Automatic Verification of Data-Aware Processes with Arithmetic

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

[1] University of Bologna, Italy
[2] Free University of Bozen-Bolzano, Italy
paolo.felli@unibo.it, montali@inf.unibz.it, winkler@inf.unibz.it

Abstract

Data-aware processes combine structural and behavioral constraints in a single model, and are increasingly studied in business process management and artificial intelligence. In this context, Data Petri nets (DPNs) are gaining popularity due to an attractive tradeoff between simplicity with expressiveness. However, real-world processes often encompasses arithmetic data constraints, which makes even basic properties undecidable. Here we show that by suitably restricting the constraint language or model structure, a number of important tasks becomes decidable for practical classes of systems: This includes linear- and branching-time model checking, strategic reasoning, and verification of data-aware soundness.

Introduction

Integrating control-flow and data aspects to faithfully capture the dynamic evolution of processes is a key problem in AI (Baral and De Giacomo 2015) and business process management (BPM) (Reichert 2012). *Data Petri nets* (DPNs) have recently been developed as a concise yet expressive formalism to capture data-aware processes. In such a model, the control-flow of the process is specified as a Petri net; the data perspective comprises a set of variables that are constrained by read-write conditions attached to net transitions. DPNs are interesting because on the one hand, they allow to formalize complex processes combining control-flow, case data, and decision models (de Leoni, Felli, and Montali 2021); on the other hand, they can be discovered automatically from logs, cf. e.g. (de Leoni, Felli, and Montali 2018).

While the integrated modeling of data and processes is notoriously error-prone, automatic discovery techniques typically come without correctness guarantees, so that verification is crucial. However, automatic verification of DPNs is highly challenging, as the interplay of control-flow and data gives rise to an infinite state space. On top of that, while arithmetic is essential to faithfully model practical applications, it renders even more analysis tasks undecidable, e.g. reachability (Deutsch et al. 2018).

Here we outline how for several relevant DPN classes where the constraint language and/or the structure are suitably restricted, important analysis tasks become decidable. These include linear and branching time model checking, data-aware soundness checking, as well as strategic reasoning. We analyze DPNs for realistic processes from a variety of domains that were presented in the literature, and show that many of these examples actually fall into one of the decidable classes. This extended abstract summarizes recent publications on decidability results for linear- and branching-time model checking, data-aware soundness checking, and strategic reasoning, and provide a link to a prototype implementation.

Data Petri Nets

A *Petri net with data* (DPN) is a Petri net that maintains a fixed, finite set of process variables V of some numeric type, and where actions are associated with guard expressions called *constraints* that can at the same time read and write the variables V. For instance, Fig. 1 shows a DPN that models a simple auction process. Here $V = \{o, t\}$, where variable o holds the last offer, and t is a timer. For instance, the constraint $t^r > 0 \wedge o^w > o^r$ associated with action bid expresses that the current value of t must be positive; and the value of o is increased by this action (i.e., superscript r refers to the read, w to the written value). Here we assume that all constraints are linear arithmetic expressions, but sometimes restrict to the following forms: (i) *monotonicity constraints* (MCs) admit only variable-to-variable/constant comparisons wrt. $=, <, \leq, \neq$ for variables with domain $\mathbb{Q}$ or $\mathbb{R}$, such as $x < y$ or $3 \leq x$; (ii) *integer periodicity constraints* (IPCs) 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 $k, d \in \mathbb{N}$; (iii) *gap-order constraints* (GCs) have the form $x - y \geq k$ for x, y integer variables or constants, and $k \in \mathbb{N}$. For analysis tasks, it is convenient to work on a simpler model. To this end, in place of DPNs we consider *data-aware dynamic systems* (DDSs), which are labeled transition systems where transitions are associated with constraints as above. Every bounded DPN can be transformed into a DDS, by considering all possible markings. The below system in Fig. 1 shows the auction process as a DDS.

Verification and Reasoning

As process executions are typically assumed to be finite, we adopt a finite trace semantics for all the below analysis tasks.

Linear- and branching time model checking. For linear time, we consider an enriched version of LTL, where con-

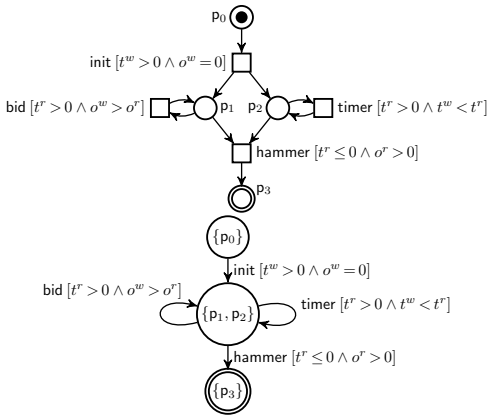

Figure 1: DPN and DDS for simple auction model.

straints over the variables V and the control states of a DDS $\mathcal{B}$ occur as atoms. The verification problem is then to decide whether, given $\mathcal{B}$ and an LTL$_f$ formula ψ, there is a run of $\mathcal{B}$ that satisfies ψ. In (Felli, Montali, and Winkler 2022b), we define an abstract property of $\mathcal{B}$ and ψ called *finite summary*, which guarantees decidability of the verification task. This property holds, e.g., if all constraints are in one of the classes MC, IPC, or GC, but also if certain structural properties hold. Moreover, the finite summary property is *modular*, in the sense that if a DDS $\mathcal{B}$ can be suitably decomposed into subsystems, then $\mathcal{B}$ enjoys the property. E.g., all constraints in the DDS of Fig. 1 are MC; so model checking is decidable with respect to an MC property, like $\psi = \Box(\mathsf{p}_3 \rightarrow o > 0)$.

For branching time, we consider a CTL*-like extension of the above specification language, and the verification task to decide whether the runs of $\mathcal{B}$ satisfy a CTL* formula χ. (More precisely, we aim to find conditions on the initial valuation of $\mathcal{B}$ such that χ is satisfied.) In (Felli, Montali, and Winkler 2022a), we extend the above approach from LTL$_f$ to CTL$^*_f$, and prove decidability for a similar abstract property. E.g., we can check that the DDS in Fig. 1 has deadlocks: $\mathsf{A}\,\mathsf{G}\,\mathsf{E}\,\mathsf{F}\,\mathsf{p}_3$ does not hold as from some states no final state is reachable (namely $\{\mathsf{p}_1, \mathsf{p}_2\}$ with $t \leq 0$).

Soundness Checking. A well-established notion of correctness for dynamic systems is that of *soundness* (van der Aalst 1998), which requires that (i) all activities in the process occur in some execution; (ii) the process can reach a *final* state from every reachable state and (iii) final states are reached in a 'clean' way, without leaving any thread of the process still hanging. For instance, as mentioned above, Fig. 1 violates the second property. In (de Leoni, Felli, and Montali 2018) it is shown that DPNs over variable-to-constant comparisons are a decidable case that is expressive enough to capture data-aware process models with S-FEEL DMN decisions (de Leoni, Felli, and Montali 2021). This result was later extended to full MCs and other systems that satisfy (a restricted form of) the finite summary property in (Felli, Montali, and Winkler 2022c).

Strategic Reasoning. The evolution of systems is often nondeterministic when it comes to resolving decision points with multiple enabled conditions, or to choosing which value to pick when updating a variable. While in reality these sources of nondeterminism are typically controlled by multiple, autonomous actors, verification has almost uniformly adopted the simplifying assumption that these actors cooperate. In (de Leoni, Felli, and Montali 2020) this premise is relaxed, and it is shown how established strategy synthesis approaches for temporal specifications can be combined with faithful data abstraction methods, towards a constructive, readily implementable technique for strategy synthesis in DDSs over MCs.

Conclusion. Important analysis tasks become decidable for DDSs, and hence DPNs, that enjoy the *finite summary property*, cf. (Felli, Montali, and Winkler 2022b,c,a). The resulting decision procedures for linear and branching time verification and soundness checking, are implemented in the tool `ada` (https://ctlstar.adatool.dev). We evaluated `ada` on DPNs from the literature, and found that most examples fall into one of the decidable classes. In future work, we plan to study further decidable classes; monitoring of DPNs, and applicability of these approaches to array-based systems.

References

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

de Leoni, M.; Felli, P.; and Montali, M. 2018. A Holistic Approach for Soundness Verification of Decision-Aware Process Models. In *37th ER*, volume 11157 of *LNCS*, 219–235.

de Leoni, M.; Felli, P.; and Montali, M. 2020. Strategy Synthesis for Data-Aware Dynamic Systems with Multiple Actors. In *Proc. 17th KR*, 315–325.

de Leoni, M.; Felli, P.; and Montali, M. 2021. Integrating BPMN and DMN: Modeling and Analysis. *J. Data Semant.*, 10(1): 165–188.

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

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

Felli, P.; Montali, M.; and Winkler, S. 2022c. Soundness of Data-Aware Processes with Arithmetic Conditions. In *Proc. 34th CAiSE*, volume 13295 of *LNCS*, 389–406.

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

van der Aalst, W. 1998. The Application of Petri Nets to Workflow Management. *J. Circuits Syst. Comput.*, 8(1): 21–66.