A Decidable Approach to Real-time System Specification

Savas Konur*
*Department of Computer Science, University of Liverpool, Liverpool L69 3BX
konur@liverpool.ac.uk

1 Introduction

Most formal methods employed for the specification and development of distributed systems are either event-based or state-based (For a more detailed account for the concepts 'events' and 'states', please see [2]). For system development both views are important [6]. More generally, in early phases of systems development, event-based methods are more suitable; in contrast, in later phases state-based methods are more suitable [1].

We can, therefore, say that a formal method should cover both event-based and state-based views in order to model the behaviour of real-time systems. Most temporal logics employed for the specification of real-time systems, however, do not support both methods. They are either event-based or state-based. For this reason, finding a temporal logic which covers both views, and which can be used in specifying real-time system properties is an important research subject in this area.

On the other hand, choosing a mathematical model of time has been also a primary concern in this context. A basic way to characterise temporal logics is whether points (instants) or intervals are used to model time. It has been turned out that the interval-based scheme provides us with a richer representation formalism than the point-based scheme. Especially, the notion of interval is necessary to represent continuous processes and to make temporal statements which are based on intervals.

Unfortunately, in the area of interval logics, undecidability is very common, and decidability is very rare. Most interval logics have indeed turned out to be (highly) undecidable. In the literature various methods have been proposed to achieve decidability for interval logics. However, most of the methods, such as translating interval logics into point-based ones, cause some syntactic and semantic restrictions. A major challenge in this area is thus to genuinely identify interval-based decidable logics, that is, logics which are not explicitly translated into point-based logics or other semantic restrictions.

In order to overcome these drawbacks we introduce a decidable event- and state-based interval temporal logic, called TL [2]. The logic TL covers both event- and state-based views. In TL intervals are used as primitive objects of the model by allowing quantification over only interval objects. Unlike many other interval logics, we do not translate the logic TL into a point-based variant, and we therefore try to minimise semantic restrictions. Since we restrict ourselves to genuine intervals, this logic is theoretically more challenging.

Another important feature of TL is that it incorporates the notion of *duration*, denoting the length of a state (or an event), and *accumulation*, denoting the total duration of a state. These notions have been found useful for reasoning about time durations of dynamic systems.

In [2] we prove that TL has the finite model property, and the satisfiability problem is decidable. Its unique 'quasi-guarded' character of the quantification is very important to guarantee the decidability. [2] also provides a tableau based decision procedure to find a complexity bound for satisfiability, showing that this problem can be solved in NEXPTIME.

2 Real-Time System Applications

In order to show the usefullness of the logic TL in specifying properties of real-time systems, in [2] we apply it to well-known mine pump and gas burner examples. The mine pump was first described in [3], where a discrete-time interval logic is used. However, in distributed systems the discrete-time approach cannot specify precisely the behaviour of the computer programs that run on different components of the system with different clocks. Since TL is a continuous-time interval logic, it resolves this problem. The gas burner system was described in [5, 4], where it was modeled by an undecidable interval logic. Since TL is a decidable logic, it removes this limitation. To the best of our knowledge, TL is the first decidable logic which formalizes the gas burner system.

2.1 Water Pump

A mine has water seepage which must be removed by operating a pump. If the water is above danger-mark, an alarm must be sounded, an operating of pump must stop. The mine also has pockets of methane which escape. When there is methane, an alarm must be sounded, and all electrical activity must be shut down to prevent explosion.

The mine pump system was first described in [3], where a logic called *QDDS* is used. QDDS is a discrete-time variant of Duration Calculus, and it can specify the behaviour of computer programs running on a component with a unique clock. However, in this case it can only specify approximately the behaviour of physical components, and if the system is distributed, it cannot precisely specify the behaviour of the computer programs that run on different components of the system with different clocks. Thus, in specifying the hybrid and distributed systems, the continuous time should be used.

In the logic TL, system states are modeled by intervals which have the finite variability meaning that in any finite interval of time there is only a finite number of discontinuous points. Thus, the time in TL is continuous, and we can reason about the behavioural timing properties of computer programs. Therefore, TL is more preferable than QDDC in this perspective.

An example specification is given as follows: The alarm will sound within δ seconds of the water level becoming dangerous. Alarm will persist while the water level is dangerous:

$$\texttt{Alarmcontrol}_1 \equiv [DWater]\,\langle Alarm\rangle^{f}_{>}\,(\ell \leq \delta)$$

2.2 Gas Burner System

A gas burner is a device to generate a flame to heat up products using gas. The gas burner system is a safety-critical system as excessive leaking of gas may lead to an accident. The gas burner system was first specified in [4], where an undecidable logic Duration Calculus [7] used. To the best of our knowledge, so far the gas burner system has not been formalized with a decidable language. So, TL is the first decidable logic to specify the requirements of the gas burner system.

One example specification is given as follows: An ignite failure happens when gas does not ignite within 0.5 s:

$$\texttt{IgniteFail} \equiv [Gas]\,[Ignition]\,(\ell \leq 0.5 \vee ([Flame]^{df}_{>}\,(\ell > 0.5)) \wedge [Flame]^{se}\bot)$$

3 Conclusion and Future Research

In this paper, we briefly discussed a new decidable interval temporal logic, called TL. TL uses intervals as primitive objects of model. TL can specify both event-based and state-based statements. TL also incorporates the notion of duration. To show the usability of TL we applied it to well-know gas-burn and water pump systems. We addressed that using TL removed some limitations with previous specifications. Future research will include implementing the tableau method presented in [2], so that (un)satisfiability of a formula can be returned automatically.

References

[1] E. Kindler and T. Vesper. Estl: A temporal logic for events and states. *ICATPN'98, LNCS 1420*, pages 365–384, 1998.

[2] S. Konur. *An Interval Temporal Logic for Real-time System Specification*. PhD thesis, University of Manchester, 2008.

[3] P. K. Pandya. Specifying and deciding quantified discrete-time duration calculus formulas using dcvalid. *Workshop on Real-Time Tools*, 2001.

[4] A. P. Ravn, H. Rischel, and K. M. Hansen. Specifying and verifying requirements of real-time systems. *IEEE Transactions on Software Engineering*, 19(1), 1993.

[5] E. V. Sorensen, A. P. Ravn, and H. Rischel. Control program for a gas burner: Part 1. informal requirements. Technical Report ID/DTH EVS2, ProCoS Case Study 1, 1990.

[6] W. van der Aalst. Three good reasons for using a petri-net-based workflow management system. *Proceedings of the International Working Conference on Information and Process Integration in Enterprises*, pages 179–201, 1996.

[7] C. Zhou and M. R. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems*. EATCS Series, Springer, 2004.