Verification of Unary Communicating Datalog Programs (Discussion Paper)

C. Aiswarya[1], Diego Calvanese[2,3,*], Francesco Di Cosmo[2] and Marco Montali[2]

[1]Chennai Mathematical Institute, Chennai, India
[2]Free University of Bozen-Bolzano, Bolzano, Italy
[3]Umeå University, Umeå, Sweden

Abstract

We study verification of reachability properties over Communicating Datalog Programs (CDPs), which are networks of relational nodes connected through unordered channels and running Datalog-like computations. Each node manipulates a local state database (DB), depending on incoming messages and additional input DBs from external services. Decidability of verification for CDPs has so far been established only under boundedness assumptions on the state and channel sizes, showing at the same time undecidability of reachability for unbounded states with only two unary relations or unbounded channels with a single binary relation. The goal of this paper is to study the open case of CDPs with bounded states and unbounded channels, under the assumption that channels carry unary relations only. We discuss the significance of the resulting model and prove the decidability of verification of variants of reachability, captured in fragments of first-order CTL. We do so through a novel reduction to coverability problems in a class of high-level Petri Nets that manipulate unordered data identifiers. We study the tightness of our results, showing that minor generalizations of the considered reachability properties yield undecidability of verification, both for CDPs and the corresponding Petri Net model.

This paper is an abridged version of a paper published in the *Proceedings of the 43rd ACM Symposium on Principles of Database Systems* (PODS 2024).

Keywords

Formal verification, Data-aware processes, Communicating Datalog Programs, Distributed computation, CTL-FO

1. Introduction

Declarative approaches to the specification of distributed data-aware systems have been extensively studied in many different contexts [1, 2, 3, 4, 5]. These approaches share the general idea that the overall behavior of the system emerges from the interaction of a number of local components (hereafter called *nodes*), mutually connected in a given topology, each running a declarative program that describes at once the input/output behavior to exchange messages with the other nodes, and the update of the node internal state. Both the state and the exchanged messages are relational, thus making the overall system a distributed version of so-called *data-aware*

SEBD 2024: 32nd Symposium on Advanced Database Systems, June 23–26, 2024, Villasimius, Sardinia, Italy
*Corresponding author.
aiswarya@cmi.ac.in (C. Aiswarya); diego.calvanese@unibz.ir (D. Calvanese); francesco.dicosmo@unibz.it (F. Di Cosmo); marco.montali@unibz.it (M. Montali)
0000-0002-4878-7581 (C. Aiswarya); 0000-0001-5174-9693 (D. Calvanese); 0000-0002-5692-5681 (F. Di Cosmo); 0000-0002-8021-3430 (M. Montali)

processes, extensively studied within the foundations of data management from the modelling and static analysis point of view [6, 7, 8, 9, 10].

In this work, we are interested in the static analysis of such distributed declarative data-aware processes, in the style of [6, 7, 8]. We focus in particular on the D2C language originally introduced in [4], which employs a suitably extended version of Datalog equipped with communication primitives and the possibility of referring to the previous and current node state. On top of the resulting model of what we call *Communicating Datalog Programs (CDPs)*, two aspects become particularly important in the light of static analysis: *(i)* the presence of communication channels with different properties on faithfulness and ordering; *(ii)* the distinction between closed systems where new data are never created, but only the data present in the initial node states can be used and exchanged, and interactive systems where new data can be acquired and exchanged during the computation.

Declarative distributed systems with asynchronous communication occurring over multiset channels (where multiple copies of the same message may exist, even when the sender and receiver coincide) were considered in seminal works in the area, but only studied in connection with static analysis in the presence of external data in [11]. Such systems are infinite-state, with the consequence that even for very simple reachability properties, static analysis is undecidable [11]. Decidable subclasses have been singled out by importing and adapting the notion of *state-boundedness* originally introduced in [12, 13, 14], and applied in [13, 15, 16] to obtain decidability of verification of data-aware processes against rich variants of first-order branching-time temporal logics. In a state-bounded system, infinitely many objects may be seen within and across runs of the system, but in each single configuration reached during the computation, their number remains bounded. In the context of CDPs, this notion has a twofold effect: it essentially bounds the number of constants that can be simultaneously stored in each node state, as well as the size of each communication channel. Under such restrictions, it has been shown that model checking first-order CTL properties is decidable [11].

In this work, we start from the observation that bounding communication channels is a severe restriction, as it cannot be enforced even by suitably controlling how nodes are programmed. At the same time, [11] has shown that even propositional reachability is undecidable to check over severely restricted CDPs that employ messages with a binary signature. We consequently focus on the verification of *unary CDPs*, i.e., CDPs where the messages range over a signature of at most *unary* relational symbols, and while the local memory and interaction with external services is bounded, the channel capacity is *not*. This is also interesting to study in the light of multiset channels, since adopting queues, as in [9, 10], would immediately yield undecidability for unary, unbounded channels. We show that the resulting model is still powerful enough to model real-case scenarios, and engage in a fine-grained study of CDP verification against variants of reachability, expressed as fragments of first-order CTL. Specifically, we establish an equivalence between this problem and that of verifying coverability over a variant of Petri nets with unordered data [17], a property that is decidable to check despite the fact that these nets are essentially infinite-state. This yields decidability for positive nested reachability queries over unary CDPs, even in the case where the logic has not only the ability of querying the states, but also that of inspecting communication channels. We finally investigate the tightness of our decidability results, showing that minor generalizations fall back into undecidability.

2. The CDP Model

In this section, we informally introduce the CDP model by Ma et al. [4]. However, for simplicity, we formalize only the fragment relevant for our study, which is the one over unordered channels, non-deterministic bounded inputs, and single-node networks.

A CDP is a fixed network of data-centric nodes sharing messages via point-to-point channels. Each node *(1)* runs a Datalog-like program, written in the language D2C, *(2)* updates its internal state, which is maintained as a *state DB* over a dedicated *state signature*, *(3)* receives information from the external environment, in the form of an *input DB* over a dedicated *input signature*, and *(4)* shares *messages*, i.e., single relational facts over a dedicated *transport signature*. The nodes react to incoming messages: when a message m from a node u is delivered to a node v, the latter gets activated and runs the program on its data-sources. In fact, the program input consists of the node state DB, the current input DB, the message m itself, and the local structure of the network at v, in the form of a *network DB*. The output provides a new state DB for v and a set of outgoing messages, each labeled by its recipient, which, in turn, are sent on the respective channels (without labels).

We assume that communication is asynchronous and channels are reliable but unordered, that is, at each time-step, only one message is delivered (and, thus, only one node gets activated), no message can be lost, but the reception order is non-deterministic. These assumptions are useful, e.g., to model communication networks where message loss is ignored but order cannot be guaranteed (e.g., because of an underlying UDP transport protocol). Since nodes react only to incoming messages, the communication network has, for each node, a self-loop channel (from the node to itself), which initially contains a special message dedicated to node activation.

CDP nodes are exposed to information from the external environment, which represents users and/or external services. Environment interaction is abstracted away by input policies, i.e., rules to provide a new input DB. In this paper, we focus on the b-bounded interactive input policies, where $b \in \mathbb{N}$: each time a node receives a message, the current input DB is substituted by a non-deterministically chosen new one with active domain of cardinality at most b. This policy is relevant to model interaction with external users that continuously provide new information, e.g., text messages for a chat application.

All these information sources are manipulated by a D2C program, i.e., a set of Datalog-like rules specialized to the interactive and distributed setting of CDPs. The specialization is achieved by *(1)* organizing relation symbols in dedicated signatures (state, input, and transport), *(2)* using the in-rule flag `prev` to distinguish queries over the previous state DB and the new one under computation, and *(3)* labeling transport literals with terms representing senders or recipients (see [11] for a non-deterministic extension of D2C). However, in this paper, we focus on a simple D2C fragment, specialized for single-node networks. In fact, while inconvenient for modelling, single-node CDPs are enough for the technical study of verification of CDPs employing unordered channels, since each such CDP can be encoded over a single node network.

2.1. Single-node CDPs

We now formalize bounded-interactive, single-node CDPs. With a slight abuse, we refer to this fragment as, simply, CDPs and ignore all other CDP variants (see [11] for the full model).

Definition 2.1. A *CDP signature* is a tuple $\Lambda = (\mathcal{S}, \mathcal{I}, \mathcal{T})$, where $\mathcal{S}$, $\mathcal{I}$, and $\mathcal{T}$ are pairwise disjoint relational signatures, respectively called *state*, *input*, and *transport*. A *state*, *input*, or *transport atom* (resp., *literal*) is an atom (resp., literal) over $\mathcal{S}$, $\mathcal{I}$, or $\mathcal{T}$, respectively. ◁

CDP programs are, syntactically, reminiscent of stratified Datalog with negation and inequality.

Definition 2.2. A *D2C rule* over a CDP signature Λ is a formula

$$H \textit{ if } L_1, \ldots, L_n \textit{ prev } L_{n+1}, \ldots, L_{n+m}, C_1, \ldots, C_h.$$

s.t.: *(1)* H is a state or transport literal, *(2)* $L_1, \ldots, L_n$ are state, input, or transport literals, *(3)* $L_{n+1}, \ldots, L_{n+m}$ are state literals, and *(4)* $C_1, \ldots, C_h$ are inequality constraints of the form $t_1 \neq t_2$, where t_1 and t_2 are terms (constants or variables). The rule *head* is H and the rule *body* is the part following *if*. The rule *scope of* `prev` is $L_{n+1}, \ldots, L_{n+m}$. The rule is *safe* if each variable occurring in the head, in a negated literal, or in an inequality constraint, also occurs in a positive literal in the rule body. The rule is *transport consistent* if each variable occurring in a transport atom in the head also occurs in a positive non-input literal. ◁

Intuitively, in the scope of `prev`, state literals query the state DB available immediately before node activation. Outside the scope of `prev`, in the body, input literals query the input DB, transport literals the incoming message, and state literals the new state DB under computation. In the head, transport atoms deduce the outgoing messages and state atoms the facts in the new state DB. At the end of the computation, the new state DB substitutes the previous one and the outgoing messages are sent on the channel. Transport consistency states that data from the input DB cannot directly flow to the channel. This matches with the assumption that only nodes have the power to send messages, which have to be preliminarily gathered in an out-buffer that contributes to the node configuration (state DB, possibly affecting its boundedness, cf. Def. 2.4). Note that state literals in the scope of prev and transport literals in the body are not involved in forming recursive dependencies. While this feature appears as a major difference with Datalog, actually, it is just a matter of making the syntax convenient for the CDP semantics. In fact, one can provide a Datalog encoding $\overline{\mathcal{P}}$ of a set $\mathcal{P}$ of D2C rules where this difference is ironed out.

Definition 2.3. A *D2C program* $\mathcal{P}$ over a CDP signature Λ is a finite set of safe and transport consistent D2C rules s.t. $\overline{\mathcal{P}}$ is stratified. Given such Λ and $\mathcal{P}$, a *CDP* is a tuple $(\Lambda, \mathcal{P}, S_0, m_0)$, where S_0 is a state DB denoting the initial state and m_0 is an initial message. ◁

The semantics of CDPs is given in terms of configuration graphs, which connect CDP configurations via transitions. Each *configuration* (S, I, C) describes a snapshot of the system, including the state DB S, the input DB I, and the channel C, represented as a multiset. The configuration is *b-input*, *s-state*, or *c-channel bounded*, for some $b, s, c \in \mathbb{N}$, if the cardinality of the active domain of I, of the active domain of S, or of C, is at most b, s, or c, respectively.

Definition 2.4. Given $b, s, c \in \mathbb{N}$, we call *b-CDP* a CDP D interpreted under the configuration graph Υ^b consisting of all b-input bounded configurations of D. A b-CDP is s-state or c-channel bounded if all configurations reachable in Υ^b from an initial configuration are s-state or c-channel bounded, respectively. ◁

3. The Verification Problem for CDPs

We study the problem of formal verification of CDPs. Previous work [11] showed that control-state reachability (that is, whether there is an initial configuration from which the target state DB is reachable — ignoring the configuration of the channel) is undecidable even for restricted CDPs that *(i)* have a single-node network, *(ii)* use the channel solely to (re)activate the node, and *(iii)* employ a unary state signature. Decidability can be gained by imposing boundedness conditions on the various CDP data sources [11]. In fact, for state- and channel-bounded CDPs, decidability holds for temporal model checking againts formulae in CTL_{CDP}, a branching-time logic mixing CTL operators to analyze the system evolution, and FOL to query the data sources.

Unfortunately, boundedness is a semantic property, undecidable to check. In addition, while there are different techniques to enforce state boundedness [14, 18], the same does not hold for channels. Furthermore, as pointed out in the introduction, imposing boundedness is particularly restrictive for communication channels. Interestingly, while undecidability of control-state reachability over state-unbounded CDPs already holds for unary signatures, in the case of CDPs with bounded states and unbounded channels it has been proved only for binary transport signatures [11]. This makes CDPs that are state- and input-bounded, but operate over unbounded channels carrying unary messages, worth investigating. We call such CDPs *unary CDPs* (uCDPs).

In the following, we study the problem of model checking variants of uCDPs against selected fragments of CTL_{CDP}. The base-level fragment we use to express reachability-like properties called $\mathsf{EF}(-, bool, st)$, essentially, mixes EF CTL temporal operators with closed FO formulas over the state signature. Specifically, given a CDP $D = (\Lambda, \mathcal{P}, S_0, m_0)$, where $\Lambda = (\mathcal{S}, \mathcal{I}, \mathcal{T})$, the language $\mathsf{EF}(-, bool, st)$ over D is defined by the rules

$$\Phi ::= \varphi \mid \mathsf{EF}\varphi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \qquad \varphi ::= t_1 = t_2 \mid S(\mathbf{t}) \mid \exists x.\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi,$$

where Φ are temporal formulas, φ are closed FO formulas over $\mathcal{S}$, t_1 and t_2 are terms, and $\mathbf{t}$ is a tuple (of proper size) of terms. Such formulas are interpreted as follows: $S(\mathbf{t})$ queries whether $S(\mathbf{t})$ is in the current state DB; $\exists x$ is an existential quantifier over the active domain of the state DB and the support of the multiset channel; and $\mathsf{EF}\varphi$ is interpreted as in standard CTL, i.e., there exists a path, in the CDP configuration graph, on which φ eventually holds [19]. For example, reachability of a state DB containing the fact $S(\mathtt{a})$ is expressed by the formula $\mathsf{EF}S(\mathtt{a})$, while reachability of the state that contains only that fact by $\mathsf{EF}(S(\mathtt{a}) \wedge \neg\exists x.S(x) \wedge \neg x = \mathtt{a})$.

We study variants of reachability properties starting from $\mathsf{EF}(-, bool, st)$ and considering formulae consisting of (positive boolean combinations of) sentences starting with an EF operator, tuning them along three dimensions: the available temporal operators beyond EF, the presence of negations in the FO queries, and which components they inspect (state DB or also channels)[1]. We identify such fragments with notation $\mathsf{EF}(\square, \triangle, \circ)$, where:

- $\square$ indicates which temporal operators can be nested; it can be one of:
  - $-$ (no nesting), as in the grammar above,
  - EF^+ (nesting of multiple EF), obtained by adding to the grammar rule $\Phi ::= \mathsf{EF}\Phi$,
  - AG (nesting of a single AG), obtained by adding to the grammar rule $\Phi ::= \mathsf{EFAG}\varphi$, or

[1]Input DBs are disregarded since their evolution is completely non-deterministic and its interaction with the state DB can be captured by slightly modifying the CDP program so as to include the bounded input DB in the state DB.

  - $(\mathsf{EF}, \mathsf{AX})^+$ (nesting of multiple EF and AX, possibly interleaved), obtained by adding to the grammar rule $\Phi ::= \mathsf{EF}\Phi \mid \mathsf{AX}\Phi$;
- $\triangle$ indicates how negation is supported by FO formulas; it can be either
  - $bool$, as defined by the grammar above, or
  - pos (no negation), obtained by dropping from the grammar rule $\varphi ::= \neg\varphi$;
- $\circ$ indicates whether formulas can only query state DBs, or also channels; it can be either:
  - st (queries only over node states), as defined by the grammar above, or
  - $st{+}ch$ (queries also over the support of channel multisets), obtained by adding to the grammar rule $\varphi ::= T(\mathbf{t})$, where T is in the transport signature.

For the formal syntax and semantics of these languages, we refer to the full paper.

We study the following model-checking problem variants.

Problem 3.1 ($\mathsf{EF}[\square, \triangle, \circ]$-MC). Let $\square \in \{-, \mathsf{EF}^+, \mathsf{AG}, (\mathsf{EF}, \mathsf{AX})^+\}$, $\triangle \in \{pos, bool\}$, and $\circ \in \{st, st{+}ch\}$. The $\mathsf{EF}[\square, \triangle, \circ]$-MC problem is defined as follows:

Input: A $b \in \mathbb{N}$, $s \in \mathbb{N}$, s-state bounded single-node uCDP D, initial configuration $\mathcal{C}_0$, and closed formula $\Phi \in \mathsf{EF}(\square, \triangle, \circ)$.

Output: Whether the configuration graph Υ^b satisfies Φ from $\mathcal{C}_0$. ◁

Verification w.r.t. all initial configurations reduces to finitely many instances of $\mathsf{EF}[\square, \triangle, \circ]$-MC. Indeed, due to state and input boundedness, the initial configurations are finitely many up to isomorphisms, and FO formulas are invariant under isomorphisms that fix the constants in them. Establishing the decidability status of the different variants of this problem is challenging, due to the subtle interplay of the CDP components, e.g., how the node state is affected by the content of the multiset channel, whose access is limited by aysnchronous communication. To attack this problem, we provide a bridge with models and techniques for the verification of data-aware extensions of PNs, in particular ρ-PNs. PNs are one of the most widely studied models for concurrent computations, particularly suited to handle asynchronous threads and message passing. Specifically, ρ-PNs lend themselves to be connected to uCDPs. In fact, tokens carrying single data elements match constants used in unary messages, and places match unary relation symbols - so that inserting a token carrying constant c in a place M naturally corresponds to having message $\mathtt{M(c)}$ in the channel. What is not at all clear, instead, is how to encode in the ρ-PN the infinitely many input and state DBs that may be encountered along a computation. Recall, in fact, that even under state-boundedness, a CDP can encounter infinitely many, genuinely distinct state DBs.

To address this issue, we represent state and input DBs up to isomorphism. This can be done by introducing dedicated places for the following purposes: *(1)* to encode the relation symbols of messages; *(2)* to represent the isomorphism types of bounded input and state DBs, over a fixed representative bounded domain; *(3)* to specify a mapping from the representative domain to the infinite domain of data values used to form input and state DBs; *(4)* to deal with the special constants that are distinguished in the CDP program, ensuring that each one of those forms a singleton isomorphism type. This constitutes the basis for reducing $\mathsf{EF}[\square, \triangle, \circ]$-MC problems over uCDPs, to coverability checks over ρ-PNs.

We proceed as follows. We first investigate the decidability status of variants of control-state reachability for ρ-PNs (Sec. 4). We then transfer these results to uCDPs, showing reductions from variants of uCDP model checking to ρ-PNs control-state reachability (Sec. 5).

4. ρ-PN Verification

We introduce now the language P-ρCTL to express coverability properties on ρ-PNs and study the decidability of the related model-checking problem. P-ρCTL features the CTL EF temporal operator, boolean conjunctions and disjunctions, and replaces propositions with markings interpreted, each on its own, up to isomorphisms.

Definition 4.1. Given a set P of places, *P-ρCTL* is the language of formulas φ defined by the following grammar, where the *atomic P-ρCTL formulas* are markings M over the place set P:

$$\varphi \ ::= \ M \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{EF}\varphi \qquad \lhd$$

The semantics of P-ρCTL is defined as for CTL, with the provision that the current marking M of a ρ-PN N satisfies an atomic formula M', if M covers, up to isomorphisms, M'.

Problem 4.2 (P-ρCTL-MC). The P-ρCTL-MC problem is defined as follows:
Input: A ρ-PN $N = (P, T, F)$, marking M_0, and P-ρCTL formula φ.
Output: Whether N satisfies φ from M_0, denoted by $N, M_0 \models \varphi$. $\lhd$

Since atomic formulas perform coverability checks, P-ρCTL-MC can be reduced to plain ρ-PN coverability. This is done by induction on the structure of the P-ρCTL formula. First, a given formula φ, to be checked on a ρ-PN N and initial marking M_0, is represented as a syntax tree. Its leafs are the occurrences of atomic formulas and the other nodes are obtained by applying to the children the corresponding boolean or temporal operator. Second, from leafs to the root, each node ψ is mapped to a ρ-PN N^ψ and initial marking M_0^ψ where *(1)* the net N^ψ contains, as sub-nets, the nets N^τ, for each sub-formula τ of ψ, *(2)* P^ψ contains the places $check^\psi$ and $cover^\psi$, *(3)* M_0^ψ places at least a distinguished identifier on $check^\psi$, and *(4)* transitions are added so that the place $cover^\psi$ can be marked with a distinguished identifier iff $N^\psi, M_0^\psi \models \psi$. The construction for non-leaves take into account the children nets and the semantics of the respective conjunction, disjunction, or EF operator, where the latter case is the most involved one. We refer to the full paper for the details.

From decidability of ρ-PN coverability we obtain:

Theorem 4.3. *For each finite place set P, P-ρCTL-MC is decidable.*

5. uCDP Model Checking

To reduce uCDP model checking to ρ-PN model checking, we encode an arbitrary s-state bounded b-uCDP $D = (\Lambda, \mathcal{P}, S_0, m_0)$ with $\Lambda = (\mathcal{S}, \mathcal{I}, \mathcal{T})$, into a ρ-PN $N = (P, T, F)$.

1. *Configuration encoding.* We use identifiers to represent the domain of DBs: $ID = \Delta \cup \{\bullet\}$. We use places of N (and related markings) to encode configurations of D, reorganized in the following way: *(i)* channel configuration, *(ii)* extension, up to isomorphisms, of the state and input DBs, over a fixed active domain of representative constants, *(iii)* mapping, via a partial function, of the representative constants to the represented state and input DB constants.

2. *Step encoding.* We use the transitions of N to encode the steps of D, organized as: *(i)* input update, and *(ii)* D2C computation, including message reception, casting, and state DB update.
3. *Formula encoding.* We encode a formula $\varphi \in \mathsf{EF}(\mathsf{EF}^+, bool, st)$ into P-ρCTL. In the full paper, we show that it is enough to handle FO formulas. Since φ ranges only over the state signature, it is satisfied by any configuration (S, I, C) s.t. $\varphi \models S$. Since D is s-state bounded, up to isomorphism there is only a finite set $\{S_1, \ldots, S_k\}$ of state DBs satisfying φ. Thus, φ is equivalent to the disjunction of markings m_i encoding the configurations $(S_i, \varnothing, \varnothing)$, for each $i \in \{1, \ldots, k\}$. Similarly, we can encode also formulas ψ from $\mathsf{EF}(\mathsf{EF}^+, pos, st{+}ch)$. In fact, since ψ is positive, it can only assert the existence of a fixed number of tuples in the state or in the channel. Thus, because of state boundedness, up to isomorphisms and channel inclusion there is only a finite set $\{(S_1, \varnothing, C_1), \ldots, (S_k, \varnothing, C_k)\}$ of configurations satisfying ψ. Thus, ψ is equivalent to the disjunction of markings m_i encoding the configurations $(S_i, \varnothing, C_i)$, for each $i \in \{1, \ldots, k\}$.

Theorem 5.1. $\mathsf{EF}[\mathsf{EF}^+, bool, st]$-MC *and* $\mathsf{EF}[\mathsf{EF}^+, pos, st{+}ch]$-MC *are decidable.*

Theorem 5.1 is essentially tight, as shown by the next result.

Theorem 5.2. $\mathsf{EF}[\mathsf{AG}, pos, st]$-MC, $\mathsf{EF}[\mathsf{AX}^2, pos, st]$-MC, *and* $\mathsf{EF}[-, bool, st{+}ch]$-MC *are undecidable.*

6. Conclusions

We recap decidability (D) and undecidability (U) for $\mathsf{EF}[\Box, \triangle, \circ]$-MC on uCDPs:

	$\Box$	$-$	$-$	EF^+	EF^+	AG	AX^2
$\circ$	$\triangle$	pos	$bool$	pos	$bool$	pos	pos
st		D	D	D	D	U	U
$st{+}ch$		D	U	D	U	U	U

The properties refer to branching-time first-order properties verified over CDPs where nodes asynchronously exchange messages with at most a unary signature, the state of each node has a bounded size, while communication channels have unbounded capacity. Our results are obtained by encoding such verification problems into corresponding coverability verification problems over ρ-PNs. In spite of the extremely high complexity in the analysis of such nets, several effective techniques and tools for the (symbolic) exploration of their state space exist (see, e.g., [20, 21]). Our work consequently paves the way towards the application of such techniques to the practical analysis of declarative distributed systems.

Acknowledgments

This research has been partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the Province of Bolzano and DFG through the project D2G2 (DFG grant n. 500249124), and by the HEU project CyclOps (under GA n. 101135513).

References

[1] J. M. Hellerstein, The declarative imperative: Experiences and conjectures in distributed logic, SIGMOD Record 39 (2010) 5–19. doi:10.1145/1860702.1860704.

[2] P. Alvaro, T. J. Ameloot, J. M. Hellerstein, W. Marczak, J. Van den Bussche, A Declarative Semantics for Dedalus, Technical Report UCB/EECS-2011-120, EECS Department, University of California, Berkeley, 2011. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-120.html.

[3] T. J. Ameloot, J. Van den Bussche, Positive Dedalus programs tolerate non-causality, J. of Computer and System Sciences 80 (2014) 1191–1213. doi:10.1016/j.jcss.2014.01.005.

[4] J. Ma, F. Le, A. Russo, J. Lobo, Declarative framework for specification, simulation and analysis of distributed applications, IEEE Trans. on Knowledge and Data Engineering 28 (2016) 1489–1502. doi:10.1109/TKDE.2016.2515604.

[5] S. Abiteboul, É. Antoine, G. Miklau, J. Stoyanovich, J. Testard, Rule-based application development using Webdamlog, in: Proc. of the 34th ACM Int. Conf. on Management of Data (SIGMOD), ACM, 2013, pp. 965–968. doi:10.1145/2463676.2465251.

[6] V. Vianu, Automatic verification of database-driven systems: a new frontier, in: Proc. of the 12th Int. Conf. on Database Theory (ICDT), 2009, pp. 1–13. doi:10.1145/1514894.1514896.

[7] D. Calvanese, G. De Giacomo, M. Montali, Foundations of data-aware process analysis: a database theory perspective, in: Proc. of the 32nd ACM Symp. on Principles of Database Systems (PODS), ACM, 2013, pp. 1–12. doi:10.1145/2463664.2467796.

[8] A. Deutsch, R. Hull, Y. Li, V. Vianu, Automatic verification of database-centric systems, ACM SIGLOG News 5 (2018) 37–56. doi:10.1145/3212019.3212025.

[9] A. Deutsch, L. Sui, V. Vianu, D. Zhou, Verification of communicating data-driven web services, in: Proc. of the 25th ACM Symp. on Principles of Database Systems (PODS), 2006, pp. 90–99. doi:10.1145/1142351.1142364.

[10] A. Deutsch, L. Sui, V. Vianu, Specification and verification of data-driven web applications, J. of Computer and System Sciences 73 (2007) 442–474. doi:10.1016/J.JCSS.2006.10.006.

[11] D. Calvanese, F. Di Cosmo, J. Lobo, M. Montali, Convergence verification of declarative distributed systems, in: Proc. of the 36th Italian Conf. on Computational Logic (CILC), volume 3002 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 62–76.

[12] F. Belardinelli, A. Lomuscio, F. Patrizi, Verification of deployed artifact systems via data abstraction, in: Proc. of the 9th Int. Joint Conf. on Service Oriented Computing (ICSOC), volume 7084 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 142–156. doi:10.1007/978-3-642-25535-9_10.

[13] B. Bagheri Hariri, D. Calvanese, G. De Giacomo, A. Deutsch, M. Montali, Verification of relational data-centric dynamic systems with external services, in: Proc. of the 32nd ACM Symp. on Principles of Database Systems (PODS), 2013, pp. 163–174. doi:10.1145/2463664.2465221.

[14] B. Bagheri Hariri, D. Calvanese, M. Montali, A. Deutsch, State-boundedness in data-aware dynamic systems, in: Proc. of the 14th Int. Conf. on Principles of Knowledge Representation

and Reasoning (KR), AAAI Press, 2014, pp. 458–467.
[15] G. De Giacomo, Y. Lespérance, F. Patrizi, Bounded Situation Calculus action theories, Artificial Intelligence 237 (2016) 172–203. doi:10.1016/j.artint.2016.04.006.
[16] D. Calvanese, G. De Giacomo, M. Montali, F. Patrizi, First-order mu-calculus over generic transition systems and applications to the Situation Calculus, Information and Computation 259 (2018) 328–347. doi:10.1016/j.ic.2017.08.007.
[17] F. Rosa-Velardo, Ordinal recursive complexity of Unordered Data Nets, Information and Computation 254 (2017) 41–58. doi:10.1016/J.IC.2017.02.002.
[18] D. Solomakhin, M. Montali, S. Tessaris, R. De Masellis, Verification of artifact-centric systems: Decidability and modeling issues, in: Proc. of the 11th Int. Joint Conf. on Service Oriented Computing (ICSOC), volume 8274 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 252–266. doi:10.1007/978-3-642-45005-1_18.
[19] C. Baier, J.-P. Katoen, Principles of Model Checking, The MIT Press, 2008.
[20] M. Westergaard, S. Evangelista, L. M. Kristensen, ASAP: an extensible platform for state space analysis, in: Proc. of the 30th Int. Conf. on Application and Theory of Petri Nets (PETRI NETS), volume 5606 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 303–312. doi:10.1007/978-3-642-02424-5_18.
[21] S. Ghilardi, A. Gianola, M. Montali, A. Rivkin, Petri net-based object-centric processes with read-only data, Information Systems 107 (2022) 102011. doi:10.1016/J.IS.2022.102011.