF(*Started* $\wedge\neg$*Ready*): It is possible to get to a state where *Started* holds but *Ready* does not hold.
- AG(*Req* $\Rightarrow$ AF *Ack*): If a request occurs, then it will be eventually Acknowledged.
- AG(EF *Restart*): From any state it is possible to get to the *Restart* state.

Although branching time logics can be defined in a tree-like structure, in many applications it is quite useful to define the logic in a different way than the tree structure. In this approach a set of states with a transition relation are considered as the basic object.

In [Pen95] a sound and complete axiomatic system is provided for CTL. It has to be shown that any consistent formula is satisfiable. The proof rests on constructing of a pseudo-Hintikka structure for a satisfiable CTL formula.

It can be shown that CTL has the finite model property. That is, if a formula ϕ is satisfiable, then it is satisfiable in a finite model whose size is bounded by some function of the length of the formula ϕ. Hence, a non-deterministic algorithm can determine the satisfiability of a CTL formula on a given structure in polynomial time. CTL is decidable [EH82]. There is also a tableau-based deterministic exponential time complete procedure for CTL satisfiability [EC82]. Efficient model checking algorithms exist for CTL (linear in the state space of the system model and the formula) [CES86].

It is noteworthy to mention that the logic UB has the finite model property, as well. It has a sound and complete axiomatisation system, and there is a deterministic exponential time lower bound for UB satisfiability.

6.1.2 Full Computational Tree Logic (CTL*)

The logic CTL* was introduced in [EH86]. CTL* is an extension over CTL by adding the properties of linear time temporal logic. There is an algorithm to decide the satisfiability of CTL* formula, which has a double exponential complexity in the length of the formula.

There are two kinds of CTL* formulas: *state* formulas and *path* formulas. State formulas are interpreted over states and path formulas, containing all state formulas, are interpreted over paths.

The axiomatizability of CTL* was an open question for a long time. A sound and complete axiomatisation for CTL* has recently been defined by Reynolds in [Rey01]. There is an algorithm to decide the satisfiability of CTL* formula, which has a double exponential complexity in the length of the formula [EJ00].

6.1.3 Full Computational Tree Logic with Past (CTL*[P])

As mentioned before there are two ways in formalising branching time temporal logics. In the semantics definitions CTL and CTL* we used states and paths as basic object types. In this section we will use discrete ω-height branching which is equivalent to state representation. That is, CTL*[P] formulas are evaluated at nodes on branches of labelled discrete rooted trees of height ω. Whereas the language of CTL* contains state and path formulas, some formulas of CTL*[P] do not depend on the path on which they are evaluated.

As in the linear case, addition of past operators to the language does not increase expressive power if we have a finite past. Decidability of CTL*[P] follows directly from the expressibility observation along with the decidability of CTL*. Until recently the axiomatizability of CTL*[P] has been a long-lasting open question. Reynolds gives a sound and complete axiomatisation system for CTL*[P] in [Rey05].

6.1.4 Other Branching Temporal Logics

In [EMSS89] a real-time extension of CTL, called RTCTL, was introduced. It is a bounded-operator extension of CTL with a point-based strictly-monotonic integer-time semantics. RTCTL includes a metric for time. The satisfiability problem for this logic is doubly-exponential-time-complete [EMSS89]. The model-checking has a polynomial time algorithm [EMSS89].

In [ACD90], a branching real-time time logic called TCTL is proposed. It is based on hidden clock bounded operators. TCTL is a bounded-operator extension of CTL with a point-based strictly-monotonic real-time semantics. For TCTL, the validity problem for dense time domains is undecidable, yet model-checking is decidable. The complexity of model checking is exponential in the number of clocks (each new process or hardware device needs its own clock), and doubly exponential in the product of the timing constants that appear in the formula. It is linear in the product of program and formula size.

Another branching time logic called TPCTL is introduced in [Han91]. The logic deals with real-time constraints and reliability. Because of the action based nature of TPCTL, it is difficult to specify state-based properties such as: “henceforth, if the train is at the crossing then the gate must be down”. Propositions such as “the gate is down” must be encoded indirectly through actions that change the state of the model, in which case the specification becomes unnecessarily complicated. TPCTL is one of the few logics that can express both hard and soft real-time deadlines, a feature useful in the verification of communication protocols in noisy media. Strong assumptions on the behaviour of the medium (e.g. the medium never loses more than three consecutive messages) can be replaced with weaker assumptions (e.g. successful transmission with some probability). The model checking algorithm is polynomial in the size of the formula and number of arithmetic expressions.

6.1.5 Expressiveness of Branching Temporal Logics

One of the main use of branching time logics in computer science is that the model-checking procedure is very efficient. The task is to represent a given system as a Kripke structure and check that it is a model of a given specification. CTL is quite adequate to express a certain set of useful properties. In contrast to the exponential complexities of model checking with a linear temporal logic (LTL), model checking with CTL is of linear complexity. Model checking with CTL* is much more complex than CTL, which is PSPACE-complete, as it needs a recursion involving checking of all paths from a particularstate [GRF00].

The branching logic systems can also be used to specify propertiesof concurrent programs. A frame of the logic represents an executiontree generated by a program.

The system properties which can be expressedby means of UB are as follows:

$\forall\Box p$: *safety property*: p is true at all states of each path.
$\forall\Diamond p$: *liveness property*: p is true at some state of each path.
$\exists\Diamond p$: *possibility property*: p is true at some state of some path.

Fairness constraints are not expressible in UB. All properties expressible in UB are also defined in CTL. The new properties, such as relative order of events, expressed by CTL contains the modality U. As in UB, fairness constraints cannot be expressible in CTL. CTL* can specify more properties over UB and CTL [Pen95]:

$\Box\Diamond p$: *impartiality property*
$\Box\Diamond p \rightarrow \Box\Diamond q$: *fairness property*
$\Diamond\Box p \rightarrow \Box\Diamond q$: *justice property*
$\exists\,((pUq) \vee \Box p)$: *weak until property*

These languages mentioned in this section can be made more expressive, while still keeping all their formulas as state formulas, by allowing classical connectives in between the temporal connectives and the path connective. If we add past operators does not increase expressiveness; it just allows more convenient notation to express useful properties. Due to complexity and expressiveness considerations some other logics have been defined, such as CTL^+ [EH82], $ECTL$ [EC82], $ECTL^+$ [EC82].

6.2 Partial-Order Temporal Logics

Partial order structures are similar to branching structures except that every time instant may also have several immediate predecessors corresponding to different pasts. The first logic based on partial orders was POTL, which is introduced in [PW84], and later its extended version, POTL$[U, S]$, defined in [KP86]. POTL and POTL$[U, S]$ can be viewed as extensions of UB and CTL by past modalities. However, their semantic structures can be linked with partial orderings representing runs of concurrent systems.

6.2.1 The Logic POTL

POTL is intended to describe partially ordered computations directly. Hence, it is possible to specify states with several successors and several predecessors. The language of POTL is an extension of the language UB by allowing quantification over backward paths. POTL does not have the finite model property due to the addition of backward operators Checking whether a POTL formula is satisfiable requires an exponential time algorithm.

POTL$[U, S]$ is the extension of POTL obtained by introducing *until* and *since*. There is a difference in the definition of POTL$[U, S]$ model that the initial or terminal states of some process may have no R-successors or predecessors, respectively.

POTL$[U, S]$ extends POTL, and since POTL does not have the finite model property, POTL$[U, S]$ does not have it either. There is a deterministic algorithm for deciding whether a POTL$[U, S]$ formula is satisfiable, of exponential complexity in the length of the tested formula. A sound and complete axiomatisation system for POTL$[U, S]$ is given in [Pen95].

6.2.2 Expressiveness of Partial Order Temporal Logics

POTL extends the expressiveness of UB by referring to the past. However, there is a difference between UB and POTL frameworks. The structure of the former represents an entire concurrent system. In the latter, a structure represents one possible run of a system composed of sequential processes. In this framework, POTL is used to specify properties involving all runs. For example, $q \rightarrow \forall \overleftarrow{\Diamond} p$ expresses that for every run, and for all backward fullpaths ending at states where q holds, there is a state at which p holds. All properties expressible in POTL can be expressible in POTL$[U, S]$. In addition, POTL$[U, S]$ allows specifying the properties concerning the relative order of events in the future and past. Model checking with POTL$[U, S]$ is more complicated than CTL. Indeed, the complexity is exponential in the size of the model and doubly exponential in the length of the tested formula [KP86]. The reason for this high complexity is that POTL$[U, S]$ formulas contain backward modalities, and they are interpreted over models corresponding to runs of concurrent systems.

7 Interval Temporal Logics

Interval temporal logics are temporal logics for representing both propositional and first-order logical reasoning about periods of time that is capable of handling both sequential and parallel composition. The interval-based scheme provides us with a richer representation formalism than the point-based approach.

In this section, we present a selection of well-known interval temporal logics. There are many other temporal logics in the literature, but most of them can be regarded as generalisations or specialisations of those discussed here.

7.1 Propositional Interval Temporal Logics

In this section we will present well-known propositional interval logics, which involve unary or binary modal operators, and whose semantic structures are over partial orderings with linear interval property, i.e. orderings in which every interval is linear (see [GMS04]).

The generic language of propositional interval temporal logics includes the set of propositional variables Φ, the propositional constants $\bot$ and $\top$, the Boolean connectives, and a set of modal operators specific to each logical system.

7.1.1 The Logic HS

The logic HS [HS91] is a relatively expressive propositional interval temporal logic. All modal operators of HS are unary. The logic HS has enough expressive power to distinguish the different situations of time's being discrete, continuous, bound, linear or complete. HS has enough expressive power to distinguish the different situations of time's being discrete, continuous, bound, linear or complete. Some of the situations are given below:

- *length0* $\equiv [B] \bot$

- *length1* $\equiv \langle B \rangle \top \wedge [B]$ *length0* (*length1* holds at intervals with no proper subintervals.)
- *dense* $\equiv \neg$*length1*
- *discrete* $\equiv$ *length0* $\vee$ *length1* $\vee$ ($\langle B \rangle$ *length1* $\wedge \langle E \rangle$ *lenght1*)

where $\langle B \rangle \phi$ is true iff ϕ holds at some interval that begins with the current interval and ends before it ends, $\langle E \rangle \phi$ is true iff ϕ holds at some interval that begins after the current interval starts and ends when it ends, and $[B] \phi$ is defined as $\neg \langle B \rangle \neg \phi$.

HS is a quite expressive logic due to its large modal operator set. However, it is not axiomatisable and is highly undecidable [HS91]. The following theorems are taken from [HS91]:

- The validity problem interpreted over any class of ordered structures with an infinitely ascending sequence is r.e.-hard (Thus, in particular, HS is undecidable for the class of all (non-strict) models, linear models, discrete linear models, dense linear models and unbounded linear models.)
- The validity problem interpreted over any class of Dedekind complete ordered structures having an infinitely ascending sequence is $\prod_1^1$-hard (For instance, the validity in any of the orderings of the natural numbers, integers, or reals is not recursively axiomatisable. Undecidability even occurs in the classes of structures with no infinitely ascending sequences.)
- The validity problem interpreted over any class of Dedekind complete ordered structures having unboundedly ascending sequences is co-r.e.-hard.

Undecidability results given above can be proved using an infinitely ascending sequence in the model to simulate the halting problem for Turing machines. Any unbounded ordering contains an infinite ascending sequence. A class of ordered structures contains an infinite ascending sequence if at least one structure in the class includes an infinite ascending sequence. In [MR99] undecidability was proved by means of tiling problem.

In [Ven90] some interesting results for the logic HS were presented. By using a geometrical representation for the modalities Venema introduced a sound and complete proof system for HS. He also proved that HS is strictly more expressive than any temporal logic based on linear orderings of time instants.

In [HS91] a translation machinery that converts a HS formula to its equivalent first-order formula on a corresponding first-order structure was provided. Such a translation is useful to reduce problems to well-known results in first-order logic.

7.1.2 The Logic CDT

The Logic CDT was introduced by Venema in [Ven91]. It is one of the most expressive propositional interval logic over linear orderings. CDT includes the binary modal operators C, D and T.

Since the logic HS is the propositional interval logic of Allen's relations [All83], every propositional interval logic with unary modalities based on Allen's relations is

subsumed by CDT. CDT can distinguish the different situations of time's being discrete, continuous, bound, linear or complete. For example, an interval's being discrete can be specified in CDT as follows:

- $(\mathit{length1}\ C\top) \wedge (\top C\ \mathit{length1})$.

In [Ven91] the author gives an axiomatic system which is sound and complete for the logic CDT which is interpreted over non-strict linear models. This axiomatic system can be extended for the classes of discrete linear orderings, dense linear orderings, etc. As a consequence of the previous results for the logic HS, the satisfiability problem for CDT is not decidable over almost all interesting classes of linear ordering, including $\mathbb{N}$,$\mathbb{Z}$,$\mathbb{R}$, etc.

A generalisation of CDT to (non-strict) partial orderings with the linear interval property, called BCDT$^+$, has been recently introduced in [GMS03a]. BCDT$^+$ features the same operators as CDT; however, it is interpreted over partially ordered domains with linear intervals. The decidability and axiomatizability of the strict versions of CDT and BCDT$^+$ have not been studied yet; but it is natural to expect that similar results apply there too.

7.1.3 The Logic PNL

Propositional Neighbourhood Logic (PNL) is the propositional fragment of First-Order Neighbourhood Logic introduced in [CH98]. It has been studied on both strict and non-strict linear structures [GMS03b]. The language with non-strict semantics is called PNL$^{\pi+}$, and it includes the modal operators $\diamondsuit_r$ (*met by*) and $\diamondsuit_l$ (*meets*), and the model constant π. The modal operators can have either strict or non-strict semantics.

Assume PNL$^+$ denotes the non-strict PNL without the modal constant π, and PNL$^-$ denotes the strict PNL without the modal constant π . It is easy to see that the logic PNL$^{\pi+}$ subsumes both PNL$^+$ and PNL$^-$.

Given that formulas are interpreted over strict linear models, PNL$^-$ has enough expressive power to distinguish the different classes of linear structures, such as discreteness, continuity, boundness, or completeness. For example, unboundness and density can be specified in PNL$^-$ as follows:

- $\mathit{unbound} \equiv \Box_r\phi \rightarrow \diamondsuit_r\phi$
- $\mathit{dense} \equiv (\diamondsuit_r\diamondsuit_r\phi \rightarrow \diamondsuit_r\diamondsuit_r\diamondsuit_r\phi) \wedge (\diamondsuit_r\Box_r\phi \rightarrow \diamondsuit_r\diamondsuit_r\Box_r\phi)$

In [GMS03b] several sound and complete axiomatic systems were provided for various classes of models. In addition to strict linear models [GMS03b] also provides sound and complete axiomatic systems for non-strict linear structures, complete unbounded linear structures, unbounded structures, dense structures, discrete structures, dense unbounded structures and discrete unbounded structures. As for decidability results, in [BMS07a] the authors show that the satisfiability problem for PNL$^{\pi+}$, PNL$^+$ and PNL$^-$ over the integers is NEXPTIME-complete. They develop a sound and complete tableau-based decision procedure, and prove that it is optimal. In [BGMS07], the expressive power of PNL$^{\pi+}$, PNL$^+$ and PNL$^-$ is compared, and it is shown that PNL$^{\pi+}$ is strictly more expressive than PNL$^+$ and PNL$^-$. Then, the authors prove that the

satisfiability problem for PNL$^{\pi+}$ over the class of all linear orders, as well as over some natural subclasses of it, such as the class of all well-orders[3] and the class of all finite linear orders, can be decided in NEXPTIME by reducing it to the satisfiability problem for the two-variable fragment of first-order logic over the same classes of structures [Ott01].

An important fragment of the PNL is the *Right Propositional Neighbourhood Logic* (RPNL) which is based on the right neighbourhood relation between intervals. As in the case of PNL, the language with non-strict semantics is called RPNL$^{\pi+}$. The non-strict fragment without the modal constant π is denoted by RPNL^{+}, an the strict fragment without the modal constant π is denoted by RPNL^{-}. As for decidability results, in [BM05] an EXSPACE tableau-based decision procedure is devised for RPNL^{-} interpreted over natural numbers. In [BMS07b] the authors develop an alternative NEXPTIME decision procedure that works for all variants of RPNL (RPNL$^{\pi+}$, RPNL^{+}, and RPNL^{-}) interpreted over natural numbers, and they prove its optimality.

7.1.4 The Logic PITL

PITL is the propositional fragment of the First-order Interval Temporal Logic (ITL) (see Section 7.2.1). PITL includes the chop operator C.

PITL was originally restricted to the class of discrete linear orderings with finite time. Intervals in such structures were identified with the finite sequences of points. The satisfiability problem for PITL interpreted over the class of non-strict discrete structures is undecidable [Mos83]. The satisfiability problem for PITL is reduced to checking the emptiness of the intersection of two context-free grammars. This problem is known to be undecidable. The satisfiability problem for PITL interpreted over non-strict linear structures and non-strict dense linear structures is also undecidable [Mos83].

Decidable variants of PITL, interpreted over discrete models, can be obtained by imposing the *locality projection principle*. Locality projection means that each propositional variable is true over an interval if and only if it is true at its first state. This allows one to collapse all the intervals starting at the same state into the single interval consisting of the first state only. Let LPITL be the logic obtained by imposing the locality projection principle to PITL. The syntax of LPITL coincides with that of PITL. However, in LPITL propositional variables are evaluated over points instead of intervals. The satisfiability for LPITL is non-elementary [Mos83].

[Mos83] introduced the logic QLPITL, which is an extension of LPITL over finite time with quantification over propositional variables. It was also shown that the satisfiability problem for the logic QLPITL interpreted over the class of non-strict discrete linear structures is (non-elementarily) decidable.

LPITL was also extended with the *chop-star* modality, denoted by * [Mos83, Mos00b, Mos00a, Mos03]. For any ϕ, ϕ^* holds over a given discrete interval if and only if the interval can be chopped into zero or more parts such that ϕ holds over each of them. The resulting logic is called LPITL*, and it is interpreted over finite or

[3] A *well-ordering* relation on a set S is a total order on S with the property that every non-empty subset of S has a least element in this ordering.

infinite discrete linear structures.

In [Mos83] the author gives a sound and complete axiomatic system for LPITL* which is interpreted over non-strict discrete linear structures.

As a matter of fact, the chop-star operator is a special case of a more general operator, called the *projection* operator. LPITL$_{proj}$, which was originally proposed by Moszkowski in [Mos83], is the extension of LPITL with the projection operator $proj$. LPITL$_{proj}$ is interpreted over finite state sequences. It extends the operators $\bigcirc$ (*strong next*) and C (*chop*) of LPITL with a new binary modal operator $proj$ (*projection*) which denotes the repetitive behaviour. $\bigcirc\varphi$ holds if and only if φ holds over an interval of length one less than the current interval, resulting from moving one state into the future. An interval satisfies $\varphi C \varphi'$ if it can be divided into two contiguous sub-intervals such that φ holds over the first sub-interval, and φ' holds over the second. An interval satisfies $\varphi \; proj \; \varphi'$ if it can be sub-divided into a series of sub-intervals, each of which satisfies φ, called the *projection formula*, and the new interval formed from the end points of these sub-intervals satisfies φ', called the *projected formula*. The satisfiability problem for LPITL$_{proj}$ is known to be non-elementary [GMS04].

7.2 First-Order Interval Temporal Logics

First-order interval temporal logics have been introduced as a tool for the formal specification and verification of hardware real time systems. ITL is the most commonly known first-order interval temporal logic. Numerous extensions of ITL, such as Duration Calculus [CHR91],Neighbourhood Logic [CHR91], etc., have been shown to be useful in the specification of various kinds of software and hardware systems.

7.2.1 The Logic ITL

ITL, interpreted over discrete linear orderings with finite time intervals, was first introduced in [Mos83]. The formulas of ITL are constructed from the following: an infinite set of global (independent of time and time intervals) variables $x, y, z, ...$, an infinite set of temporal variables $t, t', ...$, an infinite set of global function symbols $f^n, g^m, ...$, where f^n is a function of arity n and g^m is a function of arity m, an infinite set of predicate symbols $P^n, R^m, ...$, where P^n is a predicate of arity n and R^m is a predicate of arity m, an infinite set of temporal propositional letters $X, Y, ...$.

A sound and complete axiomatic system is represented in [Dut95]. A term or formula is called *flexible* if a temporal variable including the symbol ℓ or a propositional letter occurs in the term or formula. A term or formula which is not flexible is called *rigid*.

Sound and complete axiomatic systems for local variants of ITL (with locality constraint) for finite and infinite time were presented in [Dut95, Mos00a]. ITL was extended with projection in [Gue00a] where a complete axiomatic system is provided. A probabilistic extension of ITL was studied in [Gue98]. Not surprisingly ITL is highly undecidable.

7.2.2 The Logic NL

ITL does not allow looking outside of the current interval. The logic NL was proposed in [CHR91], where the left *neighbourhood modality* $\Diamond_l$ and right *neighbourhood modality* $\Diamond_r$ were introduced.

NL can express any of the modalities corresponding to the Allen's relations; thus, it can represent the properties of the underlying linear ordering, such as discreteness, density, etc. For example, the chop operator C can be expressed in terms of the modalities $\Diamond_l$ and $\Diamond_r$ as follows:

- $\phi C \psi = \exists x, y \, (\ell = x + y) \wedge \Diamond_l \Diamond_r \left((\ell = x) \wedge \phi \wedge \Diamond_r \left((\ell = y) \wedge \psi \right) \right)$

In [BRC00] a sound and complete axiomatic system is given for the logic NL. In [BC97] *up* and *down modalities,* represented by $\Diamond_u$, $\Diamond_d$ respectively, were introduced, and two dimensional version of NL, called NL^2, was proposed. NL is an undecidable logic as is ITL.

7.2.3 The Logic DC

Duration Calculus (DC) [CHR91] is a first-order interval temporal logic with the additional notion of *state*, which is characterised by a *duration*. The duration of a state is the length of the time period during which the system remains in the state. DC is an extension of ITL in the sense that temporal variables other than ℓ have a structure $\int S$, where $\int S$ is called a *state duration* and S is called a *state expression*. The special interval variable ℓ denotes the interval *length*.

DC has been successfully applied to the specification and verification of real-time systems. As a specification example, we specify the real-timerequirement of a gas burner system, which is "the proportion of leak time in an interval is not more than one-twentieth of the interval, if the interval is at least one minute long". This requirement can be expressed in DC as follows:

- $\mathrm{Req} \equiv \ell \geq 60 \Rightarrow 20 \int Leak \leq \ell$

All axioms and inference rules of ITL can be adopted in DC. However, additional axioms are needed for temporal variables. In [CH04] an axiomatic system for Duration Calculus is given. The satisfiability problem for both first-order and propositional DC has been shown to be undecidable [CHS93].

Various fragments of DC have been investigated so far. In [CHS93] a fragment of propositional DC, called RDC, was introduced. It was shown that RDC has a decidable satisfiability problem when interpreted over $\mathbb{N}$, $\mathbb{Q}$ and $\mathbb{R}$. In [Rab98] the satisfiability problems of several extensions of RDC were studied . In [Fra96] a decidable class of RDC was extended for continuous time with a restriction on the finite variability such that the number of discontinuous points of any state in any unit interval has a fixed upper bound. In [GD99] a decidability result was presented for a variant of DC where negation is removed from RDC; but an iteration operator is introduced together with some form of inequalities. In [CSC00] another fragment of propositional DC was introduced by imposing some syntactic restrictions. It was shown that this logic is expressive enough to capture Allen's relations [All83]. By proposing a sound,

complete and terminating tableau system for the logic it was shown that the satisfiability problem is decidable. The tableau system is a mixed procedure, combining standard tableau techniques with temporal constraint network resolution algorithms. In [Pan01] quantification over states was introduced. The satisfiability of formulas is still decidable. This decision algorithm was implemented as a tool called DCVALID. In [CH98] Duration Calculus and first-order neighbourhood logic were combined, and a axiomatic systems for DC and NL were merged. The fragment of DC/NL obtained by restricting the formulas to be constructed only from state expressions was proved to be decidable, while the extension with the formulas with equality is undecidable.

So far, there has been no tool available for model checking DC. In general, there is no universal model checking techniques for DC formulas. In order to achieve relatively efficient model checking techniques, we have to restrict ourselves to a class of system models and DC requirements. In [Fra96] some model checking tools were developed for a class of models which are restricted to the possible behaviours of embedded real-time systems. In [Zho94, DH96, TH98, TH04] some techniques were developed to check if a timed automaton satisfies a duration calculus formula written in the form of *linear duration invariants*. In [Pan01] a DC validity checker, called DCVALID, to check the validity of discrete time DC formulas. [Frä02] suggested *bounded validity checking* [BCCZ99] of a discrete-time DC without timing constraints by polynomial-sized reduction to propositional SAT solving. In [SHP98] some algorithms were developed to check if a DC formula is satisfied by all integer models.

7.2.4 The Logic IDL

Duration Calculus is designed to be a highly expressive logic for specifying complex requirements over real-time systems; but the automata theory for DC models is rather primitive and there are no tools supporting such models. By contrast, the notion of timed state sequences has found widespread use in modelling behaviour of real-time systems [AD96]. Their automata theory is also well investigated, and there are now powerful tools such as Hytec [AHH96], Uppaal [BLL+96], Kronos [BDM+98] etc. for the analysis of these automata.

In [Pan02] a variant of the Duration Calculus, called *Interval Duration Logic (IDL)* is introduced. IDL has finite *timed state sequences* as its models. It is a dense time interval logic, incorporating the notion of cumulative amount of time (duration) for which a condition holds in a given time interval. Because of this, IDL is well-suited for describing complex properties of real-time systems, including scheduling and planning constraints. As an example, we give a specification example from a gas burner system. The property "between two occurrences of Leak there will be at least k seconds" is specified in IDL as follows:

- $\Box((\lceil\lceil Leak\rceil\frown\lceil\lceil\neg Leak\rceil\frown\lceil\lceil Leak\rceil^0) \Rightarrow \ell \geq k)$

Due to its high expressive power, the satisfiability of IDL turns out to be undecidable. However, several approaches can be used for checking validity (and model checking) of IDL formulae in practice. [SPS05] proposes *bounded validity checking* [BCCZ99] of IDL formulae by polynomially reducing this to checking unsatisfiabil-

ity of *lin-sat* formulae. This technique is implemented and performance results are obtained by checking the unsatisfiability of the resulting lin-sat formulae using the ICS solver [FORS02], a SAT-based solver to check the lin-sat formula for satisfiability. [SPS05] also performs experimental comparisons of several approaches for checking validity of IDL formulae, including (a) digitization technique [CP03], combined with an automata-theoretic analysis[Pan01], (b) digitization technique [CP03] followed by pure propositional SAT solving [Frä02], and (c) lin-sat solving [FORS02]. The paper provides experimental results on the relative performance of these techniques on several problems drawn from the Duration Calculus literature.

[Pan02] presents a subset *LIDL* of IDL consisting only of *located time constraints* which is decidable. The paper shows that the models of an LIDL formula can be captured as timed words accepted by a finite state event-recording integrator automaton. This gives an automata theoretic decision procedure for the satisfiability of LIDL. It is also shown that LIDL has the same expressive power of an event-recording automata. A large number of examples from Duration Calculus literature can be expressed within LIDL. Thus, it is amongst the important significant subsets of DC which are known to be decidable and also practically interesting.

8 Real-Time Temporal Logics

Specification of real-time systems must be supported by formal, mathematically-founded methods in order to be satisfactory and reliable. Temporal logics have been used for this purpose for several years. Temporal logics allow the specification of system behaviour in terms of logical formulas, including temporal constraints, events, and the relationships between the two. In the last decade, temporal logics have reached a high degree of expressiveness, although not all are suitable for specifying real-time systems.

Generally speaking real-time temporal logics have been defined to satisfy specific needs. In recent years the structure and capabilitiesof temporal logics have grown. In some cases, simple temporal logicsare preferable to more complex and powerful ones, since the former can be more satisfactorily adopted in certain applications.

Below we review a number of well-known temporal logics designed for the specification of real-time systems. All these logics are different in terms of expressiveness, order, presence of a metric for time, the type of temporal operators, the fundamental time entity and the structure of time. They also have different capabilities for the specification, validation, and verification of real-time systems.

8.1 Real-Time Temporal Logic (RTTL):

RTTL [OW87, Ost89] is a first-order explicit clock logic. The temporal structure is linear and the fundamental entity is the point. Time is limited in the past and unlimited in the future. Time is defined with both a sequence of states and a sequence of temporal instants. RTTL is an explicit clock logic because an RTTL formula may explicitly use the clock time variable t with any of the arithmetic operators (e.g. $+, -, =, \geq, <$) using arbitrary first-order quantification over rigid time variables. An example of an RTTL formula is the *bounded response time* given by

- $\Box T[(red \wedge t = T) \rightarrow \Diamond(green \wedge T + 3 \leq t \leq T + 5)]$

In the above formula, the clock variable t is a flexible variable (changes from state to state in the trajectory). The quantified variable T is a rigid variable (retains the same value over all states in the trajectory). The above formula asserts that if the traffic light is *red* at time T, then eventually within 3 to 5 ticks from T the light must turn *green*. The bounded response property may be abbreviated by the formula $red \rightarrow \Diamond_{[3,5]} green$.

The possibility of adopting an explicit reference to the system clock value, and indirect quantifications on values assumed by the clock leads to the ability to write every type of ordering and quantitative constraints. This is extremely interesting for the specification of real-time systems. However, this flexibility leads to undecidability and the production of formulas that are quite difficult to understand and manipulate with respect to other temporal logics that avoid quantification over time-dependent variables.

The specification in Section 8.3 is expressed in RTTL as follows:

- $\Box(B \rightarrow \bigcirc((\Diamond_{\leq t_b} endA) \wedge \neg(\neg startA\ U\ endA)))$

where $\bigcirc$ is the next operator, $A\ U\ B$ means "A until B", and $\Diamond_{\leq t} A$ means that there is an occurrence of A which lasts t time units.

RTTL is a very expressive logic. It can be interpreted over discrete and dense time domains. The satisfiability problem is undecidable for any of the cases [AH90]. The model checking problem is also undecidable. A sound proof system is provided for RTTL.

There are some interesting syntactic fragments of RTTL. These fragments are restricted to finite state, propositional temporal properties:

XCTL [HLP90] is a discrete time propositional explicit clock logic. The atomic timing constraints allow the primitives of comparison and addition. XCTL restricts the quantification level. It allows only one outermost level of quantification over rigid time variables. The quantification is therefore never explicitly displayed. On the other hand, it allows general arithmetic timing expressions, including addition and subtraction of variables and constants. The satisfiability and model checking problems are both undecidable for XCTL over dense time [HLP90]. However, these problems are PSPACE-complete for the quantifier-free explicit-clock logic XCTL, despite the admission of addition over time [HLP90]. [HLP90] provides a single exponent decision procedure for the validity of XCTL formulas. XCTL, however, is a language that is not closed under negation and, hence, cannot be used to solve the *homogeneous* verification problem (i.e. both the model and the specification are expressed in the same formal language). However, a special model-checking algorithm for XCTL is given that is doubly exponential in the size of the formula and singly exponential in the size of the model [HLP90].

TPTL [AH90] is a discrete time propositional logic whose timing constraints allow comparison and addition (but only of integer constants, i.e. no variables). TPTL uses auxiliary static timing variables to record the value of the clock at different states, but replaces the explicit references to the clock itself by a special type of *freezing* quan-

tification[4]. The satisfiability and model checking problems are EXPSPACE-complete for TPTL over discrete time domain, but undecidable TPTL over dense time domain [AH89]. [AH89] gives a doubly-exponential-time decision procedure for TPTL. The model checking algorithm for the logic depends exponentially on the value of the product of all time constants. [AH90] shows that the addition of past operators renders the satisfiability problem for TPTL non-elementary. [Hen91] proves that there is a complete finite axiomatization for TPTL in the case of discrete time. These results show that the freeze quantification of TPTL to be superior to the classical quantification of RTTL.

8.2 Metric Temporal Logic (MTL):

MTL [Koy90] is a propositional bounded-operator logic; its temporal operators include time-bounded versions of the 'until', 'next', 'since', and 'previous' (the past dual of next) operators. It is a fragment of RTTL in which references to time are restricted to bounds on the temporal operators. For example, the formula $A \rightarrow \Diamond_{\leq 5} B$ means that if A occurs then eventually within 5 time units B must occur. The presence of the metric for time allows one to reduce the need for quantifications on a temporal domain (i.e. no references to an explicit clock are allowed), and hence MTL is called a *hidden clock* or *bounded temporal operator logic*.

Interpretations for MTL are metric point structures based on a linearly ordered time domain. A distance function provides a metric for time. Various time constraints can be imposed on the distance function, depending on the notion of time that is used (e.g. transitivity, irreflexibility, and the existence of absolute differences).

In [Koy90], a real-valued time domain is used. This allows MTL to express certain properties of continuous time variables (e.g temperature and pressure) more succinctly than discrete time logics. MTL does not allow references to an absolute point in time, nor does it allow the specifier to relate adjacent temporal contexts. As an example formalism, the specification in Section 8.3 is specified in MTL as follows:

- $B \rightarrow \mathbf{F}_{t_b} endA \wedge \neg(\neg startA \ until \ endA)$

[AH90] states that the satisfiability and model checking problems for MTL over dense time domain are undecidable, but a deductive system is available. [AH90] also shows that these problems are EXPSPACE-complete for MTL over discrete time domain. [AH90] gives a doubly-exponential-time decision procedure for MTL. The verification algorithm for MTL [AH90] depends exponentially on the value of the largest time constant involved. [Koy90] provides a sound proof system for MTL. In [HLN+90] it is shown that XCTL and MTL are incomparable. For each of these logics, there is a property expressible in one which is not expressible in the other. Each of these properties is a reasonable real-time requirement. In the discrete time case, TPTL and MTL are equally expressive (it is conjectured that this equality does not extend to dense domains [Hen91]). [OW05] finds that the satisfiability problem for MTL over finite timed words is decidable, with non-primitive recursive complexity.

[4]The freeze quantifier "$x.$" binds the associated variable x to the time of the current temporal context: the formula $x.\phi(x)$ holds at time t iff $\phi(x)$ does. Thus, in the formula $\Diamond y.\phi$ the time variable y is bound to the time of the state at which ϕ is "eventually" true.

The real-time logic MITL was introduced in [AFH91] as a restriction of MTL. MITL is a propositional linear-time logic with an interval-based strictly-monotonic real-time semantics; that is, it is interpreted over timed observation sequences. MITL employs the nonnegative reals as time domain. MITL uses the bounded-operator syntax with the restriction that the temporal operators must not be bounded (subscripted) by singular intervals. For example, the formula $\Box(p \rightarrow \Diamond_{[3,3]} q)$ is disallowed; that is, MITL rules out a form of equality constraints.

The proof of the undecidability of real-time logics over a dense time domain makes crucial use of *punctuality* properties[5]. The bounded-operator logic MITL originated in an effort to define a nontrivial real-time logic that cannot express punctuality requirements and, indeed, the satisfiability and model checking problems for MITL were shown to be EXPSPACE-complete. The doubly-exponential-time verification algorithm for MITL, which is the first such algorithm for a linear-time logic over a dense time domain, is considerably more complex than discrete-time algorithms.

The restriction of MITL time modalities to positive-length intervals was intended to guarantee decidability; but recent results [OW05, LW08] show that this restriction is not necessary for deciding MTL over finitary event-based semantics. The original version of MITL [AFH91] contains only future temporal operators. [MNP05] compares the past and future fragments of the real-time temporal logic MITL with respect to the recognizability of their models by deterministic timed automata. It proves that timed languages specified by the past fragment of MITL, can be accepted by deterministic timed automata. On the other hand, certain languages expressed in the future fragment of MITL are not deterministic.

8.3 Real-Time Logic (RTL):

RTL is a formal language for reasoning about events and their times of occurrence. It is introduced in [JM86]. RTL is an extension of first-order logic with a so-called *occurrence* function which assigns a time value to each occurrence of an event. RTL formulas may be used to express real-time requirements in a formal way or to set up additional safety constraints that the system must comply with. Mechanical methods have been devised to translate real-time requirements into RTL formulas and to validate a class of safety assertions against a specification [JM86].

RTL presents an absolute clock to measure time progression. The value of this clock can be referenced in the formulas. The temporal domain is the set of natural numbers, and is linear, discrete, limited in the past, unlimited in the future, and totally ordered. The fundamental entity is the time instant. It was shown in [AH90] that RTL is undecidable even when the syntax is restricted.

In RTL, there are no problems in specifying ordering and quantitative temporal constraints, since it is possible to make explicit reference to time even through quantification. The main problem with RTL is the fact that absolute system time is referenced, with a low level of abstraction, leading to very complex formulas required

[5] A *punctuality* property states that the event B follows A in exactly 3 seconds. For any real-time logic that is closed under boolean operations, and that can express punctuality, the satisfiability problem is undecidable for a dense time domain.

to describe the system. Let us consider the following temporal constraint: for each occurrence of an event B which happens at a time instant t_0, the predicates *startA* and *endA* hold (marking an interval [*startA*, *endA*] at which A is true), and the interval [*startA*, *endA*] is subsumed by the interval $[t_0, t_0 + t_b]$, where $t_0 \leq startA \leq endA \leq t_0 + t_b$. This constraint is specified in RTL as follows:

- $\forall t. \forall i. @(\Omega B, i) = t \rightarrow (\exists j. (t \leq @(\uparrow A, j)) \wedge (@(\downarrow A, j) \leq t + t_b))$

where ΩB denotes the occurrence of the event B, t denotes time, $\uparrow A$ denotes the beginning of the action A, $\downarrow A$ denotes the completion of the action A, and i and j are the occurrences of the events marked with the operator @. Time is captured by the occurrence function @ which assigns time values to event occurrences. $@(\Omega B, i)$ is defined as the time of the i-th occurrence of ΩB.

RTL's event occurrence function allows for a rich expression of periodic and non-periodic real-time properties. However, RTL is undecidable [AH90]. It does not treat data structures or infinite state systems. RTL formulas impose a partial order on computational actions which is useful for representing high level timing requirements.

The decision and verification algorithms for RTL are not practical in general. To improve the decision procedures RTL formulas are better structured, using domain knowledge, into a computation graph. With this improved structuring an exponential time decision procedure (in the worst case) is obtained. In [JS88], a visual formalism called Modecharts is introduced. Modecharts have some similarities to Statecharts. Modecharts specify a decidable fragment of RTL, in a "natural", state-based, visual fashion preferred by design engineers. A method is provided for translating Modecharts into computational graphs, from which the verification can be performed.

8.4 Real-Time Interval Logic (RTIL):

RTIL is introduced in [RG89]. RTIL includes a metric for time. Intervals are constructed by assigning numerical values to interval bounds. It is also possible to measure the interval duration. This characteristic makes RTIL interesting for the specification of real-time systems. Instants can be specified absolutely or relative to the beginning of the current context. RTIL also permits quantification over finite domains. This feature does not enhance the expressiveness of the logic, but simplifies the writing of complex and repetitive formulas.

The specification in Section 8.3 is specified in RTIL as follows:

- $\Box [\odot B \hookrightarrow t_b]^* (\odot startA \Rightarrow \odot endA)$

where $\odot A$ extracts the time instant in which A becomes true, and the operator$*$ means there exists a subinterval.

8.5 Tempo Reale ImplicitO (TRIO):

TRIO [GMM90] is a formal language and a method for the specification, analysis and verification of critical, real-time systems. The TRIO language is based on a metric extension of first-order temporal logic and exploits typical object-oriented features to support the managing of large, complex, and maintainable specifications.

The choice of first order temporal logic was motivated by reasons of naturalness, simplicity and compactness of specification. The temporal structure is linear and totally ordered: possible temporal domains are the natural numbers, the integers, the real numbers, or an interval of one of these set. The fundamental temporal entity is the point and a metric for time is available. On that basis, it is possible to measure the distance of two points and the length of an interval.

TRIO presents only two temporal operators: *Future(A,t)* and *Past(A,t)* for specifying that A occurs at time instant t in the future and past, respectively. From these two basic operators many other derived temporal properties can be defined such as "always in the future" and "sometime in the past".

The temporal operators introduced by TRIO, with the possibility of quantification on temporal variables without any restriction, permit the expression of order and quantitative temporal constraints as needed for real-time systems specification. It is necessary to use quantification over the time domain, so formulas are often complex and difficult to read and manipulate.

The specification example in Section 8.3 can be written in TRIO as follows:

- $Alw(B \rightarrow \exists t((0 < t < t_b) \wedge \mathbf{Futr}(endA, t) \wedge \exists t'(0 < t' < t \wedge \mathbf{Futr}(startA, t')))$

where $\mathbf{Futr}(A, t)$ denotes that A occurs at a time instant t in the future, and Alw is a user-defined operator which stands for $\forall t(t > 0 \rightarrow \mathbf{Futr}(A, t)) \wedge A \wedge \forall t(t > 0 \rightarrow \mathbf{Past}(A, t))$.

TRIO has mainly been used for the validation and verification of system requirements through testing activity (history-checking), and not by means of the proof of system properties. TRIO has been described as an executable logic language in the general sense. It can be used to build a model of the system under specification as TRIO formulas. Histories of system variables can be checked against the specification in order to verify whether they satisfy the specification. So TRIO must be considered a specific case of model-checking.

Since TRIO is an extension of FOL, which is undecidable, TRIO is also an undecidable logic.

8.6 Temporal Interval Logic with Compositional Operators (TILCO):

TILCO [Mat96, MN96] extends FOL with a set of temporal operators to create a logic language that can specify both relationships between events and time and data domain transformations. TILCO uses the interval as a fundamental temporal entity. The temporal structure is linear and presents a metric for time that associates an integer number to every temporal instant; no explicit temporal quantification is allowed.

TILCO can be used to specify temporal constraints among events in either a qualitative or quantitative manner. Therefore, interval boundaries, which specify the length of intervals and actions, can be expressed relative to other events (qualitatively) or with an absolute measure (quantitatively). This allows for the definition of expressions of ordering relationships among events or delays and time-outs. These features are mandatory for specifying the behaviour of real-time systems. The TILCO deductive approach is sound and, thus, consistent.

In TILCO, the same formalism used for system specification is employed for describing high-level properties that should be satisfied by the system itself. These must be proved on the basis of the specification in the system-validation phase. Since TILCO operators quantify over intervals, instead of using time points, TILCO is more concise in expressing temporal constraints with time bounds, as needed in specifying real-time systems. In fact, TILCO can be effectively used to express invariants, precedence among events, periodicity, liveness and safety conditions, etc. These properties can be formally verified by automatic theorem-proving techniques.

The specification in Section 8.3 can be expressed in TILCO as follows:

- $B \rightarrow endA?(0, t_b) \wedge \neg \mathbf{until}(endA, \neg startA)$

where ? denotes universal temporal quantification.

A sound deductive system for TILCO is presented in [Mat96, MN00]. This system is used in the context of the general theorem-prover Isabelle [Pau94] to provide support for proving TILCO formulas. Since TILCO extends FOL, it is undecidable in the general case. However, the subset of formulas that presents only quantifications on finite sets is decidable.

9 Probabilistic Logics

Probabilistic reasoning has been recognised as a useful tool in many fields of computer science and artificial intelligence. There is an extensive investigations about formal systems for reasoning in the presence of uncertainty. In probabilistic temporal reasoning, it is usual to start with classical logic and to add probabilistic operators that behave like modal operators. It is also possible to combine the probabilistic approach with some other nonclassical logics. For example, the probabilistic operators are added to modal logic of knowledge in [FH94], while a probabilistic extension of Intuitionistic Logic is analysed in [MOR03].

9.1 Probabilistic Temporal Logics

9.1.1 The Logics PCTL and PCTL*

Probabilistic Computation Tree Logic, *PCTL*, is a probabilistic branching time temporal logic which allows for probabilistic quantification of described properties. It was defined by Hansson and Jonsson in [HJ89, HJ94]. PCTL is a useful logic for stating *soft deadline properties*, e.g. "after a request for a service, there is at least a 98% probability that the service will be carried out within 2 seconds." Soft deadline are interesting in systems in which a bound on the response time is important; but the failure to meet the response time does not result in a disaster.

This logic extends the temporal logic CTL by Emerson, Clarke and Sistla [CES86]. In PCTL time is discrete and one time unit corresponds to one transaction along an execution path. To enable reasoning about soft deadlines path quantifiers have been replaced with probabilities. Formulas in this logic are interpreted over discrete-time Markov chains. The set of PCTL formulas is divided into *path formulas* and *state*

formulas. Intuitively, state formulas represent properties of states and path formulas represent properties of paths (i.e. sequences of states).

The main difference between PCTL and branching time temporal logics, such as CTL, is the quantification over paths and the ability to specify quantitative time. CTL allows universal and existential quantification over paths, i.e. One can state that a property should hold for all computations (paths) or that it should hold for some computations (paths). It is not possible to state that a property should hold for a certain portion of the computations, e.g. for at least 50% of the computations. In PCTL, on the other hand arbitrary probabilities can be assigned to path formulas, thus obtaining a more general quantification over paths.

In PCTL it is possible to state that a property will hold continuously during a specific time interval, or that a property will hold sometime during a time interval. Some real-time requirements are specified in PCTL as follows:

- *(i)* $AGf \equiv f\mathcal{U}_{\geq 1}^{\leq \infty} false$ *(ii)* $EFf \equiv true U_{>0}^{\leq \infty} f$.

where $f_1 \mathcal{U}_{\geq t}^{\leq p} f_2$ means that "there is at least a probability p that either f_1 will remain true for at least t time units, or that both f_2 will become true within t time units and that f_1 will be true from now on until f_2 becomes true"; and $f_1 U_{\geq t}^{\leq p} f_2$ means that "there is at least a probability p that both f_2 will become true within t time units and that f_1 will be true from now on until f_2 becomes true." Therefore, AGf intuitively means that "f is always true (in all states that can be reached with non-zero probability), and EFf means that there exists a state where f holds which can be reached with non-zero probability."

In [HJ94] several model checking algorithms for the logic PCTL, with different suitability for different classes of formulas, are presented. The algorithms require a polynomial number of arithmetic operations, in size both of the formula and the *Markov chain*[6].

[ASB95] defines another probabilistic variant of CTL [CES86]. This new logic is called *PCTL**. The logic expresses quantitative stochastic properties of systems, which are themselves modelled as discrete *Markov processes*[7]; furthermore, it exhibits an elementary model checking procedure. Discrete Markov processes exhibit a natural notion of bisimulation; this is shown to be sound and complete with respect to PCTL*. [ASB95] also extends the universe of models to include *generalized discrete Markov processes*[8] which can be used for modelling systems where the transition probabilities are not completely specified; these systems allow notions of abstraction and refinement. Using characterisations of discrete Markov processes and results on the decidability of real closed fields [BOKR84], [ASB95] derives an elementary decision procedure for model checking PCTL* over generalized discrete Markov processes. The

[6] A *Markov chain* is a pair (S, P) where S is a (potentilly infinite) set of *states* and $P : S \times S \to [0,1]$ is the *transition probability matrix* satisfying the condition $(\forall s \in S) \sum_{s' \in S} P(s, s') = 1$

[7] A (finite) *Markov process* is a 4-tuple $(AP, S, P, \mathcal{L})$, where AP is a finite set of *atomic propositions*, S is a countable set of *states*, $P : S \times S \to [0,1]$ is the *transition probability matrix* satisfying the condition $(\forall s \in S) \sum_{s' \in S} P(s, s') = 1$ and $\mathcal{L} : S \to 2^{AP}$ is *labeling function*.

[8] A *generalized Markov process* is a 3-tuple $(AP, S, \mathcal{L})$, where AP, S and $\mathcal{L}$ are defined as in Markov processes, and a finite set of constraints on the transition probabilities.

model-checking algorithm presented in [ASB95] can be used to determine the validity of PCTL* formulas. In fact, [ASB95] shows that the decision problem for PCTL* formulas on generalized Markov processes is decidable. However, no efficient computational method is given for this problem. In addition, no sound and complete axiomatisation of the logic is given.

[BA95] presents model-checking algorithms for extensions of PCTL and PCTL* to systems in which the probabilistic behaviour coexists with nondeterminism (probabilistic-nondeterministic systems), and shows that these algorithms have a polynomial-time complexity in the size of the system. This provides a practical tool for reasoning on the reliability and performance of parallel systems. It is already known from [HJ89, HJ94, ASB95] that PCTL and PCTL* model checking on Markov chains can be done in polynomial time in the size of the system. This therefore means that adding nondeterminism still preserves the polynomial time bound, provided the size of the system takes into account not only the number of states, but also the encoding of the transition probabilities. The situation is different for the time bounds expressed in terms of the size of the formula. Model checking of PCTL formulas can be done in linear time on the size of the formula for both Markov chains [HJ89, HJ94] and probabilistic-nondeterministic systems. However, while PCTL* model checking on Markov chains can be done in single exponential time in the size of the formula [ASB95], PCTL* model checking on probabilistic-nondeterministic systems requires at least doubly exponential time in the size of the formula.

9.1.2 The Logic PLTL

[Ogn06] describes a way in which probabilistic reasoning can be enriched with some temporal features. It introduces a Probabilistic Propositional Temporal Logic (*PLTL*), which is a propositional probability discrete-linear temporal logic. The temporal part of the logic is a standard discrete linear-time logic, where the time flow is isomorphic to natural numbers, i.e. each moment of time has a unique possible future, while the corresponding language contains the 'next' operator and the reflexive strong 'until' operator, while the operators 'sometime' and 'always' are definable from the 'next' and 'until' operators. In this logic the probabilistic operators quantify events along a single time line. It allows one to express sentences such as "(according to the current set of information) the probability that, sometime in the future, α is true is at least n." As the knowledge can evolve during time, the probability of α might change too.

Given that $\bigcirc$, F and G are the 'next', 'sometime' and 'always' operators, respectively, and $P_{\tau\alpha}$ ($\tau \in \{<, \leq, =, \geq, >\}$) is a unary probabilistic operator, an example of a PLTL formula is

- $\bigcirc P_{\geq r} p \wedge F P_{<s}(p \rightarrow q) \rightarrow G P_{=t} q$

which can be read as "if the probability of p in the next moment is at least r and sometime in the future q follows from p with the probability less than s, then the probability of q will always be equal to t."

[Ogn06] analyses completeness, decidability and complexity of the logic PLTL. It describes a class of so called measurable models. It is proved that PLTL restricted to

the class of all measurable models ($PLTL_{Meas}$) has a sound and complete (infinitary) axiomatisation. The term infinitary means the language and formulas are finite, while only proofs are allowed to be infinite (The completeness cannot be proved with finitary axiomatisation). A $PLTL_{Meas}$-satisfiable formula is satisfiable in an ultimately periodic model in which various parameters are bounded by functions depending on he size of the formula. [Ogn06] also shows that the satisfiability problem for $PLTL_{Meas}$ is PSPACE-hard, and that it belongs to NEXPTIME.

In [Ogn06] also introduces First-order Probabilistic Temporal Logic (*FOPLTL*), which is the first-order version of PLTL. The complete infinitary axiomatisation is extended for the logic FOPLTL (No complete finitary axiomatisation is possible). The set of all FOPLTL-valid sentences is not recursively enumerable [GHR94].

9.1.3 The Logics PTL_f and PTL_b

[HS84] presents two (closely-related) propositional probabilistic temporal logics, called PTL_f and PTL_b, based on temporal logics of branching time as introduced in [BAMP81] and [CE82]. These logics are interpreted over models which can simulate the execution of probabilistic programs; for PTL_f these are essentially finite Markov chains, whereas for PTL_b they are infinite stochastic processes whose state-transition probabilities are bounded away from 0 (this assumption holds for finite-state concurrent probabilistic programs since there are only finitely many different state-transitions).

PTL_f and PTL_b are expressive enough to allow one to express various properties of programs, such as invariant and liveness properties, without explicit reference to the values of the transition probabilities. PTL_f is intended for reasoning about sequential programs whereas PTL_b extends PTL_f and is intended for reasoning about concurrent programs. To show the syntax of the logics, let us consider the formula $p\forall Uq$. This formula intuitively means that along all paths w starting with the initial state and consisting only of transitions with nonzero probability, p holds at all states of w up to the first state, if any, at which q holds.

It turns out that satisfiability of formulae in both logics is decidable, in one-exponential time, by decision procedures based on the tableau technique which-generalise similar procedures for the nonprobabilistic logics of [BAMP81] and [CE82]. Together with these decision procedures, [HS84] also provides complete axiomatisations for both logics, and shows that the same decision procedures can be used to construct a proof of the negation of any unsatisfiable formula. Several meta-results, including the absence of a finite-model property for PTL_b, and the connection between satisfiable formulas of PTL_b and finite state concurrent probabilistic programs, are also discussed.

Some of the related works in the literature are given as follows: Like the logic PTL_b, the logics proposed by Pnueli [Pnu83] and by Lehmann and Shelach [LS83] also aim to reason about concurrent probabilistic programs and do not refer explicitly to the values of probabilities involved. However, the logic of Pnueli is not complete; the logic of Lehmann and Shelach is more expressive than PTL_b, and consequently the presently available decision procedures for that logic are much more inefficient than PTL_b. Both these logics are based on temporal logic of linear time.

We would like to finally mention that [CY88] determines the complexity of testing

whether a finite state (sequential or concurrent) probabilistic program satisfies its specification expressed in linear temporal logic (PTL). For sequential programs [CY88] gives an exponential time algorithm, and shows that the problem is in PSPACE; this improves the previous upper bound by two exponentials and matches the known lower bound. For concurrent programs it is shown that the problem is complete in double exponential time, improving the previous upper and lower bounds by one exponential each. [CY88] also addresses these questions for specifications described by ω-automata or formulas in extended temporal logic.

9.1.4 The Logic PDC

The Probabilistic Duration Calculus (PDC) was introduced in [LRSZ92] as an extension of Duration Calculus [CHR91]. PDC presents a calculus that enables the design of an embedded real-time system to reason about and calculate whether a given requirement will hold with a sufficiently high probability for a given failure probabilities of components used in the system design. The main idea is to specify requirements in DC, to define satisfaction probabilities for formulas in this calculus, and establish a calculus with rules that support calculation of the probability for a composite formula from probabilities of its constituents. This ensures that reasoning about probabilities is consistent with requirements and design decisions. In this paper, separate models for requirements and reliability analysis are not introduced. The system model is a finite automaton with fixed transition probabilities. This defines discrete Markov processes as the basis for the calculus. The approach to introducing PDC is as follows: Consider some finite probabilistic timed automata A. The behaviours of A can be represented as a set of M of DC models. The probabilistic principles that manage the working of A used to introduce probability on the subsets of M. Given a DC formula D, the term $\pi(D)(t)$ denotes the probability of those models from M that satisfy D at the interval $[0, t]$. A term of this sort is the component of PDC language. An example PDC formulas is given below:

- $\pi_{s_0}((true; \lceil s \rceil); (\lceil s' \rceil; true))(t) = 0$

In [LRSZ92] the authors focused on the case of discrete time for the sake of simplicity. That is, the model of implementations is based on probabilistic automata, in which transitions (events, actions) take place at discrete time points represented by integers. In a later work, [HC94], PDC was introduced for the case of continuous time. It uses probabilistic automata with transitions occurring in continuous time to model implementations, and then establishes PDC for continuous time. The paper deals with dependability of imperfect implementations of given requirements. The requirements are assumed to be written as formulas in DC. Implementations of given requirements are modelled by continuous semi-Markov processes with finite space, which are expressed as finite automata with stochastic delays of state transitions (such an automata is called continuous time probabilistic automata). A probabilistic model for DC formulas is introduced, so that the satisfaction probabilities of DC formulas with respect to semi-Markov processes can be defined, reasoned about and calculated through a set of axioms and rules (which is an extension of the set of axioms and rules

of DC) of the model. To our best knowledge, no complete proof system for PDC has been proposed so far. As for the decidability, PDC is, not surprisingly, an undecidable logic.

In [HZ07] another probabilistic extension of Duration Calculus is considered to express dependability requirements for real-time systems. This new logic is called *Simple Probabilistic Duration Calculus (SPDC)*. SPDC has an intuitive semantics based on probabilistic timed automata, and has a simple grammar that allows to write formulas to reason about the probability of the satisfaction of a duration formula by a probabilistic timed automaton as well as to specify real-time properties of the system itself. The paper uses the behavioural model proposed in [KNSS02][9] to define the semantics of the logic, and then develops a model checking technique to decide if a set of SPDC models generated by a probabilistic timed automaton satisfies a SPDC formula. Depending on different forms of model sets we can have different model checking problems. The problem is decidable for a class of SPDC formulas of the form *linear duration invariants*, or a formula for bounded liveness. The technique for model checking is an extension of the technique developed earlier in [TH04] to check if a timed automaton satisfies a DC formula in the form of *linear duration invariants* or discretisable DC formulas based on searching the integral reachability graph of the timed automaton. The complexity of the decision procedure is high in general.

9.1.5 The Logic PNL

In [Gue00b] a probabilistic extension of Neighbourhood Logic, called Probabilistic Neighbourhood Logic (*PNL*), is introduced. The study of such an extension is motivated by the need to supply Probabilistic Duration Calculus with a proof system. [Gue00b] presents a complete proof system for the new logic. The proof system of NL was extended to obtain a complete one for PNL. PNL also generalises the notion of finite probabilistic timed automaton in [HC94].

A PNL language is built starting from the same kinds of symbols as a NL language. PNL languages are two-sorted. Together with the well-known sort of durations, they have a sort of probabilities. The function symbols of P_a from automata-related languages take an argument of the duration sort to make a term of the probability sort. We now consider an example. Let b abbreviate a formula which is true at intervals between two consecutive processes. The assumption that the probability for the duration of such a period to be no bigger than x is a function of x which is the interpretation of the function symbol F in a satisfying model can be expressed by the formula

- $p(b \wedge \ell \leq x = F(x))$

PNL has the expressive power of PDC, except for state expressions and their durations. Since PNL is an extension of NL, it is an undecidable logic.

[9] The authors propose a variant of probabilistic timed automata that allows probabilistic choice only at discrete transitions. To resolve the nondeterminism between the passage of time and discrete transitions they use the concept of *adversary* which is essentially a deterministic schedule policy. Then, the set of executions of a probabilistic time automaton according to an adversary forms a Markov chain, and hence the satisfaction of a probabilistic CTL formula by this set can be defined, and then based on the region graph of the timed automaton the satisfaction of a probabilistic CTL formula by the timed automaton can be also verified.

9.2 Probabilistic Dynamic Logics

Following the popularity of probabilistic algorithms for solving problems which are hard or even unsolvable, some attention was given to the formalisation of correctness proof methods for probabilistic programs. Perhaps the first step in this direction is Kozen's definition of a formal semantics for such programs [Koz79]. The next step would be a formal programming logic for probabilistic programs. Indeed, several systems, in particular various *probabilistic dynamic logics*, were proposed. Some historical developments in this area are given below:

Feldman and Harel's *Pr(DL)* [FH82] is a typical example of a first-order probabilistic dynamic logic. Pr(DL) enables reasoning about probabilistic programs or, alternatively, reasoning probabilistically about conventional programs. The syntax of Pr(DL) derives from Pratt's first-order dynamic logic [Pra76] and the semantics extends Kozen's semantics of probabilistic programs. An axiom system for Pr(DL) is presented and shown to be complete relative to an extension of first-order analysis [FH82]. For discrete probabilities it is shown that first-order analysis actually suffices. Examples are presented, both of the expressive power of Pr(DL), and of a proof in the axiom system. Unfortunately, the underlying theory is highly undecidable (equivalent to second-order arithmetic in the discrete case).

On propositional level, some early important logics are Feldman's *P-Pr(DL)* [Fei83] and Kozen's *PPDL* [Koz83]. [Fei83] defines a propositional version P-Pr(DL) of Feldman and Harel's Pr(DL) which preserves many of the powerful characteristics of that logic, such as the ability to use full first-order real-number theory for dealing with probabilities, and deterministic regular programs, while still being decidable. The complexity of the decision procedure for P-Pr(DL) is not addressed. In addition, no axiomatisation is given.

In [Koz83] probabilistic analog *PPDL* of Propositional Dynamic Logic is given. [Koz83] proves a small model property, and gives a polynomial space decision procedure for formulas involving well-structured programs. [Koz83] also gives a deductive calculus and illustrates its use by a program example.

In [Fel84] a Propositional Dynamic Logic with *explicit* probabilities is introduced. Some important features of this logic are as follows: The formal language is based on the Propositional Dynamic Logic; the main objects are programs; and probabilistic operators can be applied on a limited class of formulas. [Fel84] provides a double-exponential space decision procedure for the logic, also by reduction to the decision problem for the theory of real closed fields. The completeness problem for the logic is not solved.

[Gue99] introduces a propositional logic for reasoning about probabilistic processes, such as semi-Markov processes. The author obtains a logic, called *DQP*, by extending the propositional logic of qualitative probabilities (*QP*) [Seg71] with one binary operator $\leq$. The informal reading of $\varphi \leq \psi$, where φ and ψ stand for arbitrary formulas, is that "the probability of φ is not greater than the probability of ψ." A model of this logic is introduced as a set of possible worlds. [Seg71] presents a complete deductive system for QP, which contains an infinite set of axiom schemata. The logic QP can be regarded as a minimal logic for reasoning about probability with respect to how much

it extends classical propositional logic.

The logic *DQP* is obtained as an extension of QP with many $\leq$-operators and operations among them that are analogous to the operations of composition, union, and iteration on modal operators known in propositional dynamic logic. The informal reading of $w \models \varphi \leq_t \psi$ is that "the probability for a transition (experiment) t to transform w into a possible world that satisfies φ is smaller or equal to the probability for t to transform w into a possible world that satisfies ψ." The probabilities are restricted only to the finitely additive ones. Just like QP, DQP can be regarded as a minimal purely propositional system that allows reasoning about probabilistic processes. That is why DQP language, frames and the properties it displays can be expected to be at least in part shared by all other, more application-oriented and more complicated formal systems for reasoning about probabilistic processes.

[Gue99] gives the following reasons for the study of DQP: First, DQP is a natural approach to making Probabilistic Dynamic Logic (*PDL*) [FH82] probabilistic. An alternative approach is Kozen's PPDL [Koz83], but the completeness of PPDL's system of axioms can be so far backed up by example derivations alone. Besides, PPDL's language admits arithmetical terms, and derivations in PPDL rely on arithmetic, which means that it actually has a first order component. Second, the language of QP seems to be simple, and therefore it can capture reasoning about probability, and quantity, in general. Although proofs in QP can be shaped after quantitative intuition, no explicit mention of quantity is needed. All the arithmetic that is intuitively needed for probabilistic reasoning is implemented by means of Boolean operations. [Gue99] finds a ω-complete proof system for DQP. The proof of the ω-completeness theorem involves the construction of a 'canonical' model, that is not finite; hence DQP is not decidable.

9.3 Probabilistic Mu-Calculus

[CIN05] presents a Mu-Calculus-based modal logic, called *Generalised Probabilistic Logic (GPL)*, for describing properties of reactive probabilistic labelled transition systems (RPLTSs). GPL is a uniform framework for defining temporal logics on reactive probabilistic transition systems. In a reactive model the nonprobabilistic choices are external: the environment, not the system itself, selects which action to perform. This point of view stands in contrast to models like Markov decision processes, in which nonprobabilistic choices are internal. GPL is based on the distinction between (probabilistic) 'systems' and (nonprobabilistic) 'observations': using the modal mu-calculus, one may specify sets of observations, and the semantics of this logic then enable statements to be made about the measures of such sets at various system states. This logic is expressive enough to encode some of probabilistic modal and temporal logics (i.e. PCTL*). [CIN05] also presents a model-checking procedure that relies on solving non-linear equations. The logic induces an equivalence on RPLTSs that coincides with accepted notions of probabilistic bisimulation in the literature. Finally, we consider an example: $P_{\geq 1}(\nu X.\phi \wedge [.][.]X)$ says that it is almost always true that ϕ holds at all even time instants ($[.]\phi \equiv \bigwedge_{a \in Act}[a]\phi$, where Act denotes a set of actions).

9.4 Probabilistic Instuitionistic Logics

[MOR03] introduces a probabilistic extension of propositional instuitionistic logic. There is a popular view of instuitionistic logic: In addition to propositions which are proved to be true and those which are proved to be false, there is a third class of propositions which may turn out either way and intuitionism allows us to reason about them. [MOR03] adds probabilistic operators to the propositional instuitionistic language, which enables making statements such as $P_{\geq n}\alpha$ with the intended meaning "the probability of truthfulness of α is at least n". In this logic nesting of probabilistic operators, i.e. higher-order probabilities, are not allowed. At the semantics level, a class of models that combine properties of instuitionistic Kripke models and probabilities are introduced. A sound and complete infinitary axiomatic system is given. It is also proved that the logic is decidable.

9.5 Probabilistic Logics with New Types of Probability Operators

[OR99] introduces a probability logic, called $LP_{P,Q,O}$, with new types of probability operators of the form Q_F, where F is a set from a recursive family O of recursive rational subsets of $[0,1]$. Informally, a formula $Q_F\alpha$ means that "the probability of α is in F." To give semantics to probability formulas a possible-world approach is used. In the paper, the authors assume the so called *measurable case*, i.e. every propositional formula is associated a well-defined probability in every model.

The probability operators introduced in this logic are different than ordinary probability operators. Namely, the new probability operators Q_F cannot be definable in a probability language which contains $P_{\geq}$-operators only. Thus, $LP_{P,Q,O}$ is more expressive.

This logic is suitable for reasoning about discrete sample spaces. For example, consider an experiment which consists of tossing a fair coin an arbitrary, but finite number of times. Then, $Q_F\alpha$ holds in this model, where α means that only heads are observed in the experiment, and F denotes the set $\{\frac{1}{2}, \frac{1}{2^2}, \frac{1}{2^3}, ...\}$. Since Q_F is not definable over the probability language $L_P = \{\neg, \wedge, P_{\geq}\}$, this sentence cannot be described in the probabilistic logics which are obtained by adding probability operators to the classical propositional language.

It turns out that the choice of the family O of recursive rational subsets of $[0,1]$ that appear in the probability operators Q is important, because it affects the decidability and expressiveness of the corresponding probability logic. Every particular choice of the family O produces a different probability language, a different set of probability formulas, and a distinct $LP_{P,Q,O}$ logic.

Although the logic $LP_{P,Q,O}$ is not decidable in general, [OR99] provides a sublanguage which is shown to be decidable. [OR99] also provides a sound and complete axiomatic systems for a number of probability logics augmented with the Q_F-operators.

9.6 Probabilistic Logics for Reasoning About Knowledge and Uncertainty

Halpern et. al., in a series of articles, studied reasoning about knowledge and probability. In [FHM90] the authors focus on technical issues, such as axiomatisations and decision procedures. In [FH91] the authors consider the issue of appropriate models for reasoning about uncertainty in more detail and compare the probabilistic approach to the Dempster-Shafer approach [Sha76]. In [FH94] the authors consider a logic of knowledge and probability that allows arbitrary nesting of knowledge and probability operators. They are able to prove technical results about complete axiomatisations and decision procedures for the resulting logics extending those of this paper. There is also a general look at the interaction between knowledge and probability. The paper [HT93] focuses on knowledge and probability in distributed systems. [AH94] considers issues of reasoning about probability in a first-order context. Below we give more detailed accounts of the articles mentioned:

In [FHM90] a language is considered for reasoning about probability which allows making statements such as "the probability of E_1 is less than $1/3$ and the probability of E_1 is at least twice the probability of E_2.", where E_1 and E_2 are arbitrary events. [FHM90] considers the case where all events are *measurable* (i.e. represent measurable sets), and the more general case where they may not be measurable. The measurable case is essentially a formalisation of the propositional fragment of Nilssons' probabilistic logic [Nil86]. The general *nonmeasurable* case corresponds precisely to replacing probability measures by Dempster-Shafer belief functions [Sha76]. In both cases, a complete axiomatisation is provided, and it is shown that the problem of deciding satisfiability is NP-complete, no worse than that of propositional logic. This is actually done by reducing the problem of validity to a linear programming problem. As a tool for proving the complete axiomatisations a complete axiomatisation is given for reasoning about Boolean combinations of linear inequalities which is of independent interest. This proof and others make crucial use of results from the theory of linear programming. The language is then extended to allow reasoning about conditional probability, and it is shown that the resulting logic is decidable and completely axiomatisable by making use of the theory of real closed fields.

Some related works to that of [FHM90] is as follows: [GKP88] presents a less expressive logic, which is shown to be NP-complete. The measurable case of the logic proposed by [FHM90] can also be viewed as a fragment of the Probabilistic PDL by [Fel84]Temporal . [Koz83] also considers a Probabilistic PDL, which is PSPACE-complete; but this logic is not closed under Boolean combination, and it does not allow linear combinations.

[FH91] introduces a new probabilistic approach to dealing with uncertainty, based on the observation that probability theory does not require that every event be assigned a probability. For a *nonmeasurable* event (one to which we do not assign a probability), one can talk about only the *inner* measure and *outer* measure of the event. In addition to removing the requirement that every event be assigned a probability, this approach circumvents other criticisms of probability-based approaches to uncertainty. For example, the measure of belief in an event turns out to be represented

by an interval (defined by the inner and outer measure), rather than by a single number. Further, this approach allows assigning a belief (inner measure) to an event E without committing to a belief about its $\neg E$ (since the inner measure of an event plus the inner measure of its negation is not necessarily one). Inner measures induced by probability measures turn out to correspond in a precise sense to Dempster-Shafer belief functions [Sha76]. Hence, in addition to providing promising new conceptual tools for dealing with uncertainty, this approach shows that a key part of the important Dempster-Shafer theory of evidence is firmly rooted in classical probability theory.

[FH94] presents an abstract model for reasoning about knowledge and probability in which they assign to each agent-state pair a probability space to be used when computing the probability, according to that agent at that state, that a formula φ is true. The language considered extends the traditional logic of knowledge by allowing explicit reasoning about probability. Probabilities can be explicitly mentioned in formulas, so that the language has formulas that essentially say "according to agent t, formula φ holds with probability at least b." In order to reason about an agent's knowledge as well as the probability he places on certain events, [FH94] extends the logic considered in [FHM90], which is essentially a formalisation of Nilsson's probability logic [Nil86]. The language also allows reasoning about higher-order probabilities, as well as following explicit comparisons of the probabilities an agent places on distinct events. [FH94] presents a general framework for interpreting such formulas, and considers various properties that might hold of the interrelationship between agents' probability assignments at different states. [FH94] provides a complete axiomatisation for reasoning about knowledge and probability, proves a small model property, and obtains some decision procedures. The authors then consider the effects of adding common knowledge and a probabilistic variant of common knowledge to the language.

In [AH94] decidability and expressiveness issues for two first-order logics of probability are considered. In one the probability is on possible worlds, whereas in the other it is on the domain. It turns out that in both cases it takes very little to make reasoning about probability highly undecidable. All the results are proved under the assumption that the probability is discrete. The complexity of the logics gets even worse if arbitrary probability distributions are allowed. One implication of these results is that sound and complete axiom systems cannot be found for these logics (since the existence of such an axiom system would imply that the validity problem would be *r.e.*). There are a few special cases where these results show that it is possible to get complete axiomatisations, for example, in the case where the language consists only of unary predicates and the case where we restrict to bounded domains. In particular, when combined with the standard axioms for reasoning about first-order logic, the axioms for reasoning about probabilities over the domain are complete for a language if it contains only unary predicates; when combined with axioms for equality and an axiom that says that the domain has at most n elements, the axioms are complete for the language if we restrict attention to domains with at most n elements.

10 Conclusion

In this paper we have analysed various temporal formalisms, including propositional/first-order linear temporal logics, branching temporal logics, partial-order temporal logics, interval temporal logics, real-time temporal logics and probabilistic temporal logics. We extrapolated the notions of decidability, axiomatizability, expressiveness, model checking, etc. for each logic analysed, whenever possible. For a comparison of features of the temporal logics we discussed see Table 1. Note that we use the following abbreviations: *No**: Undecidable in general, but decidable for some fragments or specific cases; *No***: No deduction system in general, but available for some fragments or specific cases; *No****: No model checking algorithm in general, but available for some fragments or specific cases; *Yes**: Decidable for some time domains; *Yes***: Available for some time domains; *Yes****: Available for some time domains.

Most of the temporal logics examined are found not fully satisfactory for the specification of real-time systems. The general view is that a "good" temporal logic should have some certain essential characteristics. According to general point of view and the trend in temporal logics in recent years, the following features should be available to build a temporal logic which is suitable for the specification of real-time systems. It should *(i)* be decidable *(ii)* allow limited quantification on temporal variables; *(iii)* have a metric for time or duration; *iv* have probabilistic modal operators in the syntax *(v)* have a limited number of basic operators and the possibility of building special functions; *(vi)* specify quantitative temporal constraints; and *vii* allow efficient model checking techniques.

Table 1: A comparison of features of temporal logics.

Logic	**Logic Order**	**Fund. Entity**	**Temp. Struc.**	**Metric for Time**	**Decidability**	**Deductive Sys.**	**Model Checking**
LTL	Propositional	Point	Linear	No	Yes	Yes	Yes
PTL	Propositional	Point	Linear	No	Yes	Yes	Yes
QTL	First-order	Point	Linear	No	No*	No**	?
CTL	Propositional	Point	Branching	No	Yes	Yes	Yes
CTL*	Propositional	Point	Branching	No	Yes	Yes	Yes
CTL*[P]	Propositional	Point	Branching	No	Yes	Yes	Yes
TCTL	Propositional	Point	Branching	Yes	No	?	Yes
RTCTL	Propositional	Point	Branching	Yes	Yes	?	Yes
TPCTL	Propositional	Point	Branching	Yes	Yes	?	Yes
POTL	Propositional	Point	Partial	No	Yes	Yes	?
HS	Propositional	Interval	Linear	No	No	No	No
CDT	Propositional	Interval	Linear	No	No	Yes	No
PNL	Propositional	Interval	Linear	No	Yes	Yes	No
PITL	Propositional	Interval	Linear	No	No	No**	No
ITL	First-order	Interval	Linear	No	No	Yes	No
NL	First-order	Interval	Linear	Yes	No*	Yes	No
DC	First-order	Interval	Linear	Yes	No*	Yes	No***
IDL	First-order	Interval	Linear	Yes	No*	No	No***
RTL	First-order	Interval	Linear	Yes	No*	No	No***
RTIL	Propositional	Interval	Linear	Yes	Yes	No	?
RTTL	First-order	Point	Linear	Yes	No	Yes	No
TPTL	Propositional	Point	Linear	Yes	Yes*	Yes**	Yes***
MTL	Propositional	Point	Linear	Yes	Yes*	Yes	Yes***
MTIL	Propositional	Interval	Linear	Yes	Yes	?	Yes
XCTL	Propositional	Point	?	Yes	Yes*	?	Yes***
TRIO	First-order	Point	Linear	Yes	No	Yes	No***
TILCO	First-order	Interval	Linear	Yes	No*	Yes	No***
PCTL	Propositional	Point	Branching	No	Yes	?	Yes
PCTL*	Propositional	Point	Branching	No	Yes	?	Yes
PLTL	Propositional	Point	Linear	No	No*	No**	No
PTL_f	Propositional	Point	Branching	No	Yes	Yes	?
PTL_b	Propositional	Point	Branching	No	Yes	Yes	?
PDC	First-order	Interval	Linear	Yes	No	?	?
PNL	First-order	Interval	Linear	Yes	No	Yes	?

References

[Aba89] M. Abadi. The power of temporal proofs. *Theoretical Computer Science*, 65(1):35–83, 1989.

[Abr80] K. Abrahamson. *Decidability and Expressiveness of Logics of Programs.* PhD thesis, University of Washington, 1980.

[ACD90] R. Alur, C. Courcoubetis, and D. L. Dill. Model checking for real-time systems. In *Proceedings 5th Conference on Logic in Computer Science*, pages 12–21. IEEE Computer Society Press, 1990.

[AD96] R. Alur and D.L. Dill. Automata-theoretic verification of real-time systems. *Formal Methods for Real-Time Computing*, pages 55–82, 1996.

[AF94] J. F. Allen and G. Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–57, 1994.

[AFH91] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing.* ACM, 1991.

[AH89] J. F. Allen and J. P. Hayes. Moments and points in an interval-based temporal logic. In *Computational Intelligence*, pages 225–238. Blackwell Publishers, 1989.

[AH90] R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. In *Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science*, pages 390–401. IEEE Computer Society Press, 1990.

[AH94] M. Abadi and J. Y. Halpern. Decidability and expressiveness for first-order logics of probability. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 148–153. IEEE Computer Society Press, 1994.

[AHH96] R. Alur, T. A. Henzinger, and P. H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[All83] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

[All84] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[ANS79] H. Andreka, I. Nemeti, and I. Sain. Mathematical foundations of computer science. *Lecture Notes in Computer Science*, pages 208–218, 1979.

[ANvB98] H. Andreka, I. Nemeti, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, pages 217–274, 1998.

[ASB95] A. Aziz, V. Singhal, and F. Balarin. It usually works: The temporal logic of stochastic systems. In *Proceedings of the 7th International Conference on Computer Aided Verification*, pages 155–165. Springer-Verlag, 1995.

[BA95] A. Bianco and L. D. Alfaro. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science*, pages 499–513. Springer-Verlag, 1995.

[BAMP81] M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. In *Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 164–176. ACM, 1981.

[BC97] R. Barua and Z. Chaochen. Neighbourhood logics. Research Report 120, UNU/IIST, 1997.

[BCCZ99] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without bdds. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 193–207. Springer-Verlag, 1999.

[BDM+98] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A model-checking tool for real-time systems. In *CAV '98: Proceedings of the 10th International Conference on Computer Aided Verification*, pages 546–550. Springer-Verlag, 1998.

[BGG97] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.

[BGMS07] D. Bresolin, V. Goranko, A. Montanari, and G. Sciavicco. On decidability and expressiveness of propositional interval neighbourhood logics. In *Proceedings of the International Symposium on Logical Foundations of Computer Science*, pages 84–99. LNCS, 2007.

[BLL+96] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi. Uppaal - a tool suite for automatic verification of real-time systems. In *Proceedings of the DIMACS/SYCON Workshop on Hybrid systems III : Verification and Control*, pages 232–243. Springer-Verlag, 1996.

[BM05] D. Bresolin and A. Montanari. A tableau-based decision procedure for right propositional neighbourhood logic. In *Proceedings of TABLEAUX 2005*, pages 63–77. LNAI, 2005.

[BMN00] P. Bellini, R. Mattolini, and P. Nesi. Temporal logics for real-time system specification. *ACM Computing Surveys*, 32(1), 2000.

[BMS07a] D. Bresolin, A. Montanari, and P. Sala. An optimal tableau-based decision algorithm for propositional neighbourhood logic. In *Proceedings of STACS 2007: 24th International Symposium on Theoretical Aspects of Computer Science*, pages 549–560. LNCS, 2007.

[BMS07b] D. Bresolin, A. Montanari, and G. Sciavicco. An optimal tableau-based decision procedure for right propositional neighbourhood logic. *Journal of Automated Reasoning*, 38:173–199, 2007.

[BOKR84] M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 457–464. ACM, 1984.

[BRC00] R. Barua, S. Roy, and Z. Chaochen. Completeness of neighbourhood logic. *Journal of Logic and Computation*, 10(2):271–295, 2000.

[Bur82] J. P. Burgess. Axioms for tense logic 2: Time periods. *Notre Dame Journal of Formal Logic*, 23(2):375–383, 1982.

[CE82] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71. Springer-Verlag, 1982.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic. *ACM Transactions on Programming Languages and Systems*, 2(8):244–263, 1986.

[CH98] Z. Chaochen and M. Hansen. An adequate first order interval logic. In *Compositionality: the Significant Difference*, pages 584–608. LNCS, 1998.

[CH04] Z. Chaochen and M. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems*. EATCS Series of Monographs in Theoretical Computer Science. Springer, 2004.

[Cho95] J. Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems*, 20:149–186, 1995.

[CHR91] Z. Chaochen, C. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40:269–276, 1991.

[CHS93] Z. Chaochen, M. Hansen, and P. Sestoft. Decidability and undecidability results for duration calculus. In *Proceedings of the 10th Symposium on Theoretical Aspects of Computer Science*, pages 58–68. LNCS, 1993.

[CIN05] R. Cleaveland, S. P. Iyer, and M. Narasimha. Probabilistic temporal logics via the modal mu-calculus. *Theoretical Computer Science*, 342(2-3):316–350, 2005.

[CP03] G. Chakravorty and P. Pandya. Digitizing interval duration logic. In *Computer Aided Verification*, pages 167–179. Springer-Verlag, 2003.

[CSC00] N. Chetcuti-Serandio and L. Del Cerro. A mixed decision method for duration calculus. *Journal of Logic and Computation*, 10(6):877–895, 2000.

[CT98] J. Chomicki and D. Toman. Temporal logic in information systems. pages 31–70, 1998.

[CY88] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 338–345. IEEE Computer Society, 1988.

[DFKL08] C. Dixon, M. Fisher, B. Konev, and A. Lisitsa. Practical first-order temporal reasoning. In *15th International Symposium on Temporal Representation and Reasoning, TIME '08*, pages 156–163. IEEE Computer Society Press, 2008.

[DFL02] A. Degtyarev, M. Fisher, and A. Lisitsa. Equality and monodic first-order temporal logic. *Studia Logica*, 72(2):147–156, 2002.

[DH96] L. X. Dong and D. V. Hung. Checking linear duration invariants by linear programming. In *Concurrency and Parallelism, Programming, Networking, and Security*, pages 321–332. Springer-Verlag, 1996.

[Dow79] D. Dowty. *Word Meaning and Montague Grammar*. Dordrecht: D. Reidel, 1979.

[Dut95] B. Dutertre. On first-order interval temporal logic. Technical Report CSD-TR-94-3, Department of Computer Science, Royal Holloway College, University of London, 1995.

[EC82] E. A. Emerson and E. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. pages 241–266, 1982.

[EH82] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 169–180. ACM, 1982.

[EH86] E. A. Emerson and J. Halpern. 'Sometimes' and 'Not Never' revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[EJ00] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal of Compututation*, 29(1):132–158, 2000.

[Eme95] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*. North-Holland Pub. Co., 1995.

[EMSS89] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinisvan. Quantitative temporal reasoning. In E. M. Clarke, A. Pnueli, and J. Sifakis, editors, *Proceedings of the Workshop on Automatic Verification Methods for Finite State Systems*. LNCS, 1989.

[FDP01] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM on Transactions of Computational Logic*, 2(1):12–56, 2001.

[Fei83] Y. A. Feidman. A decidable propositional probabilistic dynamic logic. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 298–309. ACM, 1983.

[Fel84] Y. A. Feldman. A decidable propositional dynamic logic with explicit probabilities. *Information Control*, 63(1-2):11–38, 1984.

[FH82] Y. A. Feldman and D. Harel. A probabilistic dynamic logic. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 181–195. ACM, 1982.

[FH91] R. Fagin and J. Y. Halpern. Uncertainty, belief and probability. *Computational Intelligence*, 7(3):160–173, 1991.

[FH94] R. Fagin and J. Y. Halpern. Reasoning about knowledge and probability. *Journal of the ACM*, 41(2):340–367, 1994.

[FHM90] R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.

[FHMV96] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. MIT Press, 1996.

[Fis91] M. Fisher. A resolution method for temporal logic. In *Proceedings of Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, 1991.

[FM94] J. L. Fiaderio and T. Maibum. Action refinement in a temporal logic of objects. In *Temporal Logic*. LNCS, 1994.

[FORS02] J. C. Filliatre, S. Owre, H. Ruess, and N. Shanka. Ics: Integrated canonizer and solver. In *Computer Aided Verification*, pages 246–249. Springer-Verlag, 2002.

[Fra96] M. Fraenzle. Synthesizing controllers from duration calculus. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 168–187. LNCS, 1996.

[Frä02] M. Fränzle. Take it np-easy: Bounded model construction for duration calculus. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 245–264. Springer-Verlag, 2002.

[Gal84] A. Galton. *The Logic of Aspect*. Claredon Press, Oxford, 1984.

[Gal90] A. Galton. A critical examination of Allen's theory of action and time. *Artificial Intelligence*, 42:159–198, 1990.

[GD99] D. Guelev and V. H. Dang. On the completeness and decidability duration calculus with iteration. In *Advances in Computer Science*, pages 139–150. LNCS, 1999.

[GHR94] D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 1. Clarendon Press, Oxford, 1994.

[GKP88] G. Georgakopoulos, D. Kavvadias, and C. H. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4(1):1–11, 1988.

[GKWZ02] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier, 2002.

[GMM90] C. Ghezzi, D. Mandrioli, and A. Morzenti. TRIO, a logic language for executable specifications of real-time systems. *Journal of Systems and Software*, pages 107–123, 1990.

[GMS03a] V. Goranko, A. Montanari, and G. Sciavicco. A general tableau method for propositional interval temporal logics. In *Proceedings of the International Conference Tableaux 2003*, pages 102–116. LNAI, 2003.

[GMS03b] V. Goranko, A. Montanari, and G. Sciavicco. Propositional interval neighbourhood temporal logics. *Journal of Universal Computer Science*, 9(9):1137–1167, 2003.

[GMS04] V. Goranko, A. Montanari, and G. Sciavicco. A road map of interval temporal and duration calculi. *Journal of Applied Non-Classical Logics*, 14, 2004.

[GPSS80] D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. *Conference Record of the 7th Annual ACM Symposium on Principles of Programming Languages*, pages 163–173, 1980.

[Grä99] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1999.

[GRF00] D. Gabbay, M. Reynolds, and M. Finger. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 2. Clarendon Press, Oxford, 2000.

[Gue98] D. P. Guelev. Probabilistic interval temporal logic. Technical Report 144, UNU/IIST, 1998.

[Gue99] D. P. Guelev. A propositional dynamic logic with qualitative probabilities. *Journal of Philosophical Logic*, 28:575–604, 1999.

[Gue00a] D. P. Guelev. A complete proof system for first order interval temporal logic with projection. Technical Report 202, UNU/IIST, 2000.

[Gue00b] D. P. Guelev. Probabilistic neighbourhood logic. In *Proceedings of the 6th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 264–275. Springer-Verlag, 2000.

[Ham71] C. L. Hamblin. Instants and intervals. *Stadium Generale*, 27:127–134, 1971.

[Han91] H. A. Hansson. *Time and Probability in Formal Design and Distributed Systems*. PhD thesis, Department of Computer Science, Uppsala University, Sweden, 1991.

[HC94] D. V. Hung and Z. Chaochen. Probabilistic duration calculus for continuous time. In *Formal Aspects of Computing*, pages 21–44, 1994.

[Hen91] T. A. Henzinger. *The Temporal Specification and Verification of Real-Time Systems*. PhD thesis, Department of Computer Science, Stanford University, 1991.

[HJ89] H. Hansson and B. Jonsson. A framework for reasoning about time and reliability. In *Proceedigns of Real-time Systems Symposium*, pages 102–111. IEEE, 1989.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:102–111, 1994.

[HLN+90] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, and M. Trachtenbrot. Statemate: A working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16:403–414, 1990.

[HLP90] E. Harel, O. Lichtenstein, and A. Pnueli. Explicit clock temporal logic. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science*, pages 402–413. IEEE Computer Society Press, 1990.

[HMM83] J. Y. Halpern, Z. Manna, and B. Moszkowski. A high-level semantics based on interval logic. In *Proceedings of 10th ICALP*, pages 274–291, 1983.

[HPS83] D. Harel, A. Pnueli, and Y. Stavi. Process logic: Expressiveness, decidability, completeness. *Journal of Computer and System Sciences*, 25:145–180, 1983.

[HS84] S. Hart and M. Sharir. Probabilistic temporal logics for finite and bounded models. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 1–13. ACM, 1984.

[HS91] J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, 1991.

[HT93] J. Y. Halpern and M. R. Tuttle. Knowledge, probability and adversaries. *Journal of ACM*, 40(4):917–960, 1993.

[Hum79] L. Humberstone. Interval semantics for tense logic: Some remarks. *Journal of Philosophical Logic*, 8:171–196, 1979.

[HWZ00] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, pages 85–134, 2000.

[HZ07] D. V. Hung and M. Zhang. On verification of probabilistic timed automata against probabilistic duration properties. In *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 165–172. IEEE Computer Society, 2007.

[JM86] F. Jahanian and A. K. Mok. Safety analysis of timing properties in real-time systems. *IEEE Transactions on Software Engineering SE-12*, 9:890–904, 1986.

[JS88] F. Jahanian and D. Stuart. A method for verifying properties of modechart specifications. In *Proceedings 9th Real-time Systems Symposium*, pages 12–21. IEEE Computer Society Press, 1988.

[Kam68] J. A. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

[Kam79] H. Kamp. Events, instants and temporal reference. In *Semantics from Different Points of View*, pages 376–417. Springer, 1979.

[KNSS02] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282(1):101–150, 2002.

[Koy90] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2:255–299, 1990.

[Koz79] D. Kozen. Semantics of probabilistic programs. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 101–114. IEEE Computer Society, 1979.

[Koz83] D. Kozen. A probabilistic pdl. In *Proceedings of the fifteenth annual ACM symposium on Theory of Computing*, pages 291–297. ACM, 1983.

[KP86] Y. Kornatzky and S. Pinter. An extension to partial order temporal logic (potl). Research Report 596, Department of Electrical Engineering, Technion-Israel Institute of Technology, 1986.

[Kro87] F. Kroger. *Temporal Logic of Programs*. Springer-Verlag, 1987.

[Lad87] P. Ladkin. Logical time pieces. *AI Expert*, 2(8):58–67, 1987.

[LP00] O. Lichtenstein and A. Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. D. Zuck. The glory of the past. In *Proceedings of the Conference on Logic of Programs*, pages 196–218. Springer-Verlag, 1985.

[LRSZ92] Z. Liu, A. P. Ravn, E. V. Sorensen, and C. Zhou. A probabilistic duration calculus. Technical report, University of Warwick, 1992.

[LS83] D. J. Lehmann and S. Shelah. Reasoning with time and chance (extended abstract). In *Proceedings of the 10th Colloquium on Automata, Languages and Programming*, pages 445–457. Springer-Verlag, 1983.

[LS95] F. Laroussinie and P. Schnoebelen. A hierarchy of temporal logics with past. *Theoretical Computer Science*, 148(2):303–324, 1995.

[LSWZ02] C. Lutz, H. Sturm, F. Wolter, and M. Zakharyaschev. A tableau decision algorithm for modalized $\mathcal{ALC}$ with constant domains. *Studia Logica*, 72(2):199–232, 2002.

[LW08] S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Transactions on Computational Logic*, 9(2):1–27, 2008.

[Mat96] R. Mattolini. *TILCO: A Temporal Logic for the Specification of Real-Time Systems (TILCO: una Logica Temporale per la Specifica di Sistemi di Tempo Reale)*. PhD thesis, University of Florence, 1996.

[McD82] D. McDermott. A temporal logic for reasoning about process and plans. *Cognitive Science*, 6:101–155, 1982.

[Mer92] S. Merz. Decidability and incompleteness results for first-order temporal logics of linear time. *Journal of Applied Non-classical Logic*, pages 139–156, 1992.

[MN96] R. Mattolini and P. Nesi. Compositional inductive verification of duration properties of real-time systems. In *Proceedings of the Second IEEE International Conference on Engineering of Complex Computer Systems*, pages 18–25. Chapman and Hall, 1996.

[MN00] R. Mattolini and P. Nesi. An interval logic for real-time system specification. *IEEE Transactions on Software Engineering*, 2000.

[MNP05] O. Maler, D. Nickovic, and A. Pnueli. Real time temporal logic: Past, present, future. In *Formal Modeling and Analysis of Timed Systems*, pages 2–16. Springer-Verlag, 2005.

[MOR03] Z. Marković, Z. Ognjanović, and M. Rasković. A probabilistic extension of intuitionistic logic. *Mathematical Logic Quarterly*, 49:415–424, 2003.

[Mos83] B. Moszkowski. *Reasoning about Digital Circuits*. PhD thesis, Computer Science Department, Stanford University, 1983.

[Mos00a] B. Moszkowski. A complete axiomatization of interval temporal logic with infinite time. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*, pages 242–251. IEEE Computer Society Press, 2000.

[Mos00b] B. Moszkowski. Compositional inductive verification of duration properties of real-time systems. In *Proceedings of the 27th International Conference on Automata, Languages and Programming*, pages 223–234. LNCS, 2000.

[Mos03] B. Moszkowski. A hierarchical completeness proof for interval temporal logic with finite time. In *Proceedings of the ESSLLI Workshop on Interval Temporal Logics and Duration Calculi*, pages 41–65, 2003.

[MP81a] Z. Manna and A. Pnueli. Verification of concurrent programs: Temporal proof principles. pages 200–252, 1981.

[MP81b] Z. Manna and A. Pnueli. Verification of concurrent programs: The temporal framework. pages 215–273, 1981.

[MP83] Z. Manna and A. Pnueli. Proving precedence properties: The temporal way. In *Proceedings of the Tenth Colloquium on Automata Languages and Programming*, pages 491–512. Springer, 1983.

[MR99] M. Marx and M. Reynolds. Undecidability of compass logic. *Journal of Logic and Computation*, 9(6):897–914, 1999.

[MS87] P. M. Melliar-Smith. Extending interval logic to real time systems. In *Proceedings of the Conference on Temporal Logic Specification*, pages 224–242. Springer, 1987.

[MSV02] A. Montanari, G. Sciavicco, and N. Vitacolonna. Decidability of interval temporal logics over split-frames via granularity. In *Proceedings of the 8th Europian Conference on Logics in AI*, pages 259–270. Springer, 2002.

[Nil86] N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[Ogn06] Z. Ognjanović. Discrete linear-time probabilistic logics: Completeness, decidability and complexity. *Journal of Logic and Computation*, 16(2):257–285, 2006.

[OR99] Z. Ognjanović and M. Rasković. Some probability logics with new types of probability operators. *Journal of Logic and Computation*, 9(2):181–195, 1999.

[Ost89] J. S. Ostroff. *Temporal Logic for Real-Time Systems*. Advanced Software Development Series. Research Studies Press Limited, 1989.

[Ott01] M. Otto. Two variable first-order logic over ordered domains. *Journal of Symbolic Logic*, 66(2):685–702, 2001.

[OW87] J. S. Ostroff and W. Wonham. Modeling and verifying real-time embedded computer systems. In *Proceedings of the 8th IEEE Real-Time Systems Symposium*, pages 124–132. IEEE Computer Society Press, 1987.

[OW05] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *Proceedings 20th Annual IEEE Symposium on Logic in Computer Science*, pages 188–197. IEEE Computer Society Press, 2005.

[Pan01] P. K. Pandya. Specifying and deciding quantified discrete-time duration calculus formulas using DCVALID. In *Real-Time Tools*, 2001.

[Pan02] P. K. Pandya. Interval duration logic: Expressiveness and decidability. In *Proceedings of TPTS*, 2002.

[Par78] R. Parikh. A decidability result for second order process logic. In *Proceedings of 19th FOCS*, pages 177–183. IEEE Computer Society Press, 1978.

[Pau94] L. C. Paulson. *Isabelle: A Generic Theorem Prover*. LNCS, 1994.

[Pen95] W. Penczek. Branching time and partial-order in temporal logics. pages 179–228, 1995.

[Pli97] R. Pliuskevicius. On the completeness and decidability of a restricted first order linear temporal logic. In *Proceedings of the 5th Kurt Gödel Colloquium on Computational Logic and Proof Theory*, pages 241–254. Springer-Verlag, 1997.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.

[Pnu83] A. Pnueli. On the extremely fair treatment of probabilistic algorithms. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 278–290. ACM, 1983.

[Pra76] V. R. Pratt. Semantical consideratiosn on floyd-hoare logic. Technical report, 1976.

[Pra79] V. R. Pratt. Process logic. In *Proceedings of 6th POPL*, pages 93–100. ACM, 1979.

[Pri67] A. N. Prior. *Past, Present and Future*. Oxford University Press, 1967.

[PW84] S. S. Pinter and P. Wolper. A temporal logic for reasoning about partially ordered computations (extended abstract). In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, pages 28–37. ACM, 1984.

[Rab98] A. Rabinovich. Non-elementary lower bound for propositional duration calculus. *Information Processing Letters*, 66:7–11, 1998.

[Rey96] M. Reynolds. Axiomating first-order temporal logic: Until and since over linear time. pages 279–302, 1996.

[Rey01] M. Reynolds. An axiomatization of full computation tree logic. *Journal of Symbolic Logic*, 66:1011–1057, 2001.

[Rey05] M. Reynolds. An axiomatization of pctl. *Information and Computation*, 201(1):72–119, 2005.

[RG89] R. Razouk and M. Gorlick. Real-time interval logic for reasoning about executions of real-time programs. *SIGSOFT Software Engineering Notes 14*, 8:10–19, 1989.

[RG93] A. Rao and M. Georgeff. A model-theoretic approach to the verification of situated reasoning systems. In *Proceedings of IJCAI*, 1993.

[Rop80] P. Roper. Intervals and tenses. *Journal of Philosophical Logic*, pages 451–469, 1980.

[RP86] R. Rosner and A. Pnueli. A choppy logic. In *Proceedings of the First IEEE Symposium on Logic in Computer Science*, pages 306–313. IEEE Computer Society Press, 1986.

[SC85] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32:733–749, 1985.

[Seg71] K. Segerberg. Qualitative probability in a modal setting. *Proceedings of the Second Scandinavian Logic Symposium*, pages 575–604, 1971.

[Sha76] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[SHP98] M. Satpathy, D. V. Hung, and P. K. Pandya. Some decidability results for duration calculus under synchronous interpretation. In *Proceedings of the 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 186–197. Springer-Verlag, 1998.

[Sim87] P. Simons. *Parts, A Study in Ontology*. Claredon Press, Oxford, 1987.

[SMS82] R. L. Schwartz and P. M. Melliar-Smith. From state machines to temporal logic: Specification methods for protocol standards. *IEEE Trans. Commun. 30*, pages 2486–2496, 1982.

[SMSV83] R. L. Schwartz, P. M. Melliar-Smith, and F. H. Voght. An interval logic for higher-level temporal reasoning. In *Proceedings of the Second ACM Symposium on Principles of Distributed Computing*, pages 173–186. ACM Press, 1983.

[SPS05] B. Sharma, P. Paritosh, and C. Supratik. Bounded validity checking of interval duration logic. In *TACAS 2005: Tools and Algorithms for the Construction and Analysis of Systems*, pages 301–316. Springer-Verlag, 2005.

[SVW85] A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for büchi automata with applications to temporal logic (extended abstract). In *Proceedings of the 12th Colloquium on Automata, Languages and Programming*, pages 465–474. Springer-Verlag, 1985.

[Sza95] A. Szalas. Temporal logic of programs: A standard approach. pages 1–50, 1995.

[TH98] P. H. Thai and D. V. Hung. Checking a regular class of duration calculus models for linear duration invariants. In *Proceedings of the International Symposium on Software Engineering for Parallel nd Distributed Systems*, pages 61–71. IEEE Computer Society Press, 1998.

[TH04] P. H. Thai and D. V. Hung. Verifying linear duration constraints of timed automata. In *Proceedings of ICTAC'04*, pages 295–309. Springer-Verlag, 2004.

[vB83] J. F. van Benthem. *The Logic of Time*. Kluwer Academic Publishers, Dordrecht, 1983.

[vB91] J. F. van Benthem. *The Logic of Time: A Model-Theoretic Investigation into the Varieties of Temporal Ontology and Temporal Discourse*. Kluwer, second edition, 1991.

[Ven90] Y. Venema. Expressiveness and completeness of an interval tense logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, 1990.

[Ven91] Y. Venema. A modal logic for choppping intervals. *Journal of Logic and Computation*, 1:453–476, 1991.

[Ven98] Y. Venema. *Temporal Logic*. Blackwell Guide to Philosophical Logic, Blacwell Publishers, 1998.

[Vit05] N. Vitacolonna. *Intervals: Logics, Algorithms and Games*. PhD thesis, Department of Mathematics and Computer Science, University of Udine, 2005.

[Wal47] A. G. Walker. Durees et instants. *La Revue Scientifique*, (3266), 1947.

[WZ99] F. Wolter and M. Zakharyaschev. Modal description logics: Modalizing roles. *Fundamenta Informaticae*, pages 411–438, 1999.

[WZ01] F. Wolter and M. Zakharyaschev. Axiomatizing the monadic fragment of first-order temporal logic. *Annals of Pure and Applied Logic*, 2001.

[WZ02] F. Wolter and M. Zakharyaschev. Axiomatizing the monodic fragment of first-order temporal logic. *Annals of Pure and Applied Logic*, pages 133–145, 2002.

[Zho94] C. Zhou. Linear duration invariants. In *Proceedings of the Third International Symposium Organized Jointly with the Working Group Provably Correct Systems on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 86–109. Springer-Verlag, 1994.