Linear-Time Verification of Data-Aware Processes Modulo Theories via Covers and Automata

Alessandro Gianola[1], **Marco Montali**[2], **Sarah Winkler**[2]

[1]INESC-ID/Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
[2]Faculty of Engineering, Free University of Bozen-Bolzano, Bolzano, Italy
alessandro.gianola@tecnico.ulisboa.pt, {montali,winkler}@inf.unibz.it

Abstract

The need to model and analyse dynamic systems operating over complex data is ubiquitous in AI and neighboring areas, in particular business process management. Analysing such data-aware systems is a notoriously difficult problem, as they are intrinsically infinite-state. Existing approaches work for specific datatypes, and/or limit themselves to the verification of safety properties. In this paper, we lift both such limitations, studying for the first time linear-time verification for so-called *data-aware processes modulo theories* (DMTs), from the foundational and practical point of view. The DMT model is very general, as it supports processes operating over variables that can store arbitrary types of data, ranging over infinite domains and equipped with domain-specific predicates. Specifically, we provide four contributions. First, we devise a semi-decision procedure for linear-time verification of DMTs, which works for a very large class of datatypes obeying to mild model-theoretic assumptions. The procedure relies on a unique combination of automata-theoretic and cover computation techniques to respectively deal with linear-time properties and datatypes. Second, we identify an abstract, semantic property that guarantees the existence of a faithful finite-state abstraction of the original system, and show that our method becomes a decision procedure in this case. Third, we identify concrete, checkable classes of systems that satisfy this property, generalising several results in the literature. Finally, we present an implementation and an experimental evaluation over a benchmark of real-world data-aware business processes.

Introduction

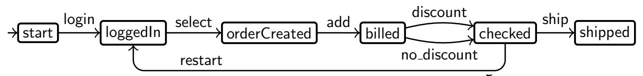
In many application domains of AI, the evolution of dynamic systems is inextricably intertwined with the progression of some form of data. This calls for representing the states of the system with richer structures than propositional assignments, making the system intrinsically infinite-state. Notable examples of such *data-aware processes* are (Situation Calculus) action theories (Baral and De Giacomo 2015; De Giacomo, Lespérance, and Patrizi 2016), lifted planning and planning over relational states/ontologies (Francès and Geffner 2016; Calvanese et al. 2016; Borgwardt et al. 2022), dynamic systems operating over databases (Calvanese, De Giacomo, and Montali 2013; Deutsch, Li, and Vianu 2019; Deutsch et al. 2018) and lightweight ontologies (Bagheri Hariri et al. 2013; Calvanese et al. 2023), and work processes in business process management (BPM) (Reichert 2012; Calvanese et al. 2019a; Gianola 2023). In these settings, verification is important due to the data-process interplay, but highly challenging due to infinity of the state space.

When considering, within this large spectrum, those approaches that come with foundational results paired with effective algorithmic techniques and implementations, we observe that they either: *(i)* operate on arbitrary datatypes while limiting verification to safety/reachability properties (Calvanese et al. 2020b; Ghilardi et al. 2023), *(ii)* handle general linear-time properties but only for specific datatypes, such as relational structures (Li, Deutsch, and Vianu 2017), or numerical variables (Demri and D'Souza 2007; Felli, Montali, and Winkler 2022), *(iii)* restrict both the model and the verification formalism (Masellis et al. 2017).

In this paper, we tackle such limitations and study linear-time verification for so-called *data-aware processes modulo theories* (DMTs), from the foundational and practical point of view. Our verification machinery is very general: DMTs capture dynamic systems manipulating variables that can store arbitrary types of data, ranging over infinite domains and equipped with domain-specific predicates, only imposing very mild assumptions on their underlying theory. In this respect, DMTs subsume several models studied in the literature for which very little is known regarding decidability of linear-time verification. Among them, particularly relevant and investigated are those where variables can store numeric values subject to linear arithmetic operations, and/or objects extracted from a read-only relational database or lightweight ontology (Demri and D'Souza 2007; Damaggio, Deutsch, and Vianu 2012; Bojańczyk, Segoufin, and Toruńczyk 2013; Calvanese et al. 2020b; Felli, Montali, and Winkler 2022; Calvanese et al. 2023). Notably, analysis is in this case reformulated as a parameterized verification problem, where the property of interest needs to hold irrespectively of how the read-only component is instantiated. Since process executions typically have a finite (yet unbounded) length, we express properties using a data-aware extension of LTL$_f$ (De Giacomo and Vardi 2013). However, data variables cause undecidability already for reachability over very limited systems (Bojańczyk, Segoufin, and Toruńczyk 2013).

We show a simple example for our verification problem.

Example 1. *In a webshop, a customer logs in using a customer database, and chooses five products from a product database. It is checked whether the customer is eligible for a 20% discount, and, if so, the discount is applied to the price of the order. If the balance in the customer's account is sufficient, the order is shipped; if not the process is restarted. The system can be modeled as a guarded transition system:*



login:	$\mathsf{Cust}(c^w, a^w, vip^w)$	select:	$\bigwedge_{i=1}^{5} \mathsf{ItemId}(p_i^w)$
add:	$t^w = \Sigma_{i=1}^{5} f_{price}(p_i^r)$	no_discount:	$\neg vip^r \wedge t^w = t^r$
ship:	$t^r \leq a^r$	restart:	$\top$
discount:	$vip^r \wedge t^w = t^r - \frac{1}{5} t^r$		

The process uses variables c, a, vip for customer id, account balance, and eligibility for discounts; p_i for product ids; and t for the total cost. These variables are read (superscript ^{r}) or written (superscript ^{w}) as specified by the transition guards. The DB relations $\mathsf{Cust}(\mathit{CustomerId}, \mathit{Account}, \mathit{IsVIP})$ *and* $\mathsf{Items}(\mathit{ItmId}, \mathit{Price})$ *hold customer and product data. Relevant verification properties are for example that discounts are only applied to eligible customers, and that each order is eventually shipped.*

To address problems like the one of Ex. 1, we develop a semi-decision procedure that constructs a faithful, symbolic abstraction of the state space. While in purely numeric settings this idea has been explored by relying on quantifier elimination (Felli, Montali, and Winkler 2022), the same approach cannot be lifted to theories formalizing data structures and relational databases, as quantifiers cannot be eliminated therein. We deal with this essential technical difficulty by combining automata-theoretic and cover computation (Calvanese et al. 2021, 2022; Gulwani and Musuvathi 2008) techniques to respectively deal with the temporal and datatype dimensions. More specifically, this paper makes the following four contributions: (1) We devise a model checking procedure for DMTs w.r.t. theories enjoying two mild theoretical assumptions: decidable satisfiability of quantifier-free formulas, and the computability of covers (Calvanese et al. 2021). (2) We propose the abstract criterion of *finite data history*, and show that for systems enjoying this criterion, our model checking procedure is a decision procedure. (3) This decidability criterion is shown to apply to several concrete classes of systems singled out in the literature, where guards combine database queries with arithmetic. The property of finite data history strictly generalizes several known decidability results from the literature (see below). (4) We demonstrate the feasibility of our approach by an SMT-based implementation of the model checking technique, and we experimentally tested it on an extensive benchmark of real dynamic systems taken from the BPM field. Our work provides the first approach for linear-time verification modulo theories of a very general class of data-aware dynamic systems, unifying in a single framework several models and results from the literature.

In the sequel, after discussing related work, we introduce the DMT model and the verification logic. We then describe the model checking procedure and how to build automata to capture data-aware LTL$_f$ properties. Next, we give a general decidability criterion and single out relevant, concrete decidable classes. We close by discussing implementation and experiments. Full proofs are in the online extended version (Gianola, Montali, and Winkler 2023).

Related Work. Notable approaches to the verification of dynamic systems operating over (read-only) relational databases (Bojańczyk, Segoufin, and Toruńczyk 2013; Calvanese et al. 2020b) or lightweight ontologies (Calvanese et al. 2023) focused on safety properties: we generalize all such approaches, since we support full LTL$_f$ verification. (Bojańczyk, Segoufin, and Toruńczyk 2013) is based on amalgamation, but it is known that the computability of covers implies amalgamation (Chang and Keisler 1990). However, (Calvanese et al. 2020b) also supports the richer setting of *relational artifact systems*, where elements extracted from the database can be inserted into updatable relations (thus going beyond states containing variables). Several works deal with linear-time verification of systems operating over purely numeric data, with no support of other datatypes (Felli, Montali, and Winkler 2022; Demri and D'Souza 2007; Demri 2006). A linear-time verification procedure was proposed for artifact systems with data dependencies and arithmetic (Damaggio, Deutsch, and Vianu 2012); this was shown to be a decision procedure for so-called *feedback free* systems, which restrict how operations can be chained over time. This is the only decidability result combining DBs and arithmetic that we are aware of. Our decidability criteria strengthen this result to the larger class of *bounded lookback* systems. A procedure for restricted linear-time verification of transition systems operating over databases was also presented by Deutsch, Li, and Vianu (2019), but the verification language is not full LTL, systems need to be *hierarchical*, and no arithmetic is supported. Related to our work is also a tableau-based semi-decision procedure for satisfiability of LTL$_f$ with general SMT constraints (Geatti, Gianola, and Gigante 2022), but no decidability results are given there.

Framework

We start with the necessary preliminaries. We consider a first-order multi-sorted *signature* $\Sigma = \langle \mathcal{S}, \mathcal{P}, \mathcal{F}, V, U \rangle$, where $\mathcal{S}$ is a set of sorts; $\mathcal{P}$ is a set of predicate and $\mathcal{F}$ a set of function symbols over $\mathcal{S}$; V is a set of *data variables*; and U is a set of variables disjoint from V that will be used for quantification; where all variables have a sort in $\mathcal{S}$. We assume that Σ contains equality predicates for all sorts in $\mathcal{S}$. Then, Σ-terms t are built in the usual way from $\mathcal{F}$, V, and U. An *atom* is of the form $p(t_1, \dots, t_k)$ for $p \in \mathcal{P}$ and terms $t_1, \dots, t_k$; and a literal is an atom or its negation. For a set of variables Z, a Σ-*constraint* c over Z is of the form $\exists u_1, \dots, u_l . \varphi$ such that $u_1, \dots, u_l \in U$, φ is a conjunction of Σ-literals, and all free variables in c are in Z; the set of all such constraints is denoted $\mathcal{C}_\Sigma(Z)$. Moreover, Σ-*formulas* φ have the form $\varphi ::= p(t_1, \dots, t_k) \mid \varphi \wedge \varphi \mid \neg \varphi \mid \exists u . \varphi$ where $p \in \mathcal{P}$, $t_1, \dots, t_k$ are terms of appropriate sort, and $u \in U$. We use the usual shorthands $\forall$, $\vee$, and $\leftrightarrow$ for universal quantification, disjunction, and equivalence. We call φ a *state formula* if all its free variables are in V. Σ-formulas without free variables are Σ-*sentences*, and a set of Σ-sentences is a

Σ*-theory* $\mathcal{T}$. It is *universal* if all its sentences have the form $\forall u_1, \dots, u_l.\, \varphi$ with φ quantifier-free.

To define semantics, we use the standard notion of a Σ-*structure* M, which associates each sort $s \in \mathcal{S}$ with a domain s^M, and each predicate $p \in \mathcal{P}$ and function symbol $f \in \mathcal{F}$ with a suitable interpretation p^M and f^M. The equality predicates have the standard interpretation given by the identity relation. The carrier of M, i.e., the union of all domains of sorts in $\mathcal{S}$, is denoted by $|M|$. A total function $\alpha\colon V \to |M|$ is called a *state variable assignment*. We also use α to denote a partial assignment $\alpha\colon V \cup U \to |M|$ with $V \subseteq Dom(\alpha)$. We always assume that variables are mapped to an element of their respective domain. We write $\alpha[u \mapsto e]$ for the extended assignment such that $\alpha[u \mapsto e](u) = e$ and $\alpha[u \mapsto e](x) = \alpha(x)$ for all $x \neq u$.

Given M and α, the *evaluation* $[t]^M_\alpha$ of a term t is defined as $[v]^M_\alpha = \alpha(v)$ if $v \in V \cup U$, and $[f(t_1, \dots, t_k)]^M_\alpha = f^M([t_1]^M_\alpha, \dots, [t_k]^M_\alpha)$. Whether a Σ-formula φ is *satisfied* by M and α, denoted $\alpha \models_M \varphi$, is defined as follows:

$\alpha \models_M p(t_1, \dots, t_k)$ if $p^M([t_1]^M_\alpha, \dots, [t_k]^M_\alpha)$ holds
$\alpha \models_M \varphi_1 \wedge \varphi_2$ if $\alpha \models_M \varphi_1$ and $\alpha \models_M \varphi_2$
$\alpha \models_M \neg\varphi$ if $\alpha \not\models_M \varphi$
$\alpha \models_M \exists u.\, \varphi$ if $\exists\, e \in |M|$ s.t. $\alpha[u \mapsto e] \models_M \varphi$

Note that $\alpha \models_M \varphi$ is always defined if φ is a state formula and α a state variable assignment. If φ is a Σ-sentence, we write $\models_M \varphi$ for $\emptyset \models_M \varphi$. We adopt a common SMT perspective and view a Σ-theory $\mathcal{T}$ as the set of all Σ-structures that satisfies all axioms of $\mathcal{T}$, and write $M \in \mathcal{T}$ if M is a model of $\mathcal{T}$, i.e., $\models_M \varphi$ holds for all φ in $\mathcal{T}$. A state formula φ is $\mathcal{T}$*-satisfiable* if there is some $M \in \mathcal{T}$ and state variable assignment $\alpha\colon V \to |M|$ such that $\alpha \models_M \varphi$. Moreover, two state formulas φ_1 and φ_2 are $\mathcal{T}$*-equivalent*, denoted $\varphi_1 \equiv_\mathcal{T} \varphi_2$, if $\neg(\varphi_1 \leftrightarrow \varphi_2)$ is not $\mathcal{T}$-satisfiable.

A Σ-theory $\mathcal{T}$ has *quantifier elimination* (QE) if for every Σ-formula φ there is a quantifier-free formula φ' that is $\mathcal{T}$-equivalent to φ. As QE is a strong requirement, we make use of the weaker notion of *covers*. Intuitively, a $\mathcal{T}$-cover of an existential formula ψ is the strongest quantifier-free formula $\mathcal{T}$-implied by ψ. Precisely, given a universal theory $\mathcal{T}$ and an existential formula $\psi := \exists x.\, \psi'$ where ψ' has free variables $\{x\} \cup Y$, a $\mathcal{T}$-cover of ψ is a quantifier-free formula φ with free variables Y such that: 1) $\mathcal{T} \models \psi \to \varphi$; and 2) for all quantifier-free formulae φ' with free variables $Y \cup Z$ such that $\mathcal{T} \models \psi \to \varphi'$, we have $\mathcal{T} \models \varphi \to \varphi'$. Computing covers in $\mathcal{T}$ is equivalent to $\mathcal{T}$ having a *model completion* (Calvanese et al. 2019b, 2021): a universal Σ-theory $\mathcal{T}$ has a model completion (Ghilardi 2004; Chang and Keisler 1990) iff there exists a Σ-theory $\mathcal{T}^*$ such that (i) every Σ-constraint satisfiable in a model of $\mathcal{T}$ is satisfiable in a model of $\mathcal{T}^*$, and (ii) $\mathcal{T}^*$ has QE. Throughout the paper, in the formal proofs we make use of model completions, since they are easier to handle when adopting, as we do, a model-theoretic approach.

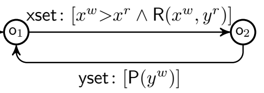
We will sometimes refer to common SMT theories (Barrett et al. 2021): the theory of equality and uninterpreted functions (EUF) for a given Σ, and linear arithmetic over rationals (LRA), integers (LIA), or both (LIRA). While the arithmetic theories have QE (Presburger 1929), EUF admits model completion and so do certain *tame* combinations of LIRA and EUF (Calvanese et al. 2020a, 2022). We denote the sorts of rationals and integers by rat and int.

Data-Aware Processes Modulo Theories. Let $\Sigma = \langle \mathcal{S}, \mathcal{P}, \mathcal{F}, V, U \rangle$ be a signature. For each data variable $v \in V$, let v^r and v^w be two annotated variables of the same sort, and set $V^r = \{v^r \mid v \in V\}$ and $V^w = \{v^w \mid v \in V\}$. These copies of V are called the *read* and *write* variables; they will denote the variable values before and after a transition, respectively. Moreover, $\overline{V}$ denotes a vector that sorts the variables V in an arbitrary but fixed order.

Definition 2. A *data-aware process modulo theories* over Σ (Σ-DMT for short) is a labelled transition system $\mathcal{B} = \langle \Sigma, V, I, T \rangle$, where: *(i)* Σ is a signature, *(ii)* V is the finite, nonempty set of data variables in Σ; *(iii)* the transition formulae $T(V^r, V^w)$ are a set of constraints in $\mathcal{C}_\Sigma(V^r \cup V^w)$; *(iv)* $I\colon V \to \mathcal{F}_0$, called initial function, initializes variables, where $\mathcal{F}_0$ is the set of Σ-constants.

Example 3. *As a running example, we consider a simple DMT* $\mathcal{B} = \langle \Sigma, V, I, T \rangle$ *where the theory combines LRA with EUF using uninterpreted sorts status and elem, uninterpreted predicates* $\mathsf{R} \subseteq rat \times elem$, $\mathsf{P} \subseteq elem$ *and constants* a, b *of sort elem, and* o_1, o_2 *of sort status. The set of variables* V *consists of* x *(sort rat) and* y *(sort elem), and a variable* s *of sort status that takes values* o_1 *and* o_2. *We set* $I(s){=}\mathsf{o}_1$, $I(x){=}0$ *and* $I(y){=}\mathsf{a}$. *The transitions* $T = \{\mathsf{xset}, \mathsf{yset}\}$ *are as follows:*

$$\mathsf{xset} = (s^r = \mathsf{o}_1 \wedge s^w = \mathsf{o}_2 \wedge x^w{>}x^r \wedge \mathsf{R}(x^w, y^r))$$
$$\mathsf{yset} = (s^r = \mathsf{o}_2 \wedge s^w = \mathsf{o}_1 \wedge \mathsf{P}(y^w))$$


xset: $[x^w{>}x^r \wedge \mathsf{R}(x^w, y^r)]$
o_1
o_2
yset: $[\mathsf{P}(y^w)]$

The transition system is a visualization of $\mathcal{B}$ *where the status* s *is interpreted as a control state, to help readability. Transitions simultaneously express conditions on read variables (superscript* ^{r}*), and updates on the written ones (superscript* ^{w}*): e.g.,* xset *requires the current control state* (s^r) *to be* o_1, *fixes the next control state* (s^w) *to* o_2, *and nondeterministically sets* x *to a new value* (x^w) *that is greater than its current value* (x^r), *and in relation* R *with* y^r.

Similarly, also Ex. 1 can be formalized as a Σ-DMT.

We next define the semantics for Σ-DMTs. For a Σ-theory $\mathcal{T}$ and a model $M \in \mathcal{T}$, a *state* of a Σ-DMT $\mathcal{B}$ is an assignment $\alpha\colon V \to |M|$. A *guard assignment* β is a function $\beta\colon V^r \cup V^w \to |M|$. As defined next, a transition t can transform a state α into a new state α', updating the variable values in agreement with t, while variables that are not explicitly written keep their previous value as per α.

Definition 4. A Σ-DMT $\mathcal{B} = \langle \Sigma, V, I, T \rangle$ admits a $\mathcal{T}$*-step* from state α to α' via transition $t \in T$ w.r.t. a model $M \in \mathcal{T}$, denoted $\alpha \xrightarrow{t}_M \alpha'$, if there is some guard assignment β s.t. $\beta \models_M t$, $\beta(v^r) = \alpha(v)$ and $\beta(v^w) = \alpha'(v)$ for all $v \in V$.

A $\mathcal{T}$*-run* of $\mathcal{B}$ is a pair (M, ρ) of a model $M \in \mathcal{T}$ and a sequence of steps of the form $\rho\colon \alpha_0 \xrightarrow{t_1}_M \alpha_1 \xrightarrow{t_2}_M \cdots \xrightarrow{t_n}_M \alpha_n$ where $\alpha_0(v) = I(v)^M$ for all $v \in V$. Note that given M, the initial assignment α_0 of a run is uniquely determined by the initializer I of $\mathcal{B}$. The transition sequence $\langle t_1, \dots, t_n \rangle$ is called the *abstraction* of ρ, and is denoted by $\sigma(\rho)$. If clear from the context, we omit the model M in the notation $\xrightarrow{t}_M$; and sometimes we refer to a *run*, leaving the theory implicit.

For Exs. 1 and 3, a natural choice for the theory $\mathcal{T}$ is that of arithmetic over $\mathbb{Q}$ together with equality and uninterpreted functions (i.e., the union of LRA and EUF), plus axioms that fix entries in the database. E.g. for Ex. 3, if M satisfies $\mathsf{R}(4, \mathsf{a})$ and $\mathsf{R}(8, \mathsf{b})$, then (M, ρ) is a run, for

$$\rho\colon \begin{Bmatrix} s=\mathsf{o}_1\\ x=0\\ y=\mathsf{a}\end{Bmatrix} \xrightarrow{\mathsf{xset}} \begin{Bmatrix} s=\mathsf{o}_2\\ x=4\\ y=\mathsf{a}\end{Bmatrix} \xrightarrow{\mathsf{yset}} \begin{Bmatrix} s=\mathsf{o}_1\\ x=4\\ y=\mathsf{b}\end{Bmatrix} \xrightarrow{\mathsf{xset}} \begin{Bmatrix} s=\mathsf{o}_2\\ x=8\\ y=\mathsf{b}\end{Bmatrix}.$$

Verification language. We assume that V is the set of data variables of a given DMT $\mathcal{B}$, and all variables in verification properties are in V. Formally, our verification language $\mathcal{L}_\Sigma$ (called *data-LTL$_f$*) consists of all properties ψ:

$$\psi := c \mid \psi{\wedge}\psi \mid \psi{\vee}\psi \mid \mathsf{X}\psi \mid \mathsf{G}\psi \mid \psi \mathbin{\mathsf{U}} \psi$$

where c is a constraint in $\mathcal{C}_\Sigma(V)$. Note that $\mathcal{L}_\Sigma$ does not support negation, but can express formulae in negation normal form. For convenience, we abbreviate $\top := (v = v)$ for some $v \in V$, and $\mathsf{F}\psi := \top \mathbin{\mathsf{U}} \psi$. We adopt LTL semantics over finite traces (LTL$_f$) (De Giacomo and Vardi 2013):

Definition 5. A run (M, ρ) of $\mathcal{B}$ for $\rho\colon \alpha_0 \xrightarrow{t_1}_M \alpha_1 \xrightarrow{t_2}_M \cdots \xrightarrow{t_n}_M \alpha_n$ *satisfies* $\psi \in \mathcal{L}_\Sigma$, denoted $\rho \models_M \psi$, iff $\rho \models^0_M \psi$ holds, where for all i, $0 \le i \le n$:

$\rho \models^i_M c$ iff $\alpha_i \models_M c$
$\rho \models^i_M \psi_1 \wedge \psi_2$ iff $\rho \models^i_M \psi_1$ and $\rho \models^i_M \psi_2$
$\rho \models^i_M \psi_1 \vee \psi_2$ iff $\rho \models^i_M \psi_1$ or $\rho \models^i_M \psi_2$
$\rho \models^i_M \mathsf{X}\psi$ iff $i < n$ and $\rho \models^{i+1}_M \psi$
$\rho \models^i_M \mathsf{G}\psi$ iff $\rho \models^i_M \psi$ and ($i = n$ or $\rho \models^{i+1}_M \mathsf{G}\psi$)
$\rho \models^i_M \psi_1 \mathbin{\mathsf{U}} \psi_2$ iff $\rho \models^i_M \psi_2$, or ($i < n$ and both $\rho \models^i_M \psi_1$ and $\rho \models^{i+1}_M \psi_1 \mathbin{\mathsf{U}} \psi_2$)

A $\mathcal{T}$-run (M, ρ) is a $\mathcal{T}$*-witness* for $\psi \in \mathcal{L}_\Sigma$ if $\rho \models_M \psi$. The problem addressed in this paper is the following:

Definition 6 ($\mathcal{T}$-verification task)**.** Given a Σ-DMT $\mathcal{B}$, a Σ-theory $\mathcal{T}$, and $\psi \in \mathcal{L}_\Sigma$, check whether there exists a $\mathcal{T}$-witness for ψ in $\mathcal{B}$.

The DMT in Ex. 3 has a witness for $(x \ge 0) \mathbin{\mathsf{U}} (s = \mathsf{o}_2 \wedge x = 4)$, e.g. the run (M, ρ) above. The DMT in Ex. 1 has no witness for $\mathsf{F}(s{=}\mathsf{shipped} \wedge \neg vip \wedge a < \sum_{i=1}^5 f_{price}(p_i))$, so an order is not shipped if the account balance is insufficient.

Remark 7. *Def. 6 generalizes related verification tasks for data-aware processes, notably (1) verification of safety properties in Simple Artifact Systems (SASs) over relational databases with key dependencies (Calvanese et al. 2020b), and (2) linear-time model checking of DDSs with arithmetics (Felli, Montali, and Winkler 2022). SASs can be represented as DMTs by taking the theory EUF with constants, unary functions, and arbitrary relations, possibly together with a read-only database.* Artifact variables *of SASs can be represented as data variables in V, and safety properties can be expressed in $\mathcal{L}_\Sigma$. DDSs can be formalized as DMTs by taking the theory LIRA: The initial assignment of variables can be encoded in I, one additional designated variable can be used to model DDS control states, and the transition formulae T in DMTs can represent guarded actions. DDSs have final states, but any verification property can be extended to require that such a state is reached.*

LTL$_f$ verification is undecidable for dynamic systems over numeric variables, as reachability in 2-counter machines can be encoded (Felli, Montali, and Winkler 2022), so the verification task considered here is undecidable, too.

Model Checking

From now on, we assume a fixed Σ-theory $\mathcal{T}$ that satisfies:

Assumptions 8. (a) Satisfiability of quantifier-free $\mathcal{T}$-formulas is decidable; (b) $\mathcal{T}$ is either universal and has a model completion $\mathcal{T}^*$; or has QE, in this case let $\mathcal{T}^* := \mathcal{T}$.

Remark 9. *As constraints are existential formulas, Assumption (b) implies that $\mathcal{T}$-satisfiability and $\mathcal{T}^*$-satisfiability of Σ-constraints coincide (Calvanese et al. 2020b).*

Automata construction. It is well-known that LTL$_f$ properties can be captured by non-deterministic automata (NFA) (De Giacomo and Vardi 2013; De Giacomo, De Masellis, and Montali 2014). Our model checking procedure relies on such an NFA for a given verification property $\psi \in \mathcal{L}_\Sigma$. Its construction differs from earlier work only in that propositional/numeric atoms are replaced by constraints. We sketch here the general approach: For a Σ-DMT $\mathcal{B} = \langle \Sigma, V, I, T\rangle$ and $\psi \in \mathcal{L}_\Sigma$, let $cstr(\psi)$ be the set of all constraints in ψ. We then build an NFA $\mathcal{N}_\psi = (Q, \Theta, \varrho, q_0, \{q_f, q_e\})$ where: *(i)* the set Q of states consists of properties $\varphi \in \mathcal{L}_\Sigma \cup \{\top, \bot\}$, *(ii)* the alphabet is $\Theta := 2^{cstr(\psi)}$, i.e., a symbol is a subset of constraints in ψ, *(iii)* the initial state is $q_0 := \psi$, *(iv)* $q_f := \top$ is a final state, *(v)* q_e is an additional, designated final state, *(vi)* $\varrho \subseteq Q \times \Theta \times Q$ is a set of transitions.

To state correctness, let a word $\langle \varsigma_0, \dots, \varsigma_n\rangle \in \Theta^*$ be *consistent* with a run (M, ρ) for $\rho\colon \alpha_0 \xrightarrow{t_1} \alpha_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \alpha_n$ if $\alpha_i \models_M \bigwedge \varsigma_i$ for all $0 \le i \le n$. The main result about $\mathcal{N}_\psi$ is:

Proposition 1. *$\mathcal{N}_\psi$ accepts a word w that is consistent with (M, ρ) iff $\rho \models_M \psi$.*

All details can be found in (Gianola, Montali, and Winkler 2023). We show here only the construction for an example.

Example 10. *The NFA $\mathcal{N}_\psi$ for $\psi := (x \ge 0) \mathbin{\mathsf{U}} (s = \mathsf{o}_2 \wedge x = 4)$ is shown below. The initial state ψ has a loop labeled $x \ge 0$ that can be taken until both $s = \mathsf{o}_2$ and $x = 4$ hold; at this point the final state $\top$ can be reached. For the construction see (Gianola, Montali, and Winkler 2023, Ex. 31)*

$$\begin{tikzcd}
\psi \arrow[loop left, "\{x \ge 0\}"] \arrow[rr, "\{x=4,\, s=\mathsf{o}_2\}"] & & \boxed{\top} \arrow[loop right, "\emptyset"]
\end{tikzcd}$$

Verification procedure. We next develop our model checking technique. To this end, we assume a Σ-theory $\mathcal{T}$, a Σ-DMT $\mathcal{B} = \langle \Sigma, V, I, T\rangle$, and $\psi \in \mathcal{L}_\Sigma$ with associated NFA $\mathcal{N}_\psi$ and alphabet Θ as above. We first aim to show the crucial fact that witnesses exist in $\mathcal{T}$ iff they do in $\mathcal{T}^*$:

Theorem 11. *A Σ-DMT $\mathcal{B}$ admits a $\mathcal{T}$-witness for a property $\psi \in \mathcal{L}_\Sigma$ iff it admits a $\mathcal{T}^*$-witness for ψ.*

To prove Thm. 11, we introduce notions that will be used throughout the paper: Let $write(t) := \{x \mid x^w \in V^w$ occurs in $t\}$ be the set of variables written by a transition $t \in T$. We write $\widehat{t}$ for the *extended transition formula* $\widehat{t} := t \wedge \bigwedge_{v \notin write(t)} v^w = v^r$, which simply expresses that the transition formula t must hold and the values of all variables that are not explicitly written are propagated. Below, we consider disjoint variable sets $V_0, V_1, V_2, \dots$ that are indexed copies of V, i.e. $V_i := \{v_i \mid v \in V\}$, and assume that $\overline{V}_0, \overline{V}_1, \overline{V}_2, \dots$ are ordered in the same way as $\overline{V}$. For a formula φ with free variables V, we write $\varphi(\overline{V}_i)$ for the formula obtained by renaming $\overline{V}$ to $\overline{V}_i$; and for t with free variables $V^r \cup V^w$, the formula $t(\overline{V}_i, \overline{V}_j)$ replaces $\overline{V}^r$ by $\overline{V}_i$ and $\overline{V}^w$ by $\overline{V}_j$. Moreover, $\varphi_I := \bigwedge_{v \in V} v_0{=}I(v)$ is a formula that encodes the initial state.

We next define a formula that combines all constraints accumulated in a transition sequence, together with a word w that expresses additional constraints relevant for verification. For readability, we use $\varsigma \in \Theta$ as a formula to mean $\bigwedge \varsigma$.

Definition 12. For a transition sequence $\sigma{=}\langle t_1, \dots, t_n\rangle$ of $\mathcal{B}$ and $w = \langle \varsigma_0, \dots, \varsigma_n\rangle \in \Theta^*$, the formula $H(\sigma, w)$ is

$$\varphi_I \wedge \varsigma_0(\overline{V}_0) \wedge \widehat{t_1}(\overline{V}_0, \overline{V}_1) \wedge \varsigma_1(\overline{V}_1) \wedge \dots \wedge \widehat{t_n}(\overline{V}_{n-1}, \overline{V}_n) \wedge \varsigma_n(\overline{V}_n)$$

E.g., for the transition sequence $\sigma = \langle \mathsf{xset}\rangle$ of the DMT in Ex. 3 and $w = \langle \{x \geq 0\}, \{x{=}4, s{=}\mathsf{o}_2\}\rangle$, $H(\sigma, w) = (s_0{=}\mathsf{o}_1 \wedge x_0{=}0 \wedge y_0{=}\mathsf{a}) \wedge (x_0{\geq}0) \wedge (s_0{=}\mathsf{o}_1 \wedge s_1{=}\mathsf{o}_2 \wedge x_1{>}x_0 \wedge \mathsf{R}(x_1, y_0) \wedge y_1{=}y_0 \wedge x_1{=}4))$. In Lem. 14 below we prove that satisfying assignments for formulas $H(\sigma, w)$ are witnesses for ψ, if w is accepted by $\mathcal{N}_\psi$. To state this, we need some notation to link assignments with runs:

Definition 13. Let $\sigma = \langle t_1, \dots, t_n\rangle$ be a transition sequence, $w \in \Theta^*$, and M be a Σ-structure.

- If (M, ρ) is a run of $\mathcal{B}$ for ρ: $\alpha_0 \xrightarrow{t_1} \alpha_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \alpha_n$ the *run assignment* $\nu(\rho)$ has domain $\bigcup_{i=0}^n V_i$ and sets $\nu(\rho)(v_i) := \alpha_i(v)$, for all $v \in V$ and $0 \leq i \leq n$.
- For an assignment ν such that $\nu \models_M H(\sigma, w)$ the *decoded run* $\rho(M, \nu, \sigma)$ is the sequence $\alpha_0 \xrightarrow{t_1} \alpha_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \alpha_n$ where $\alpha_i(v) := \nu(v_i)$, for all $v \in V$ and i.

Lemma 14. (1) If $\mathcal{B}$ admits a witness (M, ρ) for ψ, there is a word w consistent with ρ and accepted by $\mathcal{N}_\psi$ such that $\nu(\rho) \models_M H(\sigma(\rho), w)$.
(2) If $\nu \models_M H(\sigma, w)$ for some assignment ν, transition sequence σ of $\mathcal{B}$, and word w accepted by $\mathcal{N}_\psi$, then $\rho(M, \nu, \sigma)$ is a witness for ψ of $\mathcal{B}$.

Proof (sketch). Both directions are by induction on ρ and σ. For the connection between a witness and $\mathcal{N}_\psi$ accepting a word consistent with ρ, Prop. 1 is used. □

At this point, we are ready to prove Thm. 11.

Proof (of Thm. 11). If (M, ρ) is a $\mathcal{T}$-witness for ψ then by Lem. 14 (1), $(M, \nu(\rho))$ satisfies $H(\sigma(\rho), w)$ for some w consistent with ρ and accepted by $\mathcal{N}_\psi$. By Rem. 9, there must be some (M', ν') which satisfies $H(\sigma(\rho), w)$ in $\mathcal{T}^*$ as $H(\sigma(\rho), w)$ is an existential formula. By Lem. 14 (2), $\rho(M', \nu', \sigma)$ is a $\mathcal{T}^*$-witness for ψ. □

We next define our main data structure for model checking: a product automaton $\mathcal{N}_\mathcal{B}^\psi$ built from the NFA $\mathcal{N}_\psi$ and the Σ-DMT $\mathcal{B}$. The construction uses an *update* function to capture how the current variable configuration, expressed as a formula φ with free variables V, changes by executing a transition t. Namely, $\mathit{update}(\varphi, t) := \exists \overline{X}. \varphi(\overline{X}) \wedge \widehat{t}(\overline{X}, \overline{V})$, where $\overline{X}$ is a fresh copy of $\overline{V}$. Below, we write $\Phi(V)$ for the set of all quantifier-free Σ-formulae over variables V.

Definition 15. Given $\mathcal{B} = \langle \Sigma, V, I, T\rangle$ and the NFA $\mathcal{N}_\psi$ as above, the *product automaton* $\mathcal{N}_\mathcal{B}^\psi := (P, R, p_0, P_F)$ with states $P \subseteq Q \times \Phi(V)$, transition relation R, initial state p_0, and final states P_F is inductively defined as follows:

- P contains the initial state $p_0 := (q_0, \varphi_I)$; and
- if $(q, \varphi) \in P$, $q \xrightarrow{\varsigma} q'$ in $\mathcal{N}_\psi$ and either
  (i) $(q, \varphi) = p_0$ and $t = \top$, or
  (ii) $(q, \varphi) \neq p_0$ and $t \in T$
  such that the formula $\xi := \mathit{update}(\varphi, t) \wedge \varsigma$ is $\mathcal{T}$-satisfiable, there is some $(q', \varphi') \in P$ s.t. $\varphi' {\equiv_{\mathcal{T}^*}} \xi$, and $(q, \varphi) \xrightarrow{t,\varsigma} (q', \varphi')$ is in R,
- P_F consists of all $(q, \varphi) \in P$ such that q is final in $\mathcal{N}_\psi$.

We add some remarks on Def. 15: First, the distinction of the two kinds of transitions (i) and (ii) is a technical requirement as constraints in the property ψ are evaluated in states of a run but constraints in $\mathcal{B}$ are on transitions. To achieve this "offset", we use a dummy transition $t = \top$ from the initial state. Second, a quantifier-free formula φ' as required in Def. 15 exists because $\mathcal{T}^*$ has QE. Finally, the product automaton is not unique due to many $\mathcal{T}^*$-equivalent formulae.

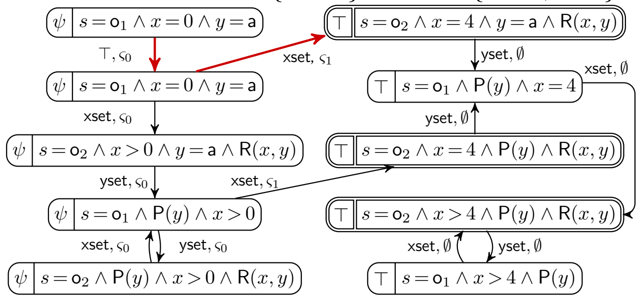
We now consider *paths* π in $\mathcal{N}_\mathcal{B}^\psi$ that start in the initial state and thus have, for some $n \geq 0$, the form

$$(q_0, \varphi_I) \xrightarrow{\top, \varsigma_0} (q_1, \varphi_1) \xrightarrow{t_1, \varsigma_1} \dots \xrightarrow{t_n, \varsigma_n} (q_{n+1}, \varphi_{n+1}) \qquad (1)$$

i.e., the first edge is labeled $(\top, \varsigma_0)$ for some ς_0. We write $\sigma_\pi = \langle t_1, \dots, t_n\rangle$ for the transition sequence of $\mathcal{B}$ and w_π for the word $\langle \varsigma_0, \dots, \varsigma_n\rangle$ read from the edge labels of π. Our main result relates paths to final states with witnesses:

Theorem 16. *(1) For every path π to a final state of $\mathcal{N}_\mathcal{B}^\psi$, $H(\sigma_\pi, w_\pi)$ is satisfiable by some (M, ν) for $M \in \mathcal{T}$, and $\rho(M, \nu, \sigma_\pi)$ is a $\mathcal{T}$-witness for ψ.*
(2) If $\mathcal{B}$ admits a $\mathcal{T}$-witness for ψ then $\mathcal{N}_\mathcal{B}^\psi$ has a final state.

Example 17. *The product of $\mathcal{B}$ from Ex. 3 with the NFA $\mathcal{N}_\psi$ from Ex. 10 for $\psi := (x \geq 0)\ \mathsf{U}\ (s = \mathsf{o}_2 \wedge x = 4)$ is as shown next. We abbreviate $\varsigma_0 = \{x \geq 0\}$ and $\varsigma_1 = \{s{=}\mathsf{o}_2, x{=}4\}$.*



ψ | $s=\mathsf{o}_1 \wedge x=0 \wedge y=\mathsf{a}$
$\top, \varsigma_0$
ψ | $s=\mathsf{o}_1 \wedge x=0 \wedge y=\mathsf{a}$
xset, ς_0
ψ | $s=\mathsf{o}_2 \wedge x>0 \wedge y=\mathsf{a} \wedge \mathsf{R}(x,y)$
yset, ς_0
ψ | $s=\mathsf{o}_1 \wedge \mathsf{P}(y) \wedge x>0$
xset, ς_0
yset, ς_0
ψ | $s=\mathsf{o}_2 \wedge \mathsf{P}(y) \wedge x>0 \wedge \mathsf{R}(x,y)$
xset, ς_1
$\top$ | $s=\mathsf{o}_2 \wedge x=4 \wedge y=\mathsf{a} \wedge \mathsf{R}(x,y)$
yset, $\emptyset$
$\top$ | $s=\mathsf{o}_1 \wedge \mathsf{P}(y) \wedge x=4$
xset, $\emptyset$
yset, $\emptyset$
$\top$ | $s=\mathsf{o}_2 \wedge x=4 \wedge \mathsf{P}(y) \wedge \mathsf{R}(x,y)$
xset, ς_1
$\top$ | $s=\mathsf{o}_2 \wedge x>4 \wedge \mathsf{P}(y) \wedge \mathsf{R}(x,y)$
xset, $\emptyset$
yset, $\emptyset$
$\top$ | $s=\mathsf{o}_1 \wedge x>4 \wedge \mathsf{P}(y)$

By Thm. 16 (1), witnesses can be obtained from all paths to accepting states. E.g., from the path π shown in red we extract the single-step transition sequence $\sigma = \langle \mathsf{xset}\rangle$ and the word $w = \langle \{x \geq 0\}, \{s{=}\mathsf{o}_2, x{=}4\}\rangle$, with $H(\sigma, w)$ as shown below Def. 12; it is $\mathcal{T}$-satisfied by any model M s.t. $(4, \mathsf{a}) \in \mathsf{R}_M$ and $\nu(s_0) = \mathsf{o}_1$, $\nu(s_1) = \mathsf{o}_2$, $\nu(x_0) = 0$, $\nu(x_1) = 4$, and $\nu(y_0) = \nu(y_1) = \mathsf{a}$. The witness $\rho(M, \nu, \sigma)$ is thus $\{s = \mathsf{o}_1, x = 0, y = \mathsf{a}\} \xrightarrow{\mathsf{xset}} \{s = \mathsf{o}_2, x = 4, y = \mathsf{a}\}$.

In general, the product construction $\mathcal{N}_\mathcal{B}^\psi$ from Def. 15 can be infinite, but Thm. 16 gives rise to a semi-decision procedure: $\mathcal{N}_\mathcal{B}^\psi$ can be approximated by expanding Def. 15 in a fair way. Then an accepting path π will be detected if it exists, so that a witness can be constructed.

Correctness and termination. To prove Thm. 16, and show that the semi-decision procedure actually serves as a decision procedure in notable cases, we employ some additional notions. We do so using ideas from (Felli, Montali, and Winkler 2022), with the key difference that reasoning is delegated to $\mathcal{T}^*$, and moreover we have no control states.

Let $\sigma = \langle t_1, \dots, t_n \rangle$ be a transition sequence and $w \in \Theta^*$ have length $n{+}1$. We define the *history constraint* $H_\exists(\sigma, w) := (\exists \overline{V}_0 \dots \overline{V}_{n-1}.\ H(\sigma, w))(\overline{V})$, which is a formula with free variables V.

The next result formalizes in which sense $\mathcal{N}^\psi_\mathcal{B}$ is a product construction: a path π of the form (1) combines σ_π with w_π and the last formula in π is equivalent to $H_\exists(\sigma_\pi, w_\pi)$.

Lemma 18. $\mathcal{N}^\psi_\mathcal{B}$ has a non-empty path π to a node (q, φ) iff $\mathcal{N}_\psi$ has a transition sequence ending in q labeled w_π and $\mathcal{B}$ a transition sequence σ_π such that $\varphi \equiv_{\mathcal{T}^*} H_\exists(\sigma_\pi, w_\pi)$ and φ is $\mathcal{T}^*$-satisfiable.

Both directions are by plain induction. Now we can show:

Proof (of Thm. 16). (1) Let π be a path to a final state (q_n, φ) in $\mathcal{N}^\psi_\mathcal{B}$. So q_n is final in $\mathcal{N}_\psi$, and we cannot have $q_n = q_0$ as $\psi \neq \top$. By Lem. 18 ($\Longrightarrow$), there is a transition sequence in $\mathcal{N}_\psi$ labeled $w_\pi = \langle \varsigma_0 \dots \varsigma_n \rangle$ and $H_\exists(\sigma_\pi, w_\pi)$ is $\mathcal{T}^*$-satisfiable. So also $H(\sigma_\pi, w_\pi)$ must be $\mathcal{T}^*$-satisfiable, and by Rem. 9 also $\mathcal{T}$-satisfied by some $M \in \mathcal{T}$ and ν. By Lem. 14 (2), $\rho(M, \nu, \sigma)$ is a witness for ψ. (2) If there is a $\mathcal{T}$-witness for ψ, by Thm. 11, there is a $\mathcal{T}^*$-witness (M, ρ). By Lem. 14 (1), there is a word w consistent with ρ and accepted by $\mathcal{N}_\psi$ such that $H(\sigma(\rho), w)$ is $\mathcal{T}$-satisfiable. By Lem. 18 ($\Longleftarrow$), the accepting transition sequence of $\mathcal{N}_\psi$ labeled w and $\sigma(\rho)$ give rise to a path π in $\mathcal{N}^\psi_\mathcal{B}$ such that $w = w_\pi$ and $\sigma(\rho) = \sigma_\pi$. As $\mathcal{N}_\psi$ accepts w, π leads to a final state. $\square$

We next state an abstract decidability criterion.

Definition 19. A *data history* for $(\mathcal{B}, \psi)$ is a minimal set of quantifier-free formulae Φ such that for every transition sequence σ of $\mathcal{B}$ and $w \in \Theta^*$, there is some $\varphi \in \Phi$ such that $H_\exists(\sigma, w) \equiv_{\mathcal{T}^*} \varphi$.

Intuitively, a data history has a formula representation of every pair of a transition sequence in $\mathcal{B}$, and a sequence of constraints needed to verify ψ. As formulas in $\mathcal{N}^\psi_\mathcal{B}$ are history constraints (Lem. 18), if $(\mathcal{B}, \psi)$ has finite data history then a finite $\mathcal{N}^\psi_\mathcal{B}$ exists, so by Thm. 16 we have decidability:

Corollary 20. The verification task is decidable if $(\mathcal{B}, \psi)$ has a finite data history.

Below, we use Cor. 20 to identify concrete, decidable classes of DMTs that involve databases and arithmetic. However, our approach is not limited to this domain, cf. (Gianola, Montali, and Winkler 2023, Ex. 32) which uses list theory.

Decidability Criteria

In this section, we exploit Cor. 20 to identify concrete, checkable classes of DMTs where our verification task is decidable. We focus on systems that query a read-only database, which are crucial to BPM and database processes (Bojańczyk, Segoufin, and Toruńczyk 2013; Calvanese, De Giacomo, and Montali 2013; Deutsch et al. 2018); and extend this setting with arithmetic, a key combination in practice. Before turning to decidable classes, we recall related work on reasoning about databases in an SMT context.

Static DB schemas. In order to define databases in an SMT context, we rely on the notion of *static DB schemas* from (Ghilardi et al. 2023), where classical relational databases are formalized in an algebraic way and are extended with integer and real values. In this setting, static DB schemas consist of the combination of two first-order theories, $\mathcal{T}_{db}$ and $\mathcal{T}_{ar}$. The theory $\mathcal{T}_{db}$ models read-only DB relations with primary and foreign key constraints by employing function symbols to represent the key dependencies: formally, this can be captured via pure identifiers and functions subject to multi-sorted EUF. The theory $\mathcal{T}_{ar}$ formalizes integer/real data attributes subject to arithmetic constraints expressed in LIRA. Static DB schemas employ an unusual functional approach, but they are able to describe the most sophisticated notion of read-only database schemas considered in the literature (Deutsch, Li, and Vianu 2019; Calvanese et al. 2019a; Ghilardi et al. 2023). Unary functions capture relations with primary and foreign keys. Specifically, the domain of a unary function is an *id sort*, representing object identifiers. Functions sharing the same id sort as domain model the attributes of such objects, which either point to other id sorts (implicitly representing foreign keys), or to so-called *value sorts* that denote primitive datatypes, such as strings or real/integer values. We call DB instances the models of static DB schemas, in line with *knowledge bases* such as *description logic ontologies* (Baader et al. 2003) studied in the KR community.[1]

In the next paragraph we define Σ-DMTs over static DB schemas, where the corresponding $\mathcal{T}$-runs have DB instances as underlying models. In this context, we tackle a verification task that is *parameterized* on the DB instances, i.e., over the instances of the read-only DB. In order to do that, we rely on a sort of *Open World Assumption* (OWA), because the DB is not fixed apriori, but can change among the possible parameters. Nevertheless, every DB instance is a model of the DB theory, and every run of Σ-DMT operates over one such model (fixed throughout the entire run). For this reason, a query of the Σ-DMT is then answered *in a (single) run* through its evaluation over that fixed model, hence obeying to the standard *Closed World Assumption* (CWA) when referring to a fixed DB instance.

Σ-DMT over static DB schemas Let a *DB$_\Sigma$-DMT* be a Σ-DMT where Σ consists of arbitrary relation, and unary function symbols, and let its theory $\mathcal{T}_{db}$ be EUF possibly extended with a set of ground facts to model an initial DB. This theory enjoys model completion (Calvanese et al. 2020b). In a *DB$_\Sigma$-DMT with arithmetic*, Σ supports in addition operations from an arithmetic theory $\mathcal{T}_{ar}$ such as LIRA, and its theory is $\mathcal{T} := \mathcal{T}_{db} \cup \mathcal{T}_{ar}$. Exs. 1 and 3 are examples for DB$_\Sigma$-DMTs with arithmetic. To ensure that the combined

[1]It is certainly of interest to study the setting where only finite models are considered. This is left for future work.

theory has model completion, we assume that the signature is *tame*, which intuitively means that no uninterpreted function symbol has an arithmetic domain (see below).

In general, both the DB and the arithmetic perspective can give rise to infinitely many terms, so in classes of Σ-DMTs where the verification task is decidable, both must be suitably restricted. Below, we explore different ways to do so.

I. Acyclic signature. First, we consider DB_Σ-DMTs without arithmetic. Acyclicity of the signature Σ is defined via the *sort graph* of Σ, which has as node set the sorts $\mathcal{S}$, and an edge from s to s' iff there is a function symbol $f\colon s \to s'$ in Σ (as we have only unary functions); we say that Σ is *acyclic* if the sort graph is so. In this case, only finitely many Σ-terms exist (Abadi, Rabinovich, and Sagiv 2010). Thus, there are also only finitely many non-equivalent quantifier-free formulas over the finite set of variables V, so finitely many history constraints. Hence if Σ is acyclic, every DB_Σ-DMT has a finite data history, so by Cor. 20:

Theorem 21. *The verification task for a DB_Σ-DMT is decidable if Σ is acyclic.*

This generalizes the result about decidability of reachability in SASs over an acyclic signature (Calvanese et al. 2020b, Thm 4.2), as we consider here full LTL_f model checking. An example for a respective process is shown in (Gianola, Montali, and Winkler 2023, Ex. 33).

II. Acyclic signature and monotonicity constraints. Next, we consider DB_Σ-DMTs with arithmetic. As mentioned above, we assume that signature Σ is *tame*, i.e., in the sort graph of Σ, sorts *rat* and *int* are leaves. In this case, the combined theory $\mathcal{T}$ is known to enjoy model completion (Calvanese et al. 2022, Thm. 7). E.g., the signatures in Exs. 1 and 3 are tame. Towards decidability, we require that Σ is acyclic, and moreover restrict arithmetic constraints to *monotonicity constraints* (MCs), i.e., constraints of the form $t \odot t'$ where t, t' are Σ-terms of sort *rat* over free variables V and $\odot$ is one of $=, \neq, \leq$, or $<$. MCs have been repeatedly considered in the verification literature (Demri and D'Souza 2007; Felli, de Leoni, and Montali 2019) and are important in BPM since transition guards of this shape can be learned automatically from data (de Leoni and van der Aalst 2013).

Theorem 22. *Let Σ be acyclic, $\mathcal{B}$ a DB_Σ-DMT with arithmetic and $\psi \in \mathcal{L}_\Sigma$. If all arithmetic constraints in $\mathcal{B}$ and ψ are MCs, the verification task is decidable.*

Proof (idea). We inspect procedure TameCombCover (Calvanese et al. 2022, Sec. 8) to show that it has finitely many possible results of the form $\varphi_1(\overline{X}) \wedge \varphi_2(\overline{Y}, \overline{t}(\overline{X}))$, up to equivalence. Here φ_1 is an EUF formula, $\overline{t}$ is a list of EUF terms, and φ_2 an arithmetic formula. Acyclicity bounds the possibilities for φ_1 and $\overline{t}$, and by results about QE of MCs, there are only finitely many choices for $\varphi_2(\overline{Y}, \overline{t}(\overline{X}))$(Felli, Montali, and Winkler 2022, Thm. 5.2). $\square$

For instance, Thm. 22 applies to Ex. 3, and explains why the product automaton in Ex. 17 is finite. Thm. 22 strictly generalizes (Felli, Montali, and Winkler 2022, Thm. 5.2) as it supports arithmetic *and* EUF.

III. Local Finiteness. Next, we consider decidability beyond acyclic signatures for DB_Σ-DMT without arithmetic, using the concept of local finiteness (Lipparini 1982). A Σ-theory $\mathcal{T}$ is called *k-locally finite*, for some $k \in \mathbb{N}$, if for every finite set of variables X, the number of Σ-terms with free variables X is upper-bounded by k, up to $\mathcal{T}$-equivalence. As local finiteness directly ensures that there are only finitely many non-equivalent quantifier-free formulas in $\mathcal{T}^*$, we get:

Theorem 23. *If $\mathcal{T}$ is a k-locally finite Σ-theory that admits model completion, the $\mathcal{T}$-verification task is decidable for any DB_Σ-DMT $\mathcal{B}$ and Σ-property ψ.*

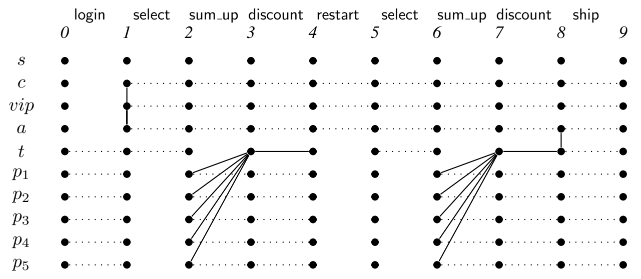
This generalizes the reachability result in (Bojańczyk, Segoufin, and Toruńczyk 2013, Thm. 5) to full LTL_f, since their Fraïssé classes, when first-order definable, coincide with universal, locally finite theories admitting a model completion (Calvanese et al. 2020b). One way to ensure that a model completion exists in Thm. 23 is by restricting axioms in $\mathcal{T}$ to single-variable sentences:

Corollary 24. The $\mathcal{T}$-verification task is decidable for a DB_Σ-DMT $\mathcal{B}$ and Σ-property if $\mathcal{T}$ is a k-locally finite Σ-theory axiomatized by single-variable axioms.

Thm. 23 generalizes Thm. 21 as acyclicity implies local finiteness but not vice versa: consider a signature where f is the only non-constant function symbol and $\mathcal{T} := \{\forall x. f^2(x) = x\}$, which is locally finite but not acyclic. One can also obtain decidability by replacing acyclicity in the combination result of Thm. 22 by local finiteness, using in the proof local finiteness to ensure that up to equivalence there are only finitely many formulas without arithmetic.

IV. Bounded lookback. The above criteria achieve decidability mostly by syntactic restrictions. However, finite data history can also be achieved by controlling interaction of transitions, so that Cor. 20 applies. This is the case for systems with *bounded lookback*, a property intuitively expressing that the behaviour of a DMT depends only on a bounded amount of information from the past (Felli, Montali, and Winkler 2022); it generalizes the feedback freedom property (Damaggio, Deutsch, and Vianu 2012). Bounded lookback is defined via *computation graphs*: Let $\sigma = \langle t_1, \dots, t_n \rangle$ be a transition sequence of a DB_Σ-DMT with arithmetic, and $w = \langle \varsigma_0, \dots, \varsigma_n \rangle \in 2^C$ for $C := cstr(\psi)$. The *computation graph* for σ and w is an undirected graph $G_{\sigma,w}$ with nodes $\mathcal{V} := \{v_i \mid v \in V, 0 \leq i \leq n\}$ and an edge (x_i, y_j) whenever x_i and y_j are in the transitive closure of variable pairs that occur in a common literal of $H(\sigma, w)$, for all $i, j \leq n$. The subgraph $E_{\sigma,w}$ of $G_{\sigma,w}$ consists of all edges (x_i, y_j) s.t. x_i and y_j are in the transitive closure of variable pairs that occur in a common *equality* atom of $H(\sigma, w)$. The graph obtained from $G_{\sigma,w}$ by collapsing equality edges (i.e., edges in $E_{\sigma,w}$) is denoted $[G_{\sigma,w}]$.

Example 25. *Consider the Σ-DMT in Ex. 1 and the transition sequence $\sigma = \langle$*login, select, sum_up, discount, restart, select, sum_up, discount, ship*$\rangle$. For $w = \langle \emptyset, \dots, \emptyset \rangle$, the graph $G_{\sigma,w}$ is shown below. Edges in $E_{\sigma,w}$ are dotted, other edges of $G_{\sigma,w}$ solid. When collapsing equality edges, i.e. in $[G_{\sigma_i,w}]$, the longest path has length 5 (e.g. from c_1 to $p_{1,6}$).*



For $k \geq 0$, $(\mathcal{B}, \psi)$ has *k-bounded lookback* if for all σ and all $w \in 2^C$ s.t. $H(\sigma, w)$ is $\mathcal{T}$-satisfiable, all acyclic paths in $[G_{\sigma,w}]$ have length at most k. E.g., one can check that Ex. 1 has 5-bounded lookback, as variables get reset when the loop repeats, so the dynamics depends only on the last five steps. Also all DMTs without loops have bounded lookback, since the length of computation graphs is bounded. We show (Gianola, Montali, and Winkler 2023):

Theorem 26. *If $\mathcal{B}$ is a DB$_\Sigma$-DMT with arithmetic such that $(\mathcal{B}, \psi)$ has k-bounded lookback for some $k \geq 0$, the verification task is decidable.*

Summary of decidability results. We showed how to reconstruct but also generalize several decidable fragments from the literature: Thm. 22 shows the combination of an acyclic DB signature and MCs to be decidable, which considerably generalizes Felli et al. (2022) as they did not have DBs; Thm. 23 generalizes Bojanczyk et al. (2013) who had no arithmetic and only safety verification; Thm. 21 generalizes Calvanese et al. (2020) from safety to linear-time verification; Thm. 26 advances Damaggio et al. (2012) as bounded lookback is more general than feedback freedom.

A large set of practical processes fit into one of the identified classes. The requirement of an acyclic signature (Thms. 21 and 22) is not very restrictive for database schemas in practice (notice, e.g., that all business processes in the experiments below satisfy this); it is also a practical guideline to avoid cyclic dependencies in DBs if possible. Also, Damaggio et al. (2012) showed that feedback freedom (generalized by bounded lookback) is a property that many real-world web applications/DB systems possess (cf. Thm. 26). Monotonicity constraints are important in BPM as they are mined by automatic data/decision mining techniques (cf. Thm. 22).

Implementation

We implemented our model checking procedure for DB$_\Sigma$-DMTs with arithmetic in the tool LinDMT. Given a property and a DB$_\Sigma$-DMT as input, it visualizes the NFA and product construction, and computes a witness as prescribed by Thm. 16, if it exists. The tool interfaces CVC5 (Deters et al. 2014) for SMT checks, and QE in LIRA. On top of that, we implemented the cover computation procedure for EUF with unary functions and arbitrary relations from (Calvanese et al. 2021, p. 961), and the TameCombCover routine (Calvanese et al. 2022) to compute covers in tame combinations of EUF and LIRA. LinDMT is a command-line tool written in Python, but it is also accessible via a web interface (https://lindmt.unibz.it); sources and benchmarks are available as well. In the input files for Σ-DMTs, control states

		T	D	R/F/C	time	SMT checks
(1)	IV	10	60	0/0/ 9	3.47 / 0.69 / 1.23	2702 / 540 / 880
(2)	IV	44	124	0/0/ 7	14.23/ 2.85 / 3.66	20470/4094/4285
(3)	IV	6	30	0/0/ 7	0.95 / 0.19 / 0.40	186 / 37 / 57
(4)	II	39	257	0/0/10	8.14 / 1.63 / 2.32	3811 / 762 /1342
(5)	IV	19	37	0/0/ 3	1.71 / 0.34 / 0.71	1130 / 226 / 433
(6)	IV	20	88	0/0/ 3	5.43 / 1.09 / 2.40	2816 / 563 /1109
(7)	II	2	9	2/0/ 1	0.31 / 0.06 / 0.11	127 / 25 / 33
(8)	IV	8	52	2/1/ 2	1.36 / 0.27 / 0.60	506 / 101 / 239
(9)	IV	20	194	3/0/ 5	8.18 / 1.64 / 2.91	11363/2272/4067
(10)	IV	20	159	2/1/ 3	54.23/10.85/17.38	17648/3529/4265
(11)	IV	19	196	5/0/ 3	3.30 / 0.66 / 0.93	568 / 113 / 288
(12)	IV	8	106	2/0/ 4	3.03 / 0.61 / 0.79	2319 / 463 / 663
(13)	IV	21	291	2/0/ 7	13.31/ 2.66 / 2.98	8144 /1628/1829
(14)	IV	12	264	1/0/ 4	17.43/ 3.49 / 6.26	2858 / 571 / 804
(15)	IV	16	107	2/6/ 6	2.70 / 0.54 / 0.98	375 / 75 / 171
(16)	IV	16	213	3/0/ 4	3.11 / 0.62 / 0.75	488 / 97 / 138
(17)	II	10	22	2/4/ 5	0.38 / 0.08 / 0.13	134 / 26 / 46
(18)	IV	8	35	1/1/ 7	6.29 / 1.26 / 5.69	560 / 112 / 195

Table 1: Experimental results

(like o_1, o_2 in Ex. 3) are supported for efficiency.

We curated a benchmark set using data-aware business processes from the literature, especially the VERIFAS problem set (Li, Deutsch, and Vianu 2017). We excluded updates of DB relations, which are not supported by DMTs; however, we can verify LTL$_f$ properties of these problems that were beyond the reach of VERIFAS. For each DMT, we checked five different properties. The results are summarized in Tab. 1, listing the decidable class I–IV for each problem, the number of transitions (**T**), total size of transition formulae (**D**), number of relations/functions/constants (**R/F/C**), the total/average/maximal time in seconds for the five properties, and the total/average/maximal number of SMT checks. All problems are in one of the decidable classes, which indicates that these classes indeed cover Σ-DMTs that are relevant in practice. Overall, about 36% of the computation time is spent on SMT checks, and 29% on QE and cover computation. We observed that transition formulae with large depth such as (10) (problem *airline checkin* from the VERIFAS set) cause performance to deteriorate. However, all problems could be solved in less than 20 secs, which we believe indicates feasibility of our approach.

Conclusion

We have tackled linear-time verification for DMTs, a very general model of data-aware processes. Our procedure is semi-decision in general, but becomes a decision procedure for several relevant, checkable settings. We obtained novel decidability criteria especially for processes operating over read-only DBs with arithmetic, generalizing several earlier results. In the future, we plan to study other verification tasks for DMTs, such as branching-time model checking and monitoring. Second, we want to study whether our techniques apply to evolving, read-write DBs, or to description logics (Koopmann 2020; Baader, Ghilardi, and Lutz 2012).

Acknowledgements

A. Gianola was partially supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under projects UIDB/50021/2020 (DOI:10.54499/UIDB/50021/2020). M. Montali and S. Winkler were partially supported by the UNIBZ projects ADAPTERS and TEKE, and the PRIN MIUR project PINPOINT Prot. 2020FNEB27.

References

Abadi, A.; Rabinovich, A.; and Sagiv, M. 2010. Decidable fragments of many-sorted logic. *J. Symb. Comput.*, 45(2): 153–172.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baader, F.; Ghilardi, S.; and Lutz, C. 2012. LTL over description logic axioms. *ACM Trans. Comput. Log.*, 13(3): 21:1–21:32.

Bagheri Hariri, B.; Calvanese, D.; Montali, M.; De Giacomo, G.; De Masellis, R.; and Felli, P. 2013. Description Logic Knowledge and Action Bases. *J. Artif. Intell. Res.*, 46: 651–686.

Baral, C.; and De Giacomo, G. 2015. Knowledge Representation and Reasoning: What's Hot. In *Proc. 29th AAAI*, 4316–4317.

Barrett, C. W.; Sebastiani, R.; Seshia, S. A.; and Tinelli, C. 2021. Satisfiability Modulo Theories. In *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, 1267–1329. IOS Press.

Bojańczyk, M.; Segoufin, L.; and Toruńczyk, S. 2013. Verification of database-driven systems via amalgamation. In *Proc. 32nd PODS*, 63–74.

Borgwardt, S.; Hoffmann, J.; Kovtunova, A.; Krötzsch, M.; Nebel, B.; and Steinmetz, M. 2022. Expressivity of Planning with Horn Description Logic Ontologies. In *Proc. 36th AAAI*, 5503–5511. AAAI Press.

Calvanese, D.; De Giacomo, G.; and Montali, M. 2013. Foundations of data-aware process analysis: a database theory perspective. In *Proc. 32nd PODS*, 1–12.

Calvanese, D.; Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2019a. Formal Modeling and SMT-Based Parameterized Verification of Data-Aware BPMN. In *Proc. 17th BPM*, volume 11675 of *LNCS*, 157–175. Springer.

Calvanese, D.; Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2019b. Model Completeness, Covers and Superposition. In *Proc. 27th CADE*, volume 11716 of *Lecture Notes in Computer Science*, 142–160. Springer.

Calvanese, D.; Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2020a. Combined Covers and Beth Definability. In *Proc. 10th IJCAR*, volume 12166 of *LNCS*, 181–200. Springer.

Calvanese, D.; Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2020b. SMT-based verification of data-aware processes: a model-theoretic approach. *Math. Struct. Comput. Sci.*, 30(3): 271–313.

Calvanese, D.; Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2021. Model Completeness, Uniform Interpolants and Superposition Calculus. *J. Autom. Reason.*, 65(7): 941–969.

Calvanese, D.; Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2022. Combination of Uniform Interpolants via Beth Definability. *J. Automat. Reason.*, 66(3): 409–435.

Calvanese, D.; Gianola, A.; Mazzullo, A.; and Montali, M. 2023. SMT Safety Verification of Ontology-Based Processes. In *Proc. 37th AAAI*, 6271–6279. AAAI Press.

Calvanese, D.; Montali, M.; Patrizi, F.; and Stawowy, M. 2016. Plan Synthesis for Knowledge and Action Bases. In *Proc. 25th IJCAI*, 1022–1029. IJCAI/AAAI Press.

Chang, C.-C.; and Keisler, J. H. 1990. *Model Theory*. Amsterdam-London: North-Holland Publishing Co.

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.

De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In *Proc. 28th AAAI*, 1027–1033.

De Giacomo, G.; Lespérance, Y.; and Patrizi, F. 2016. Bounded situation calculus action theories. *Artif. Intell.*, 237: 172–203.

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

de Leoni, M.; and van der Aalst, W. M. P. 2013. Data-aware process mining: discovering decisions in processes using alignments. In *Proc. 13th SAC*, 1454–1461. ACM.

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.

Deutsch, A.; Hull, R.; Li, Y.; and Vianu, V. 2018. Automatic verification of database-centric systems. *ACM SIGLOG News*, 5(2): 37–56.

Deutsch, A.; Li, Y.; and Vianu, V. 2019. Verification of Hierarchical Artifact Systems. *ACM Trans. Database Syst.*, 44(3): 12:1–12:68.

Felli, P.; de Leoni, M.; and Montali, M. 2019. Soundness Verification of Decision-Aware Process Models with Variable-to-Variable Conditions. In *Proc. 19th ACSD*, 82–91. IEEE.

Felli, P.; Montali, M.; and Winkler, S. 2022. Linear-Time Verification of Data-Aware Dynamic Systems with Arithmetic. In *Proc. 36th AAAI*, 5642–5650. AAAI Press.

Francès, G.; and Geffner, H. 2016. Effective Planning with More Expressive Languages. In *Proc. 25th IJCAI*, 4155–4159. IJCAI/AAAI Press.

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

Ghilardi, S. 2004. Model-Theoretic Methods in Combined Constraint Satisfiability. *J. Automat. Reason.*, 33(3-4): 221–249.

Ghilardi, S.; Gianola, A.; Montali, M.; and Rivkin, A. 2023. Safety Verification and Universal Invariants for Relational Action Bases. In *Proc. 32nd IJCAI*.

Gianola, A. 2023. *Verification of Data-Aware Processes via Satisfiability Modulo Theories*, volume 470 of *LNBIP*. Springer.

Gianola, A.; Montali, M.; and Winkler, S. 2023. Linear-Time Verification of Data-Aware Processes Modulo Theories via Covers and Automata (Extended Version). *CoRR*, abs/2310.12180.

Gulwani, S.; and Musuvathi, M. 2008. Cover Algorithms and Their Combination. In *Proc. 17th ESOP*, volume 4960 of *LNCS*, 193–207.

Koopmann, P. 2020. LETHE: Forgetting and Uniform Interpolation for Expressive Description Logics. *Künstliche Intell.*, 34(3): 381–387.

Li, Y.; Deutsch, A.; and Vianu, V. 2017. VERIFAS: A Practical Verifier for Artifact Systems. *Proc. VLDB Endow.*, 11(3): 283–296.

Lipparini, P. 1982. Locally finite theories with model companion. *Atti della Accademia Nazionale dei Lincei. Classe di Scienze Fisiche, Matematiche e Naturali. Rendiconti, Serie 8*, 72.

Masellis, R. D.; Di Francescomarino, C.; Ghidini, C.; Montali, M.; and Tessaris, S. 2017. Add Data into Business Process Verification: Bridging the Gap between Theory and Practice. In Singh, S.; and Markovitch, S., eds., *Proc. 31st AAAI*, 1091–1099. AAAI Press.

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 Mathematiciens des Pays Slaves*, 92–101.

Reichert, M. 2012. Process and Data: Two Sides of the Same Coin? In *OTM 2012*, volume 7565 of *LNCS*, 2–19.