Journal of Applied Logic 11 (2013) 30–51

Contents lists available at SciVerse ScienceDirect

Journal of Applied Logic

www.elsevier.com/locate/jal

Deductive temporal reasoning with constraints

Clare Dixon *, Boris Konev, Michael Fisher, Sherly Nietiadi

Department of Computer Science, University of Liverpool, Liverpool, UK

ARTICLE INFO

Article history:
Received 14 February 2012
Accepted 7 June 2012
Available online 14 July 2012

Keywords:
Temporal logic
Constraints
Theorem proving
Tableau

ABSTRACT

When modelling realistic systems, physical constraints on the resources available are often required. For example, we might say that at most N processes can access a particular resource at any moment, exactly M participants are needed for an agreement, or an agent can be in exactly one *mode* at any moment. Such situations are concisely modelled where literals are constrained such that at most N, or exactly M, can hold at any moment in time. In this paper we consider a logic which is a combination of standard propositional linear time temporal logic with cardinality constraints restricting the numbers of literals that can be satisfied at any moment in time. We present the logic and show how to represent a number of case studies using this logic. We propose a tableau-like algorithm for checking the satisfiability of formulae in this logic, provide details of a prototype implementation and present experimental results using the prover.

1. Introduction

Temporal logic allows the concise specification of *temporal* order. However, if we need to represent cardinality restrictions we have to introduce a large number of formulae to the specification making it hard to read and understand, and difficult for provers to deal with. In addition, while temporal logic has turned out to be a very useful notation across a number of areas, particularly the specification of concurrent and distributed systems [44,43,27], the complexity of many temporal logics is often considered to be too high for practical verification (see for example [11,8]). Consequently, simple modal logics, finite state automata, or even Boolean satisfiability, are typically used in the verification of such systems [46]. This is because the decision problem for propositional linear temporal logic (PTL) is PSPACE-complete [31] whereas techniques in many of the above areas are much simpler.

So, the question we are concerned with in this work is the following: can we represent and reason about such cardinality restrictions (here we use the term constraints) in a compact and transparent way, retaining the useful descriptive powers of temporal logics, while making the reasoning more efficient in practice? Here we propose and utilise a succinct way of specifying cardinality constraints. We show that if examples are in (or close to) a particular normal form, using this representation of constraints simplifies the reasoning. Additionally, we experiment with an implemented prototype prover for this logic.

To specify the constraints we allow statements stating that *less than or equal to k* literals, or *exactly k* literals from some subset of literals, are true at any moment in time. Note that this approach involves reasoning *in the presence* of constraints rather than reasoning *about* them. Thus, the resulting logic represents a combination of standard temporal logic with (fixed) constraints that restrict the numbers of literals that can be satisfied at any moment in time. This new approach is particularly useful for (for example):

* Corresponding author.
E-mail addresses: cldixon@liverpool.ac.uk (C. Dixon), konev@liverpool.ac.uk (B. Konev), mfisher@liverpool.ac.uk (M. Fisher), Sherly.Nietiadi@liverpool.ac.uk (S. Nietiadi).

1570-8683/$ – see front matter
http://dx.doi.org/10.1016/j.jal.2012.07.001

- ensuring that a fixed bound is kept on the number of propositions satisfied at any moment to prevent overload;
- in finite collections of communicating automata, ensuring that no more than k automata are in a particular state;
- modelling restrictions on resources, for example at most k vehicles are available or there are at most k seats available;
- modelling the necessity to elect exactly k from n participants.

Motivating Example. Consider a fixed number, n, of robots that can each *work*, *rest* or *recharge*. We assume that there are only $k < n$ recharging points and only $j < n$ workstations. Let:

- $work_i$ represent the fact that robot i is working;
- $rest_i$ represent the fact that robot i is resting; and
- $recharge_i$ represent the fact that robot i is recharging.

Now, we typically want to specify that exactly j of the n robots are working at any one time. In the syntax given later, such a logic might be defined as TLC($\mathcal{W}^{=j}, \mathcal{R}^{\leqslant k}$), where

$$\mathcal{W}^{=j} = \{work_1, \ldots, work_n\}^{=j}$$
$$\mathcal{R}^{\leqslant k} = \{recharge_1, \ldots, recharge_n\}^{\leqslant k}$$

This represents the logic with the constraints that exactly j robots must work at any moment and at most k can recharge at any moment.

This paper extends preliminary material from our earlier paper [19]. The contributions of the paper are: to define and analyse a logic which combines both temporal logic and constraints; to show how a number of case studies can be elegantly modelled using this logic; to provide a tableau-like satisfiability algorithm for formulae in this logic giving proofs of correctness; to provide algorithms for a prototype implementation of this; and give experimental results comparing the implementation with other temporal provers.

The paper is organised as follows. Section 2 gives the syntax and semantics of the constrained temporal logic, together with a normal form for this logic. In Section 3 we provide a number of case studies and show how they are specified in this logic. In Section 4 we provide an algorithm for checking satisfiability of this logic and consider its complexity. In Section 5 we give details of an implementation of the satisfiability checker for this logic and experimental details comparing this implementation to other tableau reasoners for propositional linear time temporal logic. Finally, in Section 6, we provide concluding remarks and discuss both related and future work.

2. A constrained temporal logic

Temporal Logic with Cardinality Constraints (TLC) [19] is PTL with some additional constraints, which restrict the numbers of literals that can be satisfied at any moment in time. TLC is parameterised by (not necessarily disjoint) sets $C^{\propto m}$ where $\propto \in \{=, \leqslant\}$ and $m \in \mathbb{N}$. The formulae of TLC($C_1^{\propto_1 m_1}, C_2^{\propto_2 m_2}, \ldots$) are constructed under the restriction that, depending on $\propto_i$, exactly m_i literals from every set C_i are true in every state ($\propto_i$ is $=$) or less than or equal to m_i literals from every set C_i are true in every state ($\propto_i$ is $\leqslant$). For example, consider TLC($C_1^{=2}, C_2^{\leqslant 1}$), where $C_1^{=2} = \{p, q, r\}^{=2}$ and $C_2^{\leqslant 1} = \{x, y, z\}^{\leqslant 1}$. Then, at any moment of time, exactly two of p, q, or r are true, and less than or equal to one of x, y, or z is true. In addition to these constraints, there exists a set of propositions, $\mathcal{A}$, which are standard, unconstrained propositions. Note that, the 'less than' constraint $C^{<m}$ can be expressed as $C^{\leqslant m-1}$ and the 'more than or equal to' constraint $C^{\geqslant m}$ can be expressed as $\bar{C}^{\leqslant n-m}$, where n is the number of literals in C and by definition $\bar{C} = \{\bar{x} \mid x \in C\}$, $\bar{p} = \neg p$ and $\overline{\neg p} = p$. Note that by using both $C^{\leqslant m}$ and $C^{\geqslant m}$ (encoded as $\bar{C}^{\leqslant n-m}$) we could also obtain $C^{=m}$. However we choose not to do this, as we aim for a clear and intuitive way of expressing constraints and this appears to obscure the meaning. Further, the constraint $C^{=1}$ seems to be common in applications (see Section 3) which gives additional weight for the $C^{=m}$ construct to be primitive.

We note that we can express the information in our constrained sets as temporal formulae. For example given the constraint $C_1^{=2} = \{p, q, r\}^{=2}$ above, this can be represented by the following temporal formula

$$\Box\big((p \vee q) \wedge (p \vee r) \wedge (q \vee r)\big) \wedge \Box(\neg p \vee \neg q \vee \neg r)$$

2.1. TLC syntax

A constraint $C_i^{\propto_i m_i}$ is a tuple $(C_i, \propto_i, m_i)$, where C_i is a set of literals with a cardinality restriction $\propto_i m_i$, such that $\propto_i \in \{=, \leqslant\}$ and $m_i \in \mathbb{N}$. For TLC, the future-time temporal connectives we use include '$\bigcirc$' (in the next moment) and '$\mathcal{U}$' (until). Formally, TLC($C_1^{\propto_1 m_1}, \ldots, C_n^{\propto_n m_n}$) formulae are constructed from the following elements:

- a set, $\mathsf{Props} = \{p \mid p \in C_i^{\propto_i m_i}\} \cup \{p \mid \neg p \in C_i^{\propto_i m_i}\} \cup \mathcal{A}$ of propositional symbols (where $1 \leqslant i \leqslant n$ and $\mathcal{A}$ are termed 'unconstrained' propositions);