# Monitoring Arithmetic Temporal Properties on Finite Traces

**Paolo Felli,**[1] **Marco Montali,**[2] **Fabio Patrizi,**[3] **Sarah Winkler**[2]

[1]Università di Bologna – Bologna – Italy, paolo.felli@unibo.it
[2]Free University of Bozen-Bolzano – Bolzano – Italy, {montali,winkler}@inf.unibz.it
[3]Sapienza Università di Roma – Rome – Italy, patrizi@diag.uniroma1.it

## Abstract

We study the monitoring of linear-time arithmetic properties against finite traces generated by an unknown dynamic system. The monitoring state is determined by considering at once the trace prefix seen so far, and all its possible finite-length, future continuations. This makes monitoring at least as hard as satisfiability and validity. Traces consist of finite sequences of assignments of a fixed set of variables to numerical values. Properties are specified in a logic we call $\mathrm{ALTL}_f$, combining $\mathrm{LTL}_f$ (LTL on finite traces) with linear arithmetic constraints that may carry *lookahead*, i.e., variables may be compared over multiple instants of the trace. While the monitoring problem for this setting is undecidable in general, we show decidability for (a) properties without lookahead, and (b) properties with lookahead that satisfy the abstract, semantic condition of *finite summary*, studied before in the context of model checking. We then single out concrete, practically relevant classes of constraints guaranteeing finite summary. Feasibility is witnessed by a prototype implementation.

## Introduction

Dynamic systems in AI are often constituted by autonomous agents and heterogeneous components whose internal specification is either unknown or not accessible, so that well-known techniques to ascertain their correctness at design-time, such as model checking and testing, are not applicable. This calls for approaches to check desired properties at runtime, by *monitoring* the executions of the black-box system under scrutiny. A widely known, solid approach to monitoring is that of *runtime verification* (RV), where given a logical property, the monitor emits a provably correct verdict (Leucker and Schallhart 2009). Formalisms based on LTL and its extensions have indeed been extensively applied when monitoring multi-agent systems (Dastani, Torroni, and Yorke-Smith 2018), software components (Leucker and Schallhart 2009) and business processes (Ly et al. 2015).

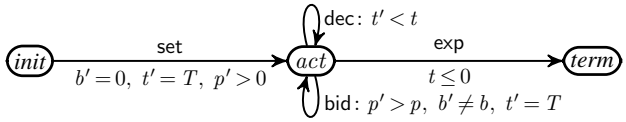
Leucker and Schallhart (2009) highlight two essential semantic desiderata of linear-time monitors. First, a monitored trace has a finite length, which calls for a finite-trace semantics. Second, a trace is the prefix of an unknown, full trace. Thus, the verdict of the monitor should be *anticipatory* (Bartocci et al. 2018), i.e., depend not only on the data seen so far, but also on its (infinitely many) possible continuations. In addition, runtime monitors need to do as little computation as possible during monitoring, and therefore all the required machinery and anticipatory test conditions need to be pre-computed. Similarly, the monitor itself cannot depend on the specific trace being monitored, but should allow for monitoring multiple traces against the same property.

In this spectrum, we build on the widely studied logic $\mathrm{LTL}_f$, or LTL over finite traces (De Giacomo and Vardi 2013), where both the monitored prefixes and their infinitely many suffixes have a finite (yet unbounded) length. Specifically, we focus on the more sophisticated setting of RV-LTL by Bauer, Leucker, and Schallhart (2010), where the verdict of the monitor of a property $\psi$ at each point in time is one of four possible values: *currently satisfied* ($\psi$ is satisfied now but may be violated in a future continuation), *permanently satisfied* ($\psi$ is satisfied now and will necessarily stay so), and the two complementary values of *permanent* and *current violation*. Thus, RV-LTL monitoring is at least as hard as satisfiability and validity. For propositional $\mathrm{LTL}_f$, an RV-LTL monitor can be constructed from the deterministic finite-state automaton (DFA) corresponding to the property, by labeling each DFA state by an RV-LTL value, depending on whether it is final and can reach final and non-final states (Maggi et al. 2011; De Giacomo et al. 2022).

Prior work on $\mathrm{LTL}_f$ monitoring like the above has often focused on propositional traces, where states are described by propositional interpretations. This coarse-grained representation creates a large abstraction gap w.r.t. dynamic systems, which often generate *data-aware* traces where states contain richer objects from an infinite domain, e.g., strings or numbers. On the other hand, for richer logics partial methods have been devised, but without guarantees that the monitoring task can be solved, e.g. (Reger, Cruz, and Rydeheard 2015). Aiming to bridge this gap, we *(i)* introduce an extension of $\mathrm{LTL}_f$, called $\mathrm{ALTL}_f$, which combines linear-time operators with linear arithmetic constraints over data variables *within and across states*, *(ii)* study how to construct automata-based RV-LTL monitors for such formulae and *(iii)* identify decidable classes of properties, given that $\mathrm{ALTL}_f$ monitoring is in general undecidable. The next example illustrates the challenge that we address.

**Example 1.** *Consider the monitoring of suspicious bidding patterns in an on-line auction. The process is supposed*

*(but not guaranteed) to execute according to the following transition system:*




*There are three variables: the current price $p$ (i.e., the last bid); the id of the last bidder $b$, and a timer $t$. In the edge labels, a primed variable (e.g. $p'$) indicates the next value of the respective variable that is written by the transition (e.g. $p$). The transition* set *fixes $p$ to some base price and initializes the timer to the duration $T$ of a round. Then, either the timer is decreased by transition* dec*, or a bidder increases the price, upon which the timer is reset to $T$. Finally,* exp *terminates the auction when the timer expires. (We assume that all variables not explicitly written in transition labels keep their value.) By adding a variable $s$ for the state, the process can be encoded in ALTL$_f$:*

$$(s = init) \wedge \mathsf{G}(\psi_{\mathsf{set}} \vee \psi_{\mathsf{dec}} \vee \psi_{\mathsf{bid}} \vee \psi_{\mathsf{exp}}) \qquad \text{(P)}$$

*where, e.g., $\psi_{\mathsf{set}} = (s = init \wedge s' = act \wedge t' = T \wedge p' > 0)$, and other transitions are encoded similarly. A first task could then be to monitor formula $P$ to check whether the actual execution deviates from this specification.*

*In addition, during an auction, it is desirable to identify users that exhibit* shilling behavior*, i.e., they drive up the auction price for the seller. Given a user $u$, consider the following behavior patterns (Xu and Cheng 2007): (OB)* Overbidding*: $u$ places an overbid (i.e.,increasing the price by at least 20%) right before the timer expires. This can be expressed as $OB_u := \mathsf{F}(b' = u \wedge t \leq \epsilon \wedge p' \geq p \cdot 1.2)$, for some small $\epsilon$. (AU)* Aggressive underbidding*:* underbidding *means that the price increase is less than 3$, and* aggressive *that the time gap between two bids is small (for simplicity, we assume here in the next state). This can be expressed as $AU_u := \mathsf{G}(b \neq u \wedge b > 0 \rightarrow p' \geq p + 3 \wedge b = u)$. Both (OB) and (AU) likely imply shilling. The following is an example process execution (also called a* trace*) with $T = 2$.*

| $s$ | $init$ | $act$ | $act$ | $act$ | $act$ | $act$ | $act$ | $act$ |
|---|---|---|---|---|---|---|---|---|
| $t$ | 0 | 2 | 2 | 2 | 1 | 2 | 2 | 2 |
| $p$ | 0 | 10 | 30 | 32 | 32 | 36 | 40 | 50 |
| $b$ | 0 | 0 | 1 | 2 | 2 | 3 | 1 | 2 |
| $OB_2$ | CV | CV | CV | CV | CV | CV | CV | PS |
| $AU_2$ | CS | CS | CS | CS | CS | CS | PV | PV |
| $shill_2$ | CS | CS | CS | CS | CS | CS | CV | PS |

*The lower part of this table shows monitoring results for the trace, namely the RV values obtained for observing (OB) and (AU) for user 2, as well as for $shill_2 := OB_2 \vee AU_2$. Note that the trace above conforms with the specification $P$. A further task could be to monitor combinations of such properties, such as $P \wedge OB_2$, which gives the ability of anticipatorily detect violations that cannot be ascribed to single properties, but only to the interplay of multiple properties and the current trace (De Giacomo et al. 2022).*

The techniques presented in this paper show how to construct anticipatory monitors for full ALTL$_f$, to tackle examples like the above. Anticipatory monitoring for plain LTL has been recognized as highly relevant in business process management (BPM) (Ly et al. 2015; Maggi et al. 2014), but these works ignore the recent trend of modelling data in processes; with the present work we want to mitigate this shortcoming. However, monitoring is also highly relevant in other areas, like program verification (Havelund, Reger, and Rosu 2019) and verification of cyber-physical systems (Geist, Rozier, and Schumann 2014).

Technically, we follow Demri and D'Souza (2007) and consider traces consisting of assignments of a fixed set of variables to numeric values. A starting point is the approach in (Felli, Montali, and Winkler 2022b), where automata-based techniques are used to model check dynamic systems against an extension of LTL$_f$ with linear arithmetic constraints, e.g., "the value of sensor $x$ differs from that of sensor $y$ by 5 units until it exceeds 25". However, that approach has two prohibitive technical limitations. The first one is operational: their automata are too weak for monitoring as they are faithful only for reachability of final, but not of non-final states. The second shortcoming is semantical: arithmetic conditions can only relate variables within the same state, but not across instants in the trace. For this reason Demri and D'Souza (2007) and, more recently, Geatti, Gianola, and Gigante (2022), allow properties to compare variables across states. Our logic ALTL$_f$ follows this route, enabling variable *lookahead* to reference variable values in different instants within arbitrary linear arithmetic constraints, (cf. e.g. $p' > p$ in Ex. 1). Our approach is thus substantially different from (Demri and D'Souza 2007), which restricts to variable-to-variable/constant comparisons to ensure decidability of satisfiability. Instead, we allow arbitrary linear arithmetic constraints with lookahead, making satisfiability (and monitoring) undecidable even for simple formulae.

Despite the rich research body on monitoring, there are few approaches that are anticipatory, feature full LTL$_f$, and incorporate arithmetic with lookahead (Falcone et al. 2021). The seminal LOLA approach (D'Angelo et al. 2005) supports arithmetic with lookahead and finite traces, but is not anticipatory. Also Faymonville et al. (2019) and (Basin et al. 2015) have only bounded anticipation. More related is MarQ (Reger, Cruz, and Rydeheard 2015), a strong, anticipatory RV tool for finite traces with relations and arithmetic, but its approach has no decidability guarantees.

We overcome these limitations as follows: (1) we show that ALTL$_f$ monitoring is decidable for linear arithmetic constraints without lookahead, and provide techniques to construct automata-based RV-LTL monitors. (2) we show, for formulae with lookahead, undecidability of ALTL$_f$ satisfiability/monitoring. (3) To mitigate this, we study how the abstract property of *finite summary*, introduced in (Felli, Montali, and Winkler 2022b), can be employed for ALTL$_f$ formulae with lookahead that satisfy this property, proving decidability of monitoring and providing a technique to construct automata-based RV-LTL monitors. (4) We use this general result to show decidability for different concrete classes of formulae enjoying finite summary, obtained by restricting either the language of constraints, or the way these interact with each other via lookahead. (5) A prototype tool witnesses feasibility of our approach.

The structure of the paper reflects Contributions (1)–(5) in this order, after the introduction of ALTL$_f$ and the monitoring problem and a section about the construction of automata for ALTL$_f$ properties afterwards.

## Preliminaries

We consider the sorts $int$ and $rat$ with domains $d(int) = \mathbb{Z}$ and $d(rat) = \mathbb{Q}$. For a set of variables $V$, let $V_\sigma \subseteq V$ be those in $V$ of sort $\sigma$. We first define arithmetic constraints:

**Definition 2.** Given a set of variables $V$, expressions $e_\sigma$ of sort $\sigma$ and *constraints* $c$ are defined as follows:

$$e_\sigma := v_\sigma \mid k_\sigma \mid e_\sigma + e_\sigma \mid k_\sigma \cdot e_\sigma$$
$$c := e_\sigma = e_\sigma \mid e_\sigma \neq e_\sigma \mid e_\sigma < e_\sigma \mid e_\sigma \leq e_\sigma \mid e_{int} \approx_n e_{int} \mid e_{int} \not\approx_n e_{int}$$

where $k_\sigma \in d(\sigma)$ is a constant, $v_\sigma \in V_\sigma$ a variable of appropriate sort, and $\approx_n$ denotes congruence modulo $n \in \mathbb{N}$.

The set of all constraints over $V$ is denoted by $\mathcal{C}(V)$. E.g., $x < y + z$, and $x - y \neq 2$ are constraints independent of the sort of $x, y, z$, but $u \approx_3 v + 1$ requires that $u$ and $v$ have sort $int$. We also consider boolean formulas with constraints as atoms; these are in the realm of SMT with linear arithmetic, which is decidable and admits *quantifier elimination* (Presburger 1929): if $\varphi$ is a formula with free variables $X \cup \{y\}$ and constraint atoms, there is some $\varphi'$ with free variables $X$ that is logically equivalent to $\exists y.\varphi$, i.e., $\varphi' \equiv \exists y.\varphi$.

From now on, let $V$ be a fixed, finite, non-empty set of variables called *state variables*. An *assignment* $\alpha$ maps every $v \in V$ to a value $\alpha(v)$ in the domain of its sort. The set of *variables with lookahead* $\mathcal{V}$ consists of all $v'^{\cdots\prime}$ such that $v \in V$ is decorated with $i$ prime symbols $'$, for $i \geq 0$, and we say that $v$ *has lookahead* $i$. Intuitively, $v'^{\cdots\prime}$ with $i$ primes refers to the value of $v$ looking $i$ instants into the future. For instance, $v'$ refers to the value of $v$ in the next step, while $v''$ is the value of $v$ two steps ahead.

**Definition 3.** Let $\mathcal{L}$ be the set of properties $\psi$ defined by the following grammar, where $c \in \mathcal{C}(\mathcal{V})$ is a constraint over variables with lookahead:

$$\psi ::= c \mid \psi \wedge \psi \mid \neg\psi \mid \mathsf{X}_s\psi \mid \psi \,\mathsf{U}\, \psi$$

Here $\mathsf{X}_s$ is the next operator (we use $\mathsf{X}_w$ for weak next) and $\mathsf{U}$ is the until operator. When needed, we write $\mathcal{L}(V)$ instead of $\mathcal{L}$ to make explicit the set of variables in constraints. The usual transformations apply, namely $\psi_1 \vee \psi_2 \equiv \neg(\psi_1 \wedge \psi_2)$, $\mathsf{X}_w\psi \equiv \neg(\mathsf{X}_s\neg\psi)$, $\mathsf{F}\psi \equiv \top \,\mathsf{U}\, \psi$, $\mathsf{G}\psi \equiv \neg\mathsf{F}\neg\psi$; moreover $\top \equiv (v = v)$ for some $v \in V$, and $\bot \equiv \neg\top$. A property $\psi$ *has lookahead* $m$ if the maximal lookahead of a variable in $\psi$ is $m$. E.g., all properties in Ex. 1 are in $\mathcal{L}(V)$ for $V = \{p, t, b\}$. Properties $p' > p$ and $AU_2$ have lookahead 1, but $t \leq 0$ has lookahead 0. Note that lookahead strictly extends the expressiveness of the language: for instance, the simple property $p' > p$ cannot be expressed by using $\mathsf{X}_s$ and $\mathsf{X}_w$ alone.

Properties are evaluated over *traces*: a trace $\tau$ of length $n \geq 1$ is a finite sequence $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ of assignments with domain $V$. We write $\tau(i)$ to denote $\alpha_i$, for $0 \leq i < n$. A constraint $c$ is *well-defined* at instant $i$ of $\tau$ if $0 \leq i < n$ and all variables with lookahead $j$ in $c$ satisfy $i + j < n$. The upper part of the table in Ex. 1 is an example of a trace, and e.g., $p' \geq p$ is well-defined at all but the last instant.

**Definition 4.** If an expression $e$ is well-defined at instant $i$ of a trace $\tau$ of length $n$, its *evaluation* $[\tau, i](e)$ at instant $i$ is:

$$[\tau,i](k) = k \qquad [\tau,i](e{+}e_2) = [\tau,i](e_1){+}[\tau,i](e_2)$$
$$[\tau,i](v'^{\cdots\prime}) = \tau(i{+}j)(v) \qquad [\tau,i](k \cdot e) = k \cdot [\tau,i](e)$$

where $v'^{\cdots\prime}$ has lookahead $j$. Then, $\tau$ *satisfies* $\psi \in \mathcal{L}$, denoted $\tau \models \psi$, iff $\tau, 0 \models \psi$ holds, where, for $0 \leq i$:

$\tau, i \models e_1 \odot e_2$ iff $0 \leq i < n$ and either
- $e_1 \odot e_2$ is not well-defined for $\tau$ and $i$, or
- $[\tau, i](e_1) \odot [\tau, i](e_2)$ holds

$\tau, i \models \psi_1 \wedge \psi_2$ iff $\tau, i \models \psi_1$ and $\tau, i \models \psi_2$

$\tau, i \models \neg\psi$ iff $\tau, i \not\models \psi$

$\tau, i \models \mathsf{X}_s\psi$ iff $i < n - 1$ and $\tau, i{+}1 \models \psi$

$\tau, i \models \psi_1 \,\mathsf{U}\, \psi_2$ iff $i < n$ and either $\tau, i \models \psi_2$, or $i < n{-}1$, $\tau, i \models \psi_1$ and $\tau, i{+}1 \models \psi_1 \,\mathsf{U}\, \psi_2$

For instance, the trace in Ex. 1 satisfies $OB_2$ but its proper prefixes do not. Note that lookahead variables have a *weak* semantics, as a constraint $c$ holds if it contains a variable $v'^{\cdots\prime}$ that refers to an instant beyond the end of the trace. If instead a strict semantics is desired, one can replace $c$ by $c \wedge \mathsf{X}_s \ldots \mathsf{X}_s \top$, where the number of $\mathsf{X}_s$ operators equals the lookahead of $c$.

We now define our main task, that is to monitor how the satisfaction of a given property changes along a trace, that is, by considering the trace (fragment) $\tau$ seen *so far*. As customary (Bauer, Leucker, and Schallhart 2010), we consider the set $RV = \{\text{PS}, \text{CS}, \text{CV}, \text{PV}\}$ of four distinct *monitoring states*: current satisfaction (CS), permanent satisfaction (PS), current violation (CV) and permanent violation (PV).

**Definition 5.** A property $\psi \in \mathcal{L}$ is in monitoring state $s \in RV$ after a trace $\tau$, written $\tau \models [\![\psi = s]\!]$, if
- $s = \text{CS}$, $\tau \models \psi$, and $\tau\tau' \not\models \psi$ for some trace $\tau'$;
- $s = \text{PS}$, $\tau \models \psi$, and $\tau\tau' \models \psi$ for every trace $\tau'$;
- $s = \text{CV}$, $\tau \not\models \psi$, and $\tau\tau' \models \psi$ for some trace $\tau'$;
- $s = \text{PV}$, $\tau \not\models \psi$, and $\tau\tau' \not\models \psi$ for every trace $\tau'$.

E.g., $\tau \models [\![\psi = \text{CS}]\!]$ means that $\psi$ is currently true after $\tau$ but there exists a possible continuation of $\tau$ (i.e., a trace $\tau\tau'$) after which $\psi$ is false. After a trace, a property $\psi$ is in exactly one possible monitoring state. For instance, the monitoring states in the table in Ex. 1 reflect Def. 5 for the given trace $\tau$, e.g., $\tau \models [\![OB_2 = \text{PS}]\!]$.

Given $\tau$ and $\psi$, the *monitoring problem* is to compute the state $s \in RV$ s.t. $\tau \models [\![\psi = s]\!]$. It is *solvable* if one can construct a procedure for $\psi$ that computes the monitoring state for any given trace. Unfortunately, we have that in general:

**Theorem 6.** *The monitoring problem is not solvable.*

To see this, call a property $\psi \in \mathcal{L}$ *satisfiable* if $\tau \models \psi$ for some (non-empty) trace $\tau$. It can be shown that satisfiability of $\mathcal{L}$ properties with lookahead 1 is undecidable, by a reduction from reachability in 2-counter machines. Now, for a formula of the form $\psi = \mathsf{X}_s\psi'$ and a trace $\tau_0$ of length 1, $\psi'$ is satisfiable iff $\tau_0 \models [\![\psi = \text{CV}]\!]$, so satisfiability of $\psi'$ reduces to monitoring $\tau_0$ against $\psi$.

In this paper we study when monitoring is solvable. First, we show that without loss of generality one can restrict to

properties with lookahead 1: Indeed, a property with lookahead $m{>}1$ can be transformed into one with $m{=}1$ by adding fresh variables and extending to these the assignments in the trace in an appropriate way. In general, $m{-}1$ fresh bookkeeping variables are needed for each variable with lookahead $m{>}1$. Intuitively, constraints are then rewritten to mention only consecutive instants, using chains of next operators. E.g., $\psi{=}\mathsf{G}(x''>x)$ is equivalent to $\widehat{\psi}{=}\mathsf{G}(u'>x \wedge \mathsf{X}_w(u=x'))$. The trace $\tau\colon \{x=2\},\{x=0\},\{x=3\}$ satisfies $\psi$ and $\widehat{\tau}\colon \{x=2,u=0\},\{x=0,u=3\},\{x=3,u=0\}$ satisfies $\widehat{\psi}$, with $\widehat{\tau}$ obtained from $\tau$ by assigning to $u$ the subsequent value of $x$, if it exists. A detailed proof of the next result can be found in (Felli et al. 2022).

**Lemma 7.** Let $\psi \in \mathcal{L}(V)$ have lookahead $m$. There is some $\psi_1 \in \mathcal{L}(V \cup X)$ with lookahead 1 for a set of fresh variables $X$ such that for every trace $\tau$ over $V$ there is a trace $\widehat{\tau}$ over $V \cup X$ satisfying $\tau \models \psi$ iff $\widehat{\tau} \models \psi_1$, where $|\widehat{\tau}| = |\tau|$ and $\widehat{\tau}(i)$ agrees with $\tau(i)$ on $V$, for all $i$.

## Automata for properties

In the sequel, we will develop first monitoring techniques for $\mathcal{L}$ properties without lookahead, and then with lookahead 1. Although the former case is simpler and could be addressed by an ad-hoc approach resembling techniques for propositional LTL$_f$ (Maggi et al. 2011), we adopt a uniform approach for both cases. To this end, we first present the construction of an automaton that represents a given property $\psi$; this will be crucial in the later sections. As a preprocessing step, two transformations are applied to the property:
(1) We assume $\psi$ in negation normal form; so it may also contain $\vee$, $\mathsf{G}$, and $\mathsf{X}_w$, so far considered syntactic sugar. Note that for constraints with lookahead, negation cannot be pushed to the constraint level, e.g., $\neg(x'=x)$ is not equivalent to $x'{\neq}x$ as the latter is satisfied by any trace of length 1 but the former is not. In the construction, we thus treat constraints with and without negation separately. For clarity, we write $neg(c)$ to flip the comparison operator in $c$: $neg(t_1=t_2) := t_1 \neq t_2$, $neg(t_1<t_2) := t_2 \leq t_1$, $neg(t_1 \approx_n t_2) := t_2 \not\approx_n t_1$, and vice versa, for all $t_1, t_2$.
(2) We want a monitor to return the monitoring state for the trace (prefix) $\tau$ seen so far, without depending on future variable assignments. However, the evaluation of a constraint with lookahead depends on the next, not yet seen, assignment (see Def. 4). To avoid this counterintuitive peculiarity and make the monitoring task clearer, we transform constraints with look*ahead* 1 into constraints with look*back* 1.
To this end, we consider variables $v_{pre}$ and $v_{cur}$, for all $v \in V$. For a constraint $c$, let $back(c)$ be defined as (a) if $c$ has no lookahead, $back(c)$ is obtained from $c$ by replacing $v$ with $v_{cur}$, and (b) if $c$ has lookahead, $back(c) := \mathsf{X}_w\bar{c}$, with $\bar{c}$ obtained from $c$ by replacing $v$ with $v_{pre}$ and $v'$ with $v_{cur}$; for all $v \in V$. We denote the property where each constraint $c$ is replaced by $back(c)$ by $\psi_{back}$, and we write $V_{pre}{=}\{v_{pre} \mid v \in V\}$ and the same for $V_{cur}$. Then $\psi$ and $\psi_{back}$ are equivalent, in the sense that $1,\tau \models \psi$ iff $1,\tau \models \psi_{back}$, where the evaluation in Def. 4 is extended to variables $V_{pre}$ and $V_{cur}$ as follows: $[\tau,i](v_{curr}) = \tau(i)(v)$ $0{\leq}i{\leq}n$ and $[\tau,i](v_{prev}) = \tau(i{-}1)(v)$ for $0{<}i{\leq}n$. The proof is straightforward and it is omitted.

**Example 8.** *Property $\psi = \mathsf{G}(x'>x) \wedge \mathsf{F}(x=2)$ is equivalent to $\psi_{back} = \mathsf{G}(\mathsf{X}_w(x_{cur}>x_{pre})) \wedge \mathsf{F}(x_{cur}=2)$, it demands that $x$ is increasing, and at some point has value 2.*

Given $\psi_{back}$, we build an NFA $\mathcal{N}_{\psi_{back}}$ using an auxiliary function $\delta$, as in (De Giacomo and Vardi 2013). Let $C$ be the set of constraints in $\psi_{back}$, and $C^{\pm} = C \cup \{neg(c) \mid c \in C\}$; and $\lambda$ be an auxiliary proposition used to mark the last element of a trace. The alphabet of the NFA will be $\Sigma = 2^{C^{\pm}}$, i.e., a symbol is a set of constraints. For the construction we also use its extension $\Sigma_\lambda = 2^{C^{\pm} \cup \{\lambda,\neg\lambda\}}$. Let $\varsigma \in \Sigma_\lambda$ be *satisfiable* if $\{\lambda,\neg\lambda\} \not\subseteq \varsigma$, and the conjunction of constraints in $\varsigma$ is satisfiable. The input of $\delta$ is a (quoted) property $\phi \in \mathcal{L} \cup \{\top,\bot\}$ over $V_{pre} \cup V_{cur}$. The output of $\delta$ is a set of pairs $(\text{“}\phi'\text{”},\varsigma)$ where $\phi' \in \mathcal{L} \cup \{\top,\bot\}$ is again a (quoted) property over $V_{pre} \cup V_{cur}$ and $\varsigma \in \Sigma_\lambda$. For two sets of such pairs $R_1$, $R_2$, let $R_1 \oslash R_2 = \{(\text{“}\psi_1 \wedge \psi_2\text{”}, \varsigma_1 \cup \varsigma_2) \mid (\text{“}\psi_1\text{”},\varsigma_1) \in R_1, (\text{“}\psi_2\text{”},\varsigma_2) \in R_2 \text{ and } \varsigma_1 \cup \varsigma_2 \text{ is satisfiable}\}$, where we simplify $\psi_1 \wedge \psi_2$ if possible; and let $R_1 \otimes R_2$ be defined in the same way, replacing con- with disjunction.

Intuitively, $(\text{“}\phi'\text{”},\varsigma) \in \delta(\phi)$ expresses that when $\phi$ is the property to evaluate, if $\varsigma$ is the current symbol of the word $w \in \Sigma^+$ that is read, then $\phi'$ is the (sub)property yet to satisfy to determine whether $w$ satisfies $\phi$.

**Definition 9.** For $\psi \in \mathcal{L} \cup \{\top,\bot\}$, $\delta$ is as follows:

$$\begin{aligned}
\delta(\text{“}\top\text{”}) &= \{(\text{“}\top\text{”},\emptyset)\} \text{ and } \delta(\text{“}\bot\text{”}) = \{(\text{“}\bot\text{”},\emptyset)\}\\
\delta(\text{“}c\text{”}) &= \{(\text{“}\top\text{”},\{c\}),(\text{“}\bot\text{”},\{neg(c)\})\}\\
\delta(\text{“}\neg c\text{”}) &= \{(\text{“}\bot\text{”},\{c\}),(\text{“}\top\text{”},\{neg(c)\})\}\\
\delta(\text{“}\psi_1 \wedge \psi_2\text{”}) &= \delta(\text{“}\psi_1\text{”}) \oslash \delta(\text{“}\psi_2\text{”})\\
\delta(\text{“}\psi_1 \vee \psi_2\text{”}) &= \delta(\text{“}\psi_1\text{”}) \otimes \delta(\text{“}\psi_2\text{”})\\
\delta(\text{“}\mathsf{X}_s\psi\text{”}) &= \{(\text{“}\psi\text{”},\{\neg\lambda\}),(\text{“}\bot\text{”},\{\lambda\})\}\\
\delta(\text{“}\mathsf{X}_w\psi\text{”}) &= \{(\text{“}\psi\text{”},\{\neg\lambda\}),(\text{“}\top\text{”},\{\lambda\})\}\\
\delta(\text{“}\mathsf{G}\psi\text{”}) &= \delta(\text{“}\psi\text{”}) \oslash \delta(\text{“}\mathsf{X}_w\mathsf{G}\psi\text{”})\\
\delta(\text{“}\psi_1 \,\mathsf{U}\, \psi_2\text{”}) &= \delta(\text{“}\psi_2\text{”}) \otimes (\delta(\text{“}\psi_1\text{”}) \oslash \delta(\text{“}\mathsf{X}_s(\psi_1 \,\mathsf{U}\, \psi_2)\text{”})).
\end{aligned}$$

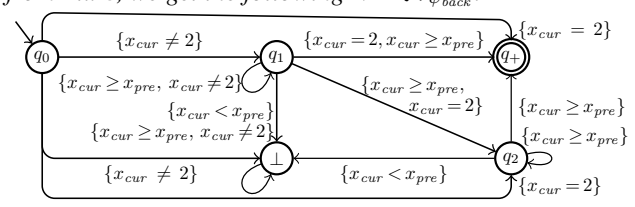
While the symbol $\lambda$ is needed for defining $\delta$, we can omit it from the NFA, and define $\mathcal{N}_\psi$ as follows:

**Definition 10.** Given a property $\psi \in \mathcal{L}$, we define the NFA as $\mathcal{N}_\psi = (Q,\Sigma,\varrho,q_0,Q_F)$ where $q_0 = \text{“}\psi_{back}\text{”}$ is the initial state, $Q_F = \{\text{“}\top\text{”}, q_+\}$ is the subset of final states, $q_-$ is an additional non-final state, and $Q$, $\varrho$ are the smallest sets such that $q_0, q_F, q_+, q_- \in Q$ and whenever $q \in Q \setminus \{q_+,q_-\}$ and $(q',\varsigma) \in \delta(q)$ then $q' \in Q$ and

(i) if $\lambda \notin \varsigma$ then $(q, \varsigma \setminus \{\neg\lambda\}, q') \in \varrho$, and
(ii) whenever $\lambda \in \varsigma$, if $q' = \text{“}\top\text{”}$ then $(q, \varsigma \setminus \{\lambda\}, q_+) \in \varrho$, and if $q' = \text{“}\bot\text{”}$ then $(q, \varsigma \setminus \{\lambda\}, q_-) \in \varrho$.

The following example illustrates the construction.

**Example 11.** *For $\psi_{back} = \mathsf{G}\mathsf{X}_w(x_{cur} \geq x_{pre}) \wedge \mathsf{F}(x_{cur}=2)$ from Ex. 8, we get the following NFA $\mathcal{N}_{\psi_{back}}$:*



A word $w = \varsigma_0, \varsigma_1, \ldots, \varsigma_{n-1} \in \Sigma^+$ is *well-formed* if there are no (negated) constraints in $\varsigma_0$ that mention $V_{pre}$. In Ex. 11 one can see that all words accepted by the NFA are well-formed (this is guaranteed by the translation to $\psi_{back}$). We next define the notion of consistency to relate words $w \in \Sigma^+$ as above (where each $\varsigma_i$ is a set of constraints) with those traces $\tau = \alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ s.t., at each step, the assignment $\alpha_i$ satisfies $\varsigma_i$, i.e., the traces fitting $w$. To that end, for assignments $\alpha$ and $\alpha'$ with domain $V$, define for all $v \in V$ the assignment $[\alpha \circledast \alpha']$ with domain $V_{cur} \cup V_{pre}$ as $[\alpha \circledast \alpha'](v_{pre}) = \alpha(v)$ and $[\alpha \circledast \alpha'](v_{cur}) = \alpha'(v)$.

**Definition 12.** A well-formed word $\varsigma_0, \varsigma_1, \ldots, \varsigma_{n-1} \in \Sigma^+$ is *consistent* with a trace $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ if $\alpha_0 \models \varsigma_0$ and $[\alpha_{i-1} \circledast \alpha_i] \models \varsigma_i$ for all $0 < i < n$.

For instance, the word $\{x_{cur} \neq 2\}\{x_{cur}=2, x_{cur} \geq x_{pre}\}$ is consistent with trace $\{x=1\}\{x=2\}$, but not with $\{x=3\}\{x=2\}$. The key property of $\mathcal{N}_\psi$ is stated in the following theorem: a word $w$ is accepted iff all the traces captured by $w$ satisfy $\psi$.

**Theorem 13.** *Given $\psi \in \mathcal{L}$, a trace $\tau$ and a well-formed word $w \in \Sigma^+$ consistent with $\tau$, $\mathcal{N}_{\psi_{back}}$ accepts $w$ iff $\tau \models \psi$.*

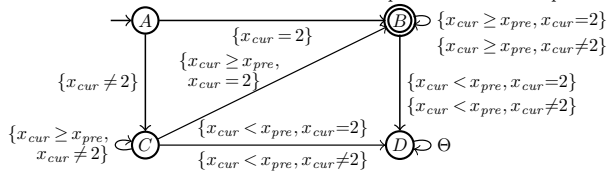
However, $\mathcal{N}_{\psi_{back}}$ is not deterministic, which is inconvenient for monitoring. Thus, we next build an equivalent DFA $\mathcal{D}_{\psi_{back}}$ (exponentially larger in the worst case), using a subset construction, and a restricted alphabet: Let $C_{cur} \subseteq C$ be the subset of constraints in $\psi_{back}$ that mention only $V_{cur}$ (but not $V_{pre}$). Let the alphabet $\Theta \subseteq \Sigma$ consist of all maximal satisfiable subsets of $C^{\pm}$, and $\Theta_{cur} \subseteq \Sigma$ consist of all maximal satisfiable subsets of $C^{\pm}_{cur}$.

**Definition 14.** For $\mathcal{N}_{\psi_{back}} = (Q, \Sigma, \varrho, q_0, Q_F)$, let $\mathcal{D}_{\psi_{back}} = (\mathcal{Q}, \Theta \cup \Theta_{cur}, \Delta, \{q_0\}, \mathcal{Q}_F)$ where $\mathcal{Q} = 2^Q$, $\mathcal{Q}_F$ consists of all $P \subseteq Q$ such that $P \cap Q_F \neq \emptyset$, and the transition function is given by $\Delta(\{q_0\}, \varsigma_0) = \{q' \mid (q_0, \varsigma', q') \in \varrho \text{ and } \varsigma' \subseteq \varsigma_0\}$ for all $\varsigma_0 \in \Theta_{cur}$, and $\Delta(P, \varsigma) = \{q' \mid q \in P, (q, \varsigma', q') \in \varrho \text{ and } \varsigma' \subseteq \varsigma\}$ for all $\varsigma \in \Theta$ and $P \neq \{q_0\}$.

Note that $\Delta$ is defined either $(i)$ for the initial state $\{q_0\}$ and $\varsigma \in \Theta_{cur}$, i.e., a constraint set not mentioning $V_{pre}$, or $(ii)$ for a non-initial state and $\varsigma \in \Theta$. This distinction ensures that $\mathcal{D}_{\psi_{back}}$ accepts only well-formed words, and $\mathcal{D}_\psi$ is *deterministic* in that for every $w \in \Theta_{cur}\Theta^*$ there is a unique state $P$ of $\mathcal{D}_{\psi_{back}}$ such that $\{q_0\} \to^*_w P$. $\mathcal{D}_\psi$ is equivalent to $\mathcal{N}_\psi$ in the following sense:

**Lemma 15.** Given a trace $\tau$, $\mathcal{D}_\psi$ accepts a word consistent with $\tau$ iff $\mathcal{N}_\psi$ accepts a word consistent with $\tau$.

**Example 16.** *We build the DFA for the property $\psi_{back}$ and the respective NFA from Ex. 11. Below, $\Theta_{cur}$ consists of $\{x_{cur} = 2\}$ and $\{x_{cur} \neq 2\}$, and $\Theta$ of all four combinations of $x_{cur} = 2$ or $x_{cur} \neq 2$ with $x_{cur} \geq x_{pre}$ or $x_{cur} < x_{pre}$:*

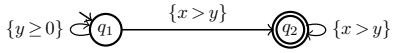



*Here $A$ corresponds to $\{q_0\}$, $B$ to $\{q_2, q_+\}$ (equivalent to $\{q_2, q_+, \bot\}$), $C$ to $\{q_1, \bot\}$, and $D$ to $\{\bot\}$.*

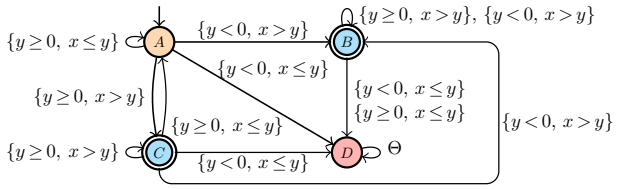
## Monitoring properties without lookahead

In this section we show that monitoring properties without lookahead is solvable, and a monitoring structure is given by the DFA $\mathcal{D}_{\psi_{back}} = (\mathcal{Q}, \Theta, \delta, \{q_0\}, \mathcal{Q}_F)$ (note that without lookahead, $\Theta_{cur}$ and $\Theta$ as defined in the last section coincide). We illustrate the construction on an example.

**Example 17.** *Let $\psi = (y \geq 0)\ \mathsf{U}\ (x > y \wedge \mathsf{G}(x > y))$. Then $\psi_{back}$ is as $\psi$ but where $x$ is replaced by $x_{cur}$ and $y$ by $y_{cur}$. As there is no lookahead, no confusion can arise, so we write $x$ for $x_{cur}$ and $y$ for $y_{cur}$. We get the following NFA $\mathcal{N}_{\psi_{back}}$ with alphabet $\Sigma = 2^{C^{\pm}}$ for $C = \{x > y,\ y \geq 0\}$:*




*The next DFA $\mathcal{D}_{\psi_{back}}$ is equivalent; its alphabet consists of all maximal satisfiable subsets of $C^{\pm}$, i.e., $\{y \geq 0,\ x > y\}$, $\{y \geq 0,\ x \leq y\}$, $\{y < 0,\ x > y\}$, and $\{y < 0,\ x \leq y\}$.*




*Here $A = \{q_1\}$, $B = \{q_2\}$, $C = \{q_1, q_2\}$, and $D = \emptyset$.*

Next, we formalize how to use $\mathcal{D}_{\psi_{back}}$ as a monitor. We call a state $P'$ *reachable* from $P$ if $P \to^*_w P'$ for some $w$.

**Theorem 18.** *If a word $w \in \Theta^+$ is consistent with a trace $\tau$ and $P$ is the state in $\mathcal{D}_{\psi_{back}}$ such that $\{q_0\} \to^*_w P$,*
*$\tau \models [\![\psi = \text{PS}]\!]$ if only final states are reachable from $P$,*
*$\tau \models [\![\psi = \text{CS}]\!]$ if $P \in \mathcal{Q}_F$ but $P \to^* P'$ for some $P' \notin \mathcal{Q}_F$,*
*$\tau \models [\![\psi = \text{CV}]\!]$ if $P \notin \mathcal{Q}_F$ but $P \to^* P'$ for some $P' \in \mathcal{Q}_F$,*
*$\tau \models [\![\psi = \text{PV}]\!]$ if no final state is reachable from $P$.*

*Proof.* By Lems. 15 and 13, $\tau \models \psi$ iff $P$ is final. (Case PS:) $P$ has no path to a non-final state. Towards contradiction, assume $\tau'$ exists s.t. $\tau\tau' \not\models \psi$. By Lems. 13 and 15, $\mathcal{D}_{\psi_{back}}$ does not accept a word consistent with $\tau\tau'$. First, we show that for every non-empty trace $\tau$ there is a unique word $w \in \Theta_{cur}\Theta^*$ consistent with $\tau$. Let $\tau = \langle \alpha_0, \ldots, \alpha_{n-1} \rangle$, and $C$ be the constraints in $\psi_{back}$. Since for every constraint $c \in C_{cur}$, either $\alpha \models c$ or $\alpha \models neg(c)$, and $\Theta_{cur}$ contains all maximal satisfiable subsets of $C_{cur} \cup C^-_{cur}$, there must be exactly one subset $\varsigma_0 \in \Theta_{cur}$ s.t. $\alpha \models \bigwedge \varsigma_0$. Similarly, $[\alpha_i \circledast \alpha_{i+1}]$ must model either $c$ or $neg(c)$ for all constraints in $c$, so there must be exactly one $\varsigma_i \in \Theta$ s.t. $[\alpha_i \circledast \alpha_{i+1}] \models \bigwedge \varsigma_i$, for all $0 < i \leq n-1$. Thus, the word $w = \langle \varsigma_0, \ldots, \varsigma_{n-1} \rangle$ is unique and consistent with $\tau$.

With this result at hand, by determinism, we can write $\bar{w} = ww'$ s.t. $\{q_0\} \to^*_w P \to^*_{w'} P'$, contradicting that only final states are reached from $P$. (Case CS:) let $P \to^*_{w'} P'$ for a non-final state $P'$. Sets in $\Theta$ are satisfiable, so there is a trace $\tau'$ s.t. $w'$ is consistent with $\tau'$, and $ww'$ with $\tau\tau'$. By the result above, $ww'$ is the unique word in $\Theta^*$ consistent with $\tau\tau'$, and not accepted. By Lems. 15 and 13, $\tau\tau' \not\models \psi$, so $\tau \models [\![\psi = \text{CS}]\!]$. The other cases are dual. □

Thm. 18 implies that monitoring is solvable, and the monitoring state only depends on the DFA state matching a trace.

**Example 19.** *Consider the property of Ex. 17. The following example trace is interleaved with the state reached by the corresponding word in $\mathcal{D}_{\psi_{back}}$, and the monitoring state:*

$$1 \begin{Bmatrix} x{=}0 \\ y{=}0 \end{Bmatrix} \underset{\text{CV}}{1} \begin{Bmatrix} x{=}0 \\ y{=}3 \end{Bmatrix} \underset{\text{CV}}{1} \begin{Bmatrix} x{=}4 \\ y{=}3 \end{Bmatrix} \underset{\text{CS}}{12} \begin{Bmatrix} x{=}0 \\ y{=}3 \end{Bmatrix} \underset{\text{CV}}{1} \begin{Bmatrix} x{=}0 \\ y{=}{-}1 \end{Bmatrix} \underset{\text{CS}}{2}$$

*Each DFA state corresponds to a monitoring state, as indicated by the colors in Ex. 17: non-final states are* CV *if a final state is reachable (yellow),* PV *otherwise (red); final states are* CS *if a non-final state is reachable (blue),* PS *otherwise.*

## Solvable cases for properties with lookahead

In this section we study the more involved case of properties with lookahead 1. The next example shows that the approach from the last section does not work for this case:

**Example 20.** *Consider the property and DFA from Exa. 16 and the following trace, interleaved with the reached state in $\mathcal{D}_{\psi_{back}}$ as well as the monitoring state for every prefix:*

$$A \underset{}{} \{x=0\} \underset{\text{CV}}{C} \{x=1\} \underset{\text{CV}}{C} \{x=3\} \underset{\text{PV}}{C} \{x=4\} \underset{\text{PV}}{C}$$

*The reached DFA state is the same for every prefix, even though the monitoring state changes.*

Note that while the state $P$ reached by a trace $\tau$ in $\mathcal{D}_{\psi_{back}}$ can still tell whether $\tau$ satisfies $\psi$, reachability of (non)final states from $P$ is no longer sufficient to predict whether the trace can be *extended* to satisfy $\psi$. We therefore develop an enhanced monitoring structure. To simplify notation, we assume that $\psi$ is the result of the $back(\cdot)$ transformation, i.e., a property over variables $V_{pre} \cup V_{cur}$. Moreover, we assume throughout this section that $\mathcal{D}_\psi$ is a DFA for $\psi$ of the form $\mathcal{D}_\psi = (\mathcal{Q}, \Theta \cup \Theta_{cur}, \Delta, \{q_0\}, \mathcal{Q}_F)$.

The idea is to abstract the variable configurations that are relevant for verification by a set of formulas. This abstraction requires some additional notation: First, let $V_0 = \{v_0 \mid v \in V\}$ be a copy of $V$, it will act as placeholders for the initial values of the variables $V$. We write $\varphi_{init}$ for the formula $\varphi_{init} = \bigwedge_{v \in V} v = v_0$. For a formula $\varphi$ with free variables $V_0 \cup V$, let $\varphi(\overline{U})$ denote the formula where $\overline{V}$ is replaced by $\overline{U}$, but $V_0$ is not replaced (with denote by $\overline{V}$ a fixed ordering of $V$ as a vector). For a set of constraints $C$ with free variables $V_{pre} \cup V_{cur}$, let $C(\overline{U}, \overline{V})$ denote the constraints where $\overline{V}_{pre}$ is replaced by $\overline{U}$ and $\overline{V}_{cur}$ is replaced by $\overline{V}$.

**Definition 21.** For a formula $\varphi$ with free variables $V_0 \cup V$ and a set of constraints $C$ with free variables $V_{pre} \cup V_{cur}$, let $update(\varphi, C) = \exists \overline{U}.\varphi(\overline{U}) \wedge \bigwedge C(\overline{U}, \overline{V})$.

Both $\varphi$ and $update(\varphi, C)$ have free variables $V_0 \cup V$. Although $update$ introduces existential quantifiers, there is an equivalent quantifier-free formula for $update(\varphi, C)$: linear arithmetic has quantifier elimination. E.g., for $\varphi = (x = x_0 \wedge x_0 \neq 2)$ and $C = \{x_{cur} \geq x_{pre}, x_{cur} = 2\}$, we have $update(\varphi, C) = \exists u.\, u = x_0 \wedge x_0 \neq 2 \wedge x \geq u \wedge x = 2$, which is equivalent to $x_0 < 2 \wedge x = 2$.

In order to design a monitoring structure, we build a *constraint graph* which, intuitively, connects in a graph all variable dependencies (described by formulas) emerging from a state $P_0$ in the DFA, and where $V_0$ acts as placeholder for the initial variable values. The constraint graph is parameterized by an equivalence relation $\sim$ on formulas used to reduce its size. Our correctness proof will require further properties of $\sim$ but a default choice is the logical equivalence relation $\equiv$.

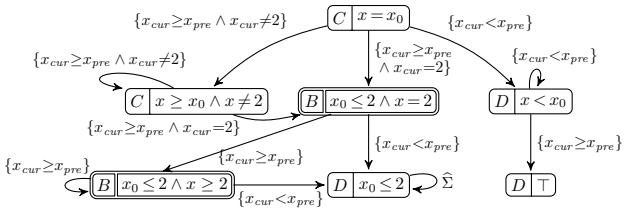



Figure 1: Constraint graph $\text{CG}_\psi(C)$.

**Definition 22.** A *constraint graph* $\text{CG}_\psi(P_0, \sim)$ for $\mathcal{D}_\psi$ and a DFA state $P_0 \in \mathcal{Q}$ is a triple $\langle S, s_0, \gamma, S_F \rangle$ where the node set $S$ consists of tuples $(P, \varphi)$ of a state $P \in \mathcal{Q}$ and a formula $\varphi$ with free variables $V_0 \cup V$, and $\gamma \subseteq S \times \Sigma \times S$. Then, $S$ and $\gamma$ are inductively defined as follows:

$(i)$ $s_0 = (P_0, \varphi_{init})$ is the initial node and $s_0 \in S$;

$(ii)$ if $s = (P, \varphi) \in S$ and $P \xrightarrow{\varsigma} P'$ in $\mathcal{D}_\psi$ such that $update(\varphi, \varsigma)$ is satisfiable, there is some $s' = (P', \varphi') \in S$ with $\varphi' \sim update(\varphi, \varsigma)$, and $s \xrightarrow{\varsigma} s'$ is in $\gamma$;

$(iii)$ $(P, \varphi) \in S$ is in the set of final nodes $S_F$ iff $P \in \mathcal{Q}_F$.

We simply write $\text{CG}_\psi(P_0)$ for $\text{CG}_\psi(P_0, \equiv)$. The constraint graph $\text{CG}_\psi(C)$ for the DFA $\mathcal{D}_{\psi_{back}}$ from Ex. 16 is shown in Fig. 1. For readability, edge labels are combined; e.g. the two edges from $B$ to $D$ labeled $\{x_{cur} < x_{pre}, x_{cur} = 2\}$ and $\{x_{cur} < x_{pre}, x_{cur} \neq 2\}$ are combined to one edge labeled $\{x_{cur} < x_{pre}\}$. The construction in Def. 22 need not terminate as infinitely many formulas may occur, but we will show in the next section that termination is guaranteed for several relevant cases.

**Definition 23.** For $\psi$ with DFA $\mathcal{D}_\psi$, a state $P_0$ in $\mathcal{D}_\psi$, and $G := \text{CG}_\psi(P_0, \sim)$ with node set $S$, let

$$\text{FSAT}(G) = \exists \overline{V}. \bigvee \{\varphi \mid (P, \varphi) \in S \text{ is a final node}\}$$
$$\text{FUNS}(G) = \exists \overline{V}. \bigvee \{\varphi \mid (P, \varphi) \in S \text{ is a non-final node}\}$$

Intuitively, $\text{FSAT}(G)$ expresses a condition on the variable values when a final state in $\mathcal{D}_\psi$ is reachable from $P_0$; dually, FUNS gives a condition for reachability of non-final states. This leads to the following monitoring procedure:

```
1: procedure MONITOR(ψ, τ)
2:     compute ψ_back and D_{ψ_back}          ▷ or take from cache
3:     w ← word in Θ_cur Θ* consistent with τ
4:     P ← state in D_{ψ_back} such that {q0} →*_w P
5:     G ← CG_ψ(P, ∼)                         ▷ or take from cache
6:     α_n ← last assignment in τ
7:     if P accepting in D_{ψ_back} then
8:         return (CS if α_n ⊨ FUNS(G) else PS)
9:     else return (CV if α_n ⊨ FSAT(G) else PV)
```

Note that the monitoring structures (DFA, constraint graphs for DFA states, FSAT, FUNS) can be computed once and for all upfront. This is particularly useful when monitoring multiple traces against the same property $\psi$. In this case, the procedure above should be implemented in a *lazy* way, where the structures are computed when needed and cached for later use. At monitoring time, given trace $\tau$, one thus only

needs to find the DFA state corresponding to $\tau$, and evaluate the pre-computed formula with the last assignment of $\tau$.

**Example 24.** *For $\psi_{back} = \mathsf{GX}_w(x_{cur} \geq x_{pre}) \wedge \mathsf{F}(x_{cur} = 2)$ from Exa. 16, by the graph in Fig. 1 we have $\text{FSAT}(C) = \exists x.(x_0 \leq 2 \wedge x \geq 2) \vee (x_0 \leq 2 \wedge x = 2)$, equivalent to $x_0 \leq 2$. We show how the procedure distinguishes two prefixes of the trace from Exa. 20 with different monitoring state:*

- *Trace $\tau_1 = \langle \{x = 0\} \rangle$ corresponds to $w = \langle \{x_{cur} \neq 2\} \rangle$, leading to the non-accepting state $C$. As $\{x = 0\} \models \text{FSAT}(G_C)(V) = x \leq 2$, we have $\tau_1 \models [\![\psi{=}\text{CV}]\!]$.*
- *Trace $\tau_3 = \langle \{x = 0\}, \{x = 1\}, \{x = 3\} \rangle$ matches $\langle \{x_{cur} \neq 2\}, \{x_{cur} \neq 2, x_{cur} \geq x_{pre}\}^2 \rangle$, leading again to state $C$. However, $\{x = 3\} \not\models x \leq 2$, so $\tau_3 \models [\![\psi{=}\text{PV}]\!]$.*

**Correctness.** To prove correctness, we need some additional notions. First, we define *history constraints* to capture the formulas obtained from stacked *update* operations:

**Definition 25.** For $w = \varsigma_0, \ldots, \varsigma_{n-1}$ in $\Theta^*$, the *history constraint* $h(w)$ is given by $h(w) = \varphi_{init}$ if $n = 0$, and if $n > 0$ then $h(w) = \mathit{update}(h(\langle \varsigma_0, \ldots, \varsigma_{n-2} \rangle), \varsigma_{n-1})$.

There is the correspondence between satisfying assignments of $h(w)$ and traces that satisfy the constraints in $w$:

**Lemma 26.** For $w \in \Theta^*$ of length $n$, $h(w)$ is satisfied by assignment $\nu$ iff $\langle \emptyset \rangle \cdot w$ is consistent with a trace $\alpha_0, \ldots, \alpha_n$ such that $\nu(v_0) = \alpha_0(v)$ and $\nu(v) = \alpha_n(v)$ for all $v \in V$.

Next, we define a *history set* to capture all variable configurations that are relevant for monitoring. More precisely, it collects pairs of history constraints for words and the state that they lead to, but where equivalent pairs are eliminated:

**Definition 27.** A *history set* for $\mathcal{D}_\psi$ is a minimal set $\Phi$ of pairs $(P, \varphi)$ of a state $P \in \mathcal{Q}$ and a formula $\varphi$ such that for all $P'$ and $w \in \Theta_{cur}\Theta^*$ with $P \to_w^* P'$ in $\mathcal{D}_\psi$, there is a $(P', \varphi) \in \Phi$ s.t. $h(w) \equiv \varphi$.

If $\Phi$ is finite, it represents an abstract representation of the relevant variable configurations and can be used to build the states of the constraint graph. For succinctness, we pair a history set with an equivalence relation.

**Definition 28.** A *summary* for $\mathcal{D}_\psi$ is a pair $(\Phi, \sim)$ of a history set $\Phi$ for $\mathcal{D}_\psi$ and equivalence relation $\sim$ on $\Phi$ such that
(1) $\sim$ contains $\equiv$ on formulas in $\Phi$ and is decidable,
(2) for all $(q, \varphi), (q, \psi) \in \Phi$ such that $\varphi \sim \psi$, (a) if $\alpha \models \varphi$, there is some $\alpha'$ such that $\alpha' \models \psi$ and $\alpha(u) = \alpha'(u)$ for all $u \in V_0$, and (b) for all transitions $q \xrightarrow{\varsigma} q'$, $\mathit{update}(\varphi, \varsigma) \sim \mathit{update}(\psi, \varsigma)$.

The summary $(\Phi, \sim)$ is *finite* if $\sim$ has finitely many equivalence classes, and $\psi$ *has finite summary* if a finite summary exists for $\mathcal{D}_\psi$. For a word $w = \varsigma_1, \ldots, \varsigma_n$, we write $(P_0, \varphi_{init}) \to_w^* (P, \varphi)$ if there is a path $(P_0, \varphi_{init}) \xrightarrow{\varsigma_1} \cdots\cdots \xrightarrow{\varsigma_n} (P, \varphi)$ in $\text{CG}_\psi(P_0, \sim)$. The next lemma states that a path in the CG emulates a path in $\mathcal{D}_\psi$ and leads to a state that is equivalent to a history constraint.

**Lemma 29.** Let $(\Phi, \sim)$ be a summary for $\mathcal{D}_\psi$. (a) If $\text{CG}_\psi(P_0, \sim)$ has a path $(P_0, \varphi_{init}) \to_w^* (P, \varphi)$ then $P_0 \to_w^* P$ in $\mathcal{D}_\psi$ and $\varphi \sim h(w)$ is satisfiable, and (b) if $P_0 \to_w^* P$ and $h(w)$ is satisfiable then there is a path $(P_0, \varphi_{init}) \to_w^* (P, \varphi)$ for some $\varphi$ such that $\varphi \sim h(w)$.

At this point we are ready to prove our main theorem:

**Theorem 30.** *Let $(\Phi, \sim)$ be a summary for $\mathcal{D}_\psi$. Given a DFA $\mathcal{D}_\psi$ for $\psi$ and a trace $\tau$, let $w \in \Theta_{cur}\Theta^*$ be the word consistent with $\tau$ and $P$ the $\mathcal{D}_\psi$ state s.t. $\{q_0\} \to_w^* P$. For $G := \text{CG}_\psi(P, \sim)$ the constraint graph from $P$,*

- *if $P \in P_F$ then $\tau \models [\![\psi{=}\text{CS}]\!]$ if $\alpha_n \models \text{FUNS}(G)(\overline{V})$, and $\tau \models [\![\psi{=}\text{PS}]\!]$ otherwise,*
- *if $P \notin P_F$ then $\tau \models [\![\psi{=}\text{CV}]\!]$ if $\alpha_n \models \text{FSAT}(G)(\overline{V})$, and $\tau \models [\![\psi{=}\text{PV}]\!]$ otherwise.*

*Proof.* Note that the word $w$ that is consistent with $\tau$ is unique (cf. proof of Thm. 18). Consider the case where $P \in P_F$, so that $\tau \models \psi$ by Lem. 13 and 15. First, suppose $\alpha_n \models \text{FUNS}(G)(\overline{V})$, so $\alpha_n \models (\exists \overline{V}.\, \varphi)(\overline{V})$ for some non-final node $(P', \varphi)$ of $G$. Let $u$ be a word s.t. there is a path $(P, \varphi_{init}) \to_u^* (P', \varphi)$ in $\text{CG}_\psi(P)$. By Lem. 29, $P \to_u^* P'$ in $\mathcal{D}_\psi$ and $\varphi \sim h(u)$. By Def. 28, $\alpha_n \models (\exists \overline{V}.\, h(u))(\overline{V})$, so there is an assignment $\nu$ with domain $V_0 \cup V$ s.t. $\nu(v_0) = \alpha_n(v)$ for all $v \in V$ and $\nu \models h(u)$. By Lem. 26, $\langle \emptyset \rangle \cdot u$ is consistent with some trace $\tau' = \langle \alpha'_0, \ldots, \alpha'_k \rangle$ s.t. $\nu(v_0) = \alpha'_0(v)$ for all $v \in V$. Thus, $\alpha_n$ and $\alpha'_0$ coincide, and $wu$ is consistent with the trace $\tau'' = \langle \alpha_0, \ldots, \alpha_n, \alpha'_1, \ldots, \alpha'_k \rangle$ as $w$ is consistent with $\tau$ and $u$ with $\tau'$. As $\{q_0\} \to_{wu}^* P'$ is not accepting, by Lem. 13, $\tau'' \not\models \psi$, hence $\tau \models [\![\psi{=}\text{CS}]\!]$.

Second, let $\alpha_n \not\models \text{FUNS}(G)(\overline{V})$. Towards a contradiction, suppose there is a trace $\tau' = \langle \alpha'_1, \ldots, \alpha'_k \rangle$ s.t. $\tau\tau' \not\models \psi$. Again, there is a unique word $u \in \Theta^*$ s.t. $wu$ is consistent with $\tau\tau'$; let $P'$ the state s.t. $\{q_0\} \to_{wu}^* P'$. By Lem. 13 and 15, the state $P'$ is not final. As $wu$ is consistent with $\tau\tau'$, $\langle \emptyset \rangle \cdot u$ is consistent with $\langle \alpha_n \rangle \tau'$, so by Lem. 26, the assignment $\nu$ s.t. $\nu(v_0) = \alpha_n(v)$ and $\nu(v) = \alpha'_k(v)$ for all $v \in V$ satisfies $h(u)$. Thus Lem. 29 implies that $\text{CG}_\psi(P)$ has a path $(P, \varphi_{init}) \to_u^* (P', \varphi)$ s.t. $\varphi \sim h(u)$. As $\nu \models h(u)$, by Def. 28, $\alpha_n \models (\exists .\, \varphi)(\overline{V})$ because $\nu(v_0) = \alpha_n(v)$ for all $v \in V$. As $P'$ is non-final, it follows that $\alpha_n \models \text{FUNS}(G)$, a contradiction; so $\tau'$ cannot exist. Thus $\tau \models [\![\psi{=}\text{PS}]\!]$. The case where $\tau \not\models \psi$ is dual, using $\text{FSAT}(G)$. □

Finally, if $\psi$ has a finite summary $(\Phi, \sim)$, by Lem. 29 a constraint graph $\text{CG}(P, \sim)$ computed from $\mathcal{D}_\psi$ can take all nodes from the finitely many equivalence classes of $\Phi$, so that the constraint graph can be finite. Therefore:

**Corollary 31.** Monitoring is solvable for $\psi \in \mathcal{L}$ if $\psi$ has finite summary.

## Concrete criteria for solvability

We next apply Cor. 31 to show solvability of concrete property classes, by restricting the constraint set or control flow.

**Monotonicity constraints** (MCs) over variables $\mathcal{V}$ and domain $D$ have the form $p \odot q$ where $p, q \in D \cup \mathcal{V}$ and $\odot$ is one of $=, \neq, \leq$, or $<$. We call an $\text{LTL}_f$ property whose constraint atoms are MCs over $D$ an *$MC_D$ property*. Ex. 11 gives a simple example of an MC property. However, MC properties are rich enough to capture practically important settings: For instance, Geist, Rozier, and Schumann (2014) model the specification of a fluxgate magnetometer of a

Swift UAS system operated by NASA, as linear-time properties with arithmetic comparisons of sensor values, which are all MCs over $\mathbb{Q}$ (cf. their Table 2). We have that:

**Theorem 32.** *Monitoring is solvable for MC$_\mathbb{Q}$ properties.*

*Proof (sketch).* This is shown as in (Felli, Montali, and Winkler 2022b, Thm. 5.2), exploiting that quantifier elimination of MC formulas over $\mathbb{Q}$ produces MC formulas with the same constants, so that a finite history set exists. □

**Integer periodicity constraints** (IPCs) confine the constraint language in a similar way as MCs, but allow equality modulo, and variable-to-variable comparisons are restricted. IPCs are e.g. used in calendar formalisms (Demri 2006). More precisely, IPC atoms have the form $x = y$, $x \odot d$ for $\odot \in \{=, \neq, <, >\}$, $x \equiv_k y + d$, or $x \equiv_k d$, for variables $x, y$ with domain $\mathbb{Z}$ and $k, d \in \mathbb{N}$. An $\mathcal{L}$ property over IPC atoms is called an *IPC property*. The next result is proven similarly as Thm. 36, cf. (Felli, Montali, and Winkler 2022a, Thm. 4).

**Theorem 33.** *Monitoring is solvable for IPC properties.*

A simple example of an IPC property is $(x \equiv_7 y + 1)\ \mathsf{U}\ (x = z)$. Also e.g. a parallel program as in (Havelund, Reger, and Rosu 2019, Fig. 3) can be modeled as an IPC property, where monitoring detect at runtime whether a trace may trigger the race on variable $x$ that the program exhibits.

**Control flow restrictions** can be used instead of confining the constraint language to obtain solvability. *Computation graphs* (Damaggio, Deutsch, and Vianu 2012) were introduced to track dependencies between variable instances at different trace events. A property $\psi$ is said to have *$k$-bounded lookback* (Felli, Montali, and Winkler 2022b) if all paths in computation graphs of $\psi$ have length at most $k$, without counting equality edges. Intuitively, this corresponds to forbid variable updates that depend on an unbounded history of values (of the same or of other variables), but only on a $k$-bounded "moving window". A property has bounded lookback (BL) if its has $k$-bounded lookback for some $k \geq 0$. E.g., $\mathsf{G}(x' > x)$ does not have BL, because there is an unbounded dependency chain between the values of $x$, whereas $\mathsf{F}(x' > 2y) \wedge \mathsf{G}(x + y > 0)$ has 1-bounded lookback as variable comparisons span at most one time unit. Also all properties with lookahead 0 have BL. Monitoring of BL properties is solvable because, roughly, the quantifier depth of history constraints is upper-bounded a priori. A formal definition, proof and examples are in (Felli et al. 2022).

**Theorem 34.** *Monitoring of BL properties is solvable.*

**Gap-order constraints** (GCs) have the form $x - y \geq k$ for $x$ and $y$ either variables with domain $\mathbb{Z}$ or integers, and $k \in \mathbb{N}$. Satisfiability of GC properties is decidable (Bozzelli and Pinchinat 2014). However, validity is undecidable, because a GC property $\psi$ is valid iff $\neg\psi$ is unsatisfiable, where $\neg\psi$ can be written as a property whose atoms are negated GCs, i.e. of the form $\neg c$ for $c$ a GC. Negated GCs can model counters (Bozzelli and Pinchinat 2014, Thm. 12), so that state reachability of a 2-counter machine can be reduced to satisfiability of a property over negated GCs. Consequently,

**Remark 35.** *Monitoring of GC properties is not solvable.*

MCs over $\mathbb{Z}$ can be written as GCs (Bozzelli and Pinchinat 2014), with the key difference that for an MC $c$ also $\neg c$ is an MC, and hence a GC. Thus, we can assume that in a DFA for an MC$_\mathbb{Z}$ property $\psi$, all constraints are GCs.

In (Felli, Montali, and Winkler 2022b, Thm. 5.5) it was shown that a transition system over GCs admits a finite summary $(\mathrm{GC}_K, \sim_K)$. Here, $\mathrm{GC}_K$ is the finite set of quantifier-free formulas with GC atoms over variables $V$ and constants $\leq K$, and $\sim_K$ is the cutoff equivalence relation, where equivalence of two formulas is checked after replacing constants greater than $K$ by $K$, for $K$ the maximal difference between constants in the input. Finite summary relies on the initial assignment of $V$ being fixed, whereas by Def. 25 history constraints start from $\varphi_{init}$, i.e., a parametric assignment $\overline{V} \mapsto \overline{V}_0$. This can hamper finiteness of the summary. E.g., for $\psi = \mathsf{X}_w(x_{cur} > x_{pre}\ \mathsf{U}\ x_{cur} > 5)$ there are finitely many equivalence classes only if the initial value $z$ of $x$ is known (after at most $5 - z$ steps $\psi$ holds). Nonetheless, Cor. 31 can be used to show that monitoring is solvable, although the procedure needs to be suitably refined.

**Theorem 36.** *Monitoring is solvable for MC$_\mathbb{Z}$ properties.*

*Proof (sketch).* We modify our procedure as follows: (1) in line 5, we compute the CG starting from $\bigwedge v{=}\alpha_n(v)$ instead of $\varphi_{init}$. By (Felli, Montali, and Winkler 2022b, Thm. 5.5) a finite summary exists, so the CG computation terminates. (2) The checks in lines 8,9 are simplified: as the initial assignment is now integrated in the CG, at line 8 it suffices to check whether the CG has a non-empty path to a non-final state (resp. to a final state in line 9). It is not hard to adapt the proof of Thm. 30 accordingly. However, the changed procedure comes with the price of computing CGs not only for every DFA state but also every assignment in the trace. □

## Conclusions

**Implementation.** We implemented our approach in a prototype, whose source code and web interface are available via *https://bit.ly/3QFoJHA*. The tool takes an ALTL$_f$ property $\psi$ and a trace as input, determines whether $\psi$ is in one of the decidable classes identified in the last section, constructs and visualizes a monitor, and computes the monitoring state. The tool is implemented in Python, using Z3 (de Moura and Bjørner 2008) and CVC5 (Deters et al. 2014) for SMT checks and quantifier elimination.

**Future work.** We want to lift our automata-based approach to the case of traces of richer states, equipped with full-fledged relations. Hence we plan to study how to integrate our approach with (Decker, Leucker, and Thoma 2016) and (Calvanese et al. 2022). The former considers states consisting of first-order structures modulo theories, but does not foresee any form of lookahead in the properties. In (Calvanese et al. 2022), instead, states are labeled by first-order interpretations and properties are expressed in a fragment of first-order LTL with a controlled first-order quantification across time; however arithmetic theories are not supported.

**Acknowledgments** This work was partially funded by the Unibz project SMART-APP and the PRIN MIUR project PINPOINT.

## References


Bartocci, E.; Falcone, Y.; Francalanza, A.; and Reger, G. 2018. Introduction to Runtime Verification. In Bartocci, E.; and Falcone, Y., eds., *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *LNCS*, 1–33. Springer.

Basin, D. A.; Klaedtke, F.; Müller, S.; and Zalinescu, E. 2015. Monitoring Metric First-Order Temporal Properties. *J. ACM*, 62(2): 15:1–15:45.

Bauer, A.; Leucker, M.; and Schallhart, C. 2010. Comparing LTL Semantics for Runtime Verification. *J. Logic and Comput.*, 20(3): 651–674.

Bozzelli, L.; and Pinchinat, S. 2014. Verification of gap-order constraint abstractions of counter systems. *Theor. Comput. Sci.*, 523: 1–36.

Calvanese, D.; De Giacomo, G.; Montali, M.; and Patrizi, F. 2022. Verification and Monitoring for First-Order LTL with Persistence-Preserving Quantification over Finite and Infinite Traces. In *Proc. 31st IJCAI*, 2553–2560.

Damaggio, E.; Deutsch, A.; and Vianu, V. 2012. Artifact systems with data dependencies and arithmetic. *ACM Trans. Database Syst.*, 37(3): 22:1–22:36.

D'Angelo, B.; Sankaranarayanan, S.; Sánchez, C.; Robinson, W.; Finkbeiner, B.; Sipma, H. B.; Mehrotra, S.; and Manna, Z. 2005. LOLA: Runtime Monitoring of Synchronous Systems. In *Proc. 12th TIME*, 166–174.

Dastani, M.; Torroni, P.; and Yorke-Smith, N. 2018. Monitoring norms: a multi-disciplinary perspective. *Knowl. Eng. Rev.*, 33.

De Giacomo, G.; De Masellis, R.; Maggi, F. M.; and Montali, M. 2022. Monitoring Constraints and Metaconstraints with Temporal Logics on Finite Traces. *ACM Trans. Softw. Eng. Methodol.*, 31(4).

De Giacomo, G.; and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In Rossi, F., ed., *Proc. 23rd IJCAI*, 854–860. IJCAI/AAAI.

de Moura, L.; and Bjørner, N. 2008. Z3: An Efficient SMT Solver. In *Proc. 14th TACAS*, volume 4963 of *LNCS*, 337–340.

Decker, N.; Leucker, M.; and Thoma, D. 2016. Monitoring modulo theories. *Int. J. Softw. Tools Technol. Transf.*, 18(2): 205–225.

Demri, S. 2006. LTL over integer periodicity constraints. *Theor. Comput. Sci.*, 360(1-3): 96–123.

Demri, S.; and D'Souza, D. 2007. An automata-theoretic approach to constraint LTL. *Inform. Comput.*, 205(3): 380–415.

Deters, M.; Reynolds, A.; King, T.; Barrett, C. W.; and Tinelli, C. 2014. A tour of CVC4: How it works, and how to use it. In *Proc. 14th FMCAD*, 7.

Falcone, Y.; Krstic, S.; Reger, G.; and Traytel, D. 2021. A taxonomy for classifying runtime verification tools. *Int. J. Softw. Tools Technol. Transf.*, 23(2): 255–284.

Faymonville, P.; Finkbeiner, B.; Schledjewski, M.; Schwenger, M.; Stenger, M.; Tentrup, L.; and Torfah, H. 2019. StreamLAB: Stream-based Monitoring of Cyber-Physical Systems. In *Proc. 31st CAV*, volume 11561 of *LNCS*, 421–431.

Felli, P.; Montali, M.; Patrizi, F.; and Winkler, S. 2022. Monitoring Arithmetic Temporal Properties on Finite Traces (Extended Version). Available from https://arxiv.org/abs/2211.17166.

Felli, P.; Montali, M.; and Winkler, S. 2022a. CTL* model checking for data-aware dynamic systems with arithmetic. In *Proc. 11th IJCAR*, volume 13385, 36–56.

Felli, P.; Montali, M.; and Winkler, S. 2022b. Linear-Time Verification of Data-Aware Dynamic Systems with Arithmetic. In *Proc. 36th AAAI*, 5642–5650. AAAI Press. Extended version available from https://arxiv.org/abs/2203.07982.

Geatti, L.; Gianola, A.; and Gigante, N. 2022. Linear Temporal Logic Modulo Theories Over Finite Traces. In *Proc. 31st IJCAI*, 2641–2647.

Geist, J.; Rozier, K. Y.; and Schumann, J. 2014. Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems. In *Proc. 5th Runtime Verification*, volume 8734 of *LNCS*, 215–230.

Havelund, K.; Reger, G.; and Rosu, G. 2019. Runtime Verification Past Experiences and Future Projections. In *Computing and Software Science - State of the Art and Perspectives*, volume 10000 of *LNCS*, 532–562. Springer.

Leucker, M.; and Schallhart, C. 2009. A brief account of runtime verification. *J. Log. Algebraic Methods Program.*, 78(5): 293–303.

Ly, L. T.; Maggi, F. M.; Montali, M.; Rinderle-Ma, S.; and van der Aalst, W. M. P. 2015. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Inf. Syst.*, 54: 209–234.

Maggi, F. M.; Francescomarino, C. D.; Dumas, M.; and Ghidini, C. 2014. Predictive Monitoring of Business Processes. In *Proc. 26th CAiSE*, volume 8484 of *LNCS*, 457–472. Springer.

Maggi, F. M.; Montali, M.; Westergaard, M.; and van der Aalst, W. M. P. 2011. Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata. In *Proc. 9th BPM*, volume 6896 of *LNCS*, 132–147. Springer.

Presburger, M. 1929. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du I congres de Mathem. des Pays Slaves*, 92–101.

Reger, G.; Cruz, H. C.; and Rydeheard, D. E. 2015. MarQ: Monitoring at Runtime with QEA. In *Proc. 21st TACAS*, volume 9035 of *LNCS*, 596–610. Springer.

Xu, H.; and Cheng, Y. 2007. Model checking bidding behaviors in internet concurrent auctions. *Comput. Syst. Sci. Eng.*, 22(4).