DQPTL: Dependency Quantified Propositional Linear-time Temporal Logic

Anonymous Author(s)
Submission Id: 8

ABSTRACT

We present an approach to specify the information flow between players in a concurrent game with LTL objectives. We use a *dependency matrix* $\mathcal{D}$ to reflect this information flow. On this basis, we define the so-called *Dependency Quantified Propositional Linear-time Temporal Logic* (DQPTL) whose formulae are of the form $\mathcal{D}\exists\vec{p}\varphi$, where $\mathcal{D}$ is a dependency matrix, $\exists\vec{p}$ informs which players form the coalition and φ is an LTL formula, whose semantics is a concurrent game. We show that our setting captures the standard semantics of Quantified Propositional Linear-time Temporal Logic (QPTL), for some adequate matrices $\mathcal{D}$. Moreover, we provide an effective criterion to decide if a DQPTL formula is *consistent* in the sense that the concurrent game yields an entire labeling of the time line, so that the winning condition φ can be evaluated.

KEYWORDS

Temporal logic, Dependency, Multiplayer Games

ACM Reference Format:
Anonymous Author(s). 2021. DQPTL: Dependency Quantified Propositional Linear-time Temporal Logic. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), London, UK, May 3–7, 2021*, IFAAMAS, 3 pages.

1 INTRODUCTION

Quantifiers in logic naturally yield a game semantics between existential and universal players. In the setting of strategic reasoning, such as Alternating-time Temporal Logic [1] or Strategy Logic [2], the quantifiers apply on strategies and the coarse-grained skolemization mechanism from classical logic is not adequate to reflect subtle dependencies, as noticed by [3, 4]. For example, while the Quantified Propositional Linear-time Temporal Logic (QPTL) [5] formula $\exists a\forall b(a \leftrightarrow \mathrm{X}b)$ is not satisfiable, as Player a has no information about how Player b will label the timeline with proposition b, there might exist a strategy of Player a if she would be informed about how Player b labels the next time point.

In this contribution, we approach these considerations by studying the simple setting of such games on the timeline with LTL objectives. We design *Dependency Quantified Propositional Linear-time Temporal Logic* (DQPTL) where the information flow between players is made explicit through a *dependency matrix* $\mathcal{D}$. Atomic formulae of DQPTL are of the form $\mathcal{D}\exists\vec{p}\varphi$ or $\mathcal{D}\forall\vec{p}\varphi$, where $\mathcal{D}$ is a dependency matrix, $\vec{p}$ is a list of propositions, and φ is an LTL-formula. General DQPTL formulae are Boolean combinations of atomic DQPTL formulae. The semantics of a DQPTL atomic formula $\mathcal{D}\exists\vec{p}\varphi$ is provided as a concurrent game. In the arena of this game, called *meta arena*, a player is in charge of incrementally labeling the timeline with the unique proposition she owns, provided she has enough information about other players moves as specified by the dependency matrix $\mathcal{D}$. Any (possibly infinite) play where every player has achieved the full labeling of the timeline thus describes a model to evaluate winning condition φ. The roadmap consists in defining *dependency matrices* and the associated meta arena. We distinguish *progressing* dependency matrices whose plays in the associated so-called *progressing* arena yield an entire labeling of the timeline. Incidentally, we provide a polynomial-time decision procedure to characterize progressing dependency matrices, based on the existence of an absorbing cycle in a graph derived from the matrix. We then trun to the full definition of the logic DQPTL, that preliminary requires a tuned notion of *uniform strategies* in the meta arena. Finally, we show an embedding of QPTL with the standard semantics of [5] into DQPTL.

From now on, we fix an infinite set of propositions AP.

2 DEPENDENCY MATRICES

In our setting, a player is responsible for a single atomic proposition. A *dependency matrix* specifies dependencies between player moves.

Definition 1. *A* dependency matrix *over AP is a finite-dimensional matrix $\mathcal{D} = (\mathcal{D}[a,b])_{a,b\in P}$, for some finite $P \subseteq AP$, and whose coefficients range over $\mathbb{Z} \cup \{\pm\infty\}$.*

Intuitively, $\mathcal{D}[a,b] \in \mathbb{Z} \cup \{\pm\infty\}$ expresses the constraint that Player a cannot choose the a-labeling at time point t if she is not informed about the b-labeling made by Player b up to time point $t + \mathcal{D}[a,b]$. In particular, it is natural to require that $\mathcal{D}[d,d] = -1$ for every Player d to reflect that each player is perfect recall. Also, setting $\mathcal{D}[a,b] = -\infty$ expresses that Player a plays independently of Player b, and finally setting $\mathcal{D}[a,b] = +\infty$ means that Player a makes her choice on the basis of Player b's labeling of the whole timeline.

One may also look at dependency matrices as a way to concurrently fill the timeline where some players may announce their moves in advance, a kind of commitment. Suppose that (1) Alice privately announces her next move to Bob and her two next moves to Carol; (2) Bob is informed about one time point ahead regarding Alice's labeling and about the current time point regarding Carol's labeling; (3) Carol is not informed at all about how the two others put their labels. The situation is reflected by the dependency matrix of Fig. 1.

A given dependency matrix $\mathcal{D}$ can be seen as a seed to start playing in the so-called *meta arena* $\mathcal{A}_{\mathcal{D}}$ that we now describe.

3 THE META ARENA $\mathcal{A}_{\mathcal{D}}$

In the rest of the paper, we fix a dependency matrix $\mathcal{D}$. A *position* in the associated meta arena $\mathcal{A}_{\mathcal{D}}$ is a labeling of the timeline by each player of P on a section of the timeline starting from the initial time point 0 up to some time point (possibly $+\infty$) called the *advancement* of the player, and that may differ between players. In the initial position of the arena, the advancement of each player is -1, namely no time point has been labeled yet by any of the players.

The dynamics of the arena is based on the successive updates of the dependency matrix in such a way that in a position with advancement t_d for each Player d, we have:

$$\mathcal{D}'[a,b] = t_a + \mathcal{D}[a,b] - t_b \tag{1}$$

for any pair of Players a and b. With such an update $\mathcal{D}'$, we can identify every Player a who can *progress*, namely who can increase her advancement by choosing the a-labeling at time point t_a+1. Those players have only strictly negative values on their corresponding line in the matrix $\mathcal{D}'$: indeed, by definition of $\mathcal{D}$, in order to label time point t_a+1, Player a needs Player b to have played at least up to time point $t_a + 1 + \mathcal{D}[a,b]$, that is $t_b \geq t_a + 1 + \mathcal{D}[a,b]$, hence $t_b > t_a + \mathcal{D}[a,b]$, so that $\mathcal{D}'[a,b] < 0$. Since this constraint is required for any other Player b, this results in a strictly negative valued line a in $\mathcal{D}'$.

When Player a labels time point $t_a + 1$, her advancement is increased which requires an update of matrix $\mathcal{D}'$ to maintain invariant (1): one can easily establish that the update amounts to uniformly increase by 1 line a and to uniformly decrease by 1 column a.

Notice that players can progress concurrently, allowing for concurrent moves in the meta arena.

We define a concurrent move in $\mathcal{A}_{\mathcal{D}}$ as the concurrent progress of all players able to do so. Note that there are as many moves as labeling choices of progressing players.

For the matrix of Fig. 1, both Player a and Player c can progress, ending up with the updated matrix below.

$$\mathcal{D}' = \left(\begin{array}{cccc} & a & b & c \\ a & -1 & 0 & -2 \\ b & 0 & -1 & -1 \\ c & -\infty & -\infty & -1 \end{array}\right)$$

However, from this updated matrix, Players a and Player b can no longer progress in $\mathcal{A}_{\mathcal{D}}$. Even if Player c can keep progressing for ever, this prevents the play from defining a labeling of the timeline for all propositions. Such blocking situation is investigated in the next section.

4 PROGRESSING DEPENDENCY MATRIX

We are only interested in meta arenas where in every (possibly infinite) play yields a full labeling of the time line, otherwise said where the advancement of each player is unbounded.

Such arenas can be characterized by a so-called *progressing property* of their dependency matrix as follows. Each dependency matrix is assigned a weighted graph, called the *dependency graph*, whose generic construction is omitted here, due to lack of space, but exemplified (see Fig. 1). This graph has a size polynomial in the size of the matrix. An *absorbing cycle* in a dependency graph is a cycle whose weight is non-positive.

Proposition 1. *A dependency matrix has the progressing property iff its dependency graph has no absorbing cycle. As a consequence, deciding the progressing property takes polynomial time.*

$$\left(\begin{array}{cccc} & a & b & c \\ a & -1 & -1 & -2 \\ b & 1 & -1 & 0 \\ c & -\infty & -\infty & -1 \end{array}\right)$$

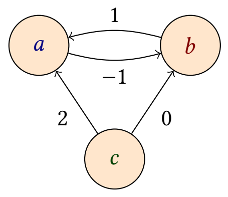

Figure 1: A dependency matrix on the left, and its dependency graph G on the right. Because of the absorbing cycle (a, b, a) in G, Players a and b will inevitably stop progressing for ever in $\mathcal{A}_{\mathcal{D}}$.

While progressing meta arenas ensure that a player has enough information to choose her labeling, we need to circumvent the legitimate amount of information a player can rely on, so that the entire machinery faithfully reflects the dependencies specified in the matrix.

5 UNIFORM STRATEGIES

Classically, a strategy is a mapping from histories to moves. Note that in a meta arena, a position contains all the labeling choices made by all players so far (up to their respective advancement along the timeline). Additionally, given a position, one can reconstruct the entire history by resorting to the dependency matrix. It seems therefore relevant to define a strategy as a mapping from positions to moves, where a move of Player a is some Boolean value for proposition a at time point $t_a + 1$ if she has enough information according to $\mathcal{D}$; otherwise the move is ε.

However, for a strategy to rely only on the legitimate information Player a has access to (as specified by $\mathcal{D}$), it needs being *uniform*: the strategy should output the same move for any two positions v_1, v_2 such that for every other Player b, it holds that $v_1(b)(t) = v_2(b)(t)$ for all $t \leq t_a + \mathcal{D}[a,b]$ where $v(b)(t)$ is the truth value of b at time point t in v.

6 SYNTAX AND SEMANTICS OF DQPTL

The syntax of DQPTL is given by $\phi ::= \mathcal{D}\exists\vec{p}\varphi \mid \mathcal{D}\forall\vec{p}\varphi \mid \neg\phi \mid \phi \wedge \phi'$, where $\mathcal{D}$ is a progressing dependency matrix over the propositions of $\varphi \in$ LTL, and $\vec{p} \subseteq$ AP. We only give the semantics of DQPTL atomic formulae.

- $\mathcal{D}\exists\vec{p}\varphi$ holds iff there is a joint strategy $\mathcal{D}$ uniform for coalition $\vec{p}$ in $\mathcal{A}_{\mathcal{D}}$ whose outcomes satisfy φ,
- $\mathcal{D}\forall\vec{p}\varphi$ holds iff the outcomes of any joint strategy $\mathcal{D}$ uniform for coalition $\vec{p}$ in $\mathcal{A}_{\mathcal{D}}$ satisfy φ.

Sentences of QPTL can be effectively translated into DQPTL in polynomial time: Take for instance the QPTL formula $\exists a \forall b\, \varphi$ (without loss of generalities, QPTL formulae in prenex form suffice). The DQPTL formula associated is $\mathcal{D}\exists\{a\}\varphi$ with

$$\mathcal{D} = \left(\begin{array}{ccc} & a & b \\ a & -\infty & -\infty \\ b & +\infty & -\infty \end{array}\right).$$

REFERENCES

[1] Rajeev Alur, Thomas A Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM (JACM)*, 49(5):672–713, 2002.

[2] Krishnendu Chatterjee, Thomas A Henzinger, and Nir Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010.

[3] Patrick Gardy. *Semantics of Strategy Logic*. Theses, Université Paris-Saclay, June 2017.

[4] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic (TOCL)*, 15(4):34, 2014.

[5] A Prasad Sistla, Moshe Y Vardi, and Pierre Wolper. The complementation problem for büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49(2-3):217–237, 1987.