DOI: 10.15514/ISPRAS-2025-37(4)-19

Relaxed Lazy Soundness Verification for Data Petri Nets

N.M. Suvorov, ORCID: 0000-0003-2871-9757 <nmsuvorov@hse.ru>
I.A. Lomazova, ORCID: 0000-0002-9420-3751 <ilomazova@hse.ru>
HSE University,
20, Myasnitskaya Ulitsa, Moscow, 101000, Russia.

Abstract. To represent a model that includes both data and resource perspectives, Data Petri nets could be used. In this formalism, each transition has a constraint that includes input and output conditions on variables. To stay within decidability, the conditions should not contain arithmetic operations, so the resources are usually represented as separate places. Existing correctness criteria such as easy, relaxed and lazy soundness could be adapted to resource-oriented Data Petri nets but deciding them requires solving a reachability problem that is known to have very high complexity even for classical Petri nets. In this paper, we propose a new correctness notion called relaxed lazy soundness that incorporates the main features of the aforementioned properties and that could be decided as a coverability problem, which is known to be less computationally complex than the reachability one. We provide an algorithm to verify this property, prove its correctness, and implement it in the existing soundness verification toolkit. The performance evaluation results confirm the applicability of the algorithm to process models of a moderate size. The algorithm could be used both for verification of resource-oriented models and for preliminary validation of arbitrary process models represented as Data Petri nets.

Keywords: data Petri net; verification of distributed processes with data; soundness verification; transition systems.

For citation: Suvorov N.M., Lomazova I.A. Relaxed Lazy Soundness Verification for Data Petri nets. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 4, part 2, 2025, pp. 69-84. DOI: 10.15514/ISPRAS-2025-37(4)-19.

Acknowledgements. This study has been supported by the Basic Research Program at HSE University, Russia.

Проверка ослабленной ленивой бездефектности для сетей Петри с данными

Н.М. Суворов, ORCID: 0000-0003-2871-9757 <nmsuvorov@hse.ru>
И.А. Ломазова, ORCID: 0000-0002-9420-3751 <ilomazova@hse.ru>
Национальный исследовательский университет «Высшая школа экономики», Россия, 101000, г. Москва, ул. Мясницкая, д. 20.

Аннотация. Для представления модели, включающей как данные, так и ресурсы, можно использовать сети Петри с данными. Каждому переходу в таких сетях сопоставлено ограничение, включающее условия на входные и выходные значения переменных. Условия на входные значения определяют правила, при которых переход может сработать. Условия на выходные значения определяют правила, согласно которым осуществляется изменение значений переменных при срабатывании перехода. Чтобы оставаться в границах разрешимости, условия не должны содержать арифметические операции, ввиду чего ресурсы обычно представляются как отдельные позиции. Существующие критерии бездефектности, такие как простая, ослабленная и ленивая бездефектность, могут быть адаптированы к ресурсно-ориентированным сетям Петри с данными, но для их вычисления требуется решение проблемы достижимости, имеющей высокую вычислительную сложность даже для классических сетей Петри. В этой статье мы предлагаем новое свойство бездефектности, называемое ослабленной ленивой бездефектностью, которое включает в себя основные характеристики вышеупомянутых свойств и которое может быть определено путем решения проблемы покрытия, имеющей меньшую вычислительную сложность чем проблема достижимости. Мы представляем алгоритм для проверки данного свойства, доказываем его корректность и реализуем его в существующем наборе инструментов проверки бездефектности. Результаты оценки производительности подтверждают применимость алгоритма для верификации моделей среднего размера. Алгоритм можно использовать как для проверки ресурсно-ориентированных моделей, так и для предварительной проверки произвольных моделей, представленных сетями Петри с данными.

Ключевые слова: сети Петри с данными; верификация распределенных процессов с данными; верификация бездефектности; системы переходов.

Для цитирования: Суворов Н.М., Ломазова И.А. Проверка ослабленной ленивой бездефектности для сетей Петри с данными. Труды ИСП РАН, том 37, вып. 4, часть 2, 2025 г., стр. 69–84 (на английском языке). DOI: 10.15514/ISPRAS–2025–37(4)–19.

Благодарности. Данное исследование поддержано Программой фундаментальных исследований Национального исследовательского университета «Высшая Школа Экономики», Россия.

1. Introduction

A business process consists of a set of activities that are performed in a coordinated manner in an organizational and technical environment and allow a business goal to be achieved [1]. Analyzing business process models at the design stage allows us to identify inconsistencies and potential errors before implementing the process. Our paper focuses on a specific correctness criterion for process models, called soundness. Different types of soundness properties have been defined for business processes. The most well-known are classical [2], weak [3], generalized [4], relaxed [5], and lazy [6] soundness. The paper [7] presents a classification of the main soundness properties proposed for process models.

A classical approach to verifying soundness of a process model is to represent it as a Petri net and check the soundness property for this representation. However, in data-aware process models, where data influence the execution of the process, classical Petri nets are not appropriate. In such cases, different extensions of Petri nets are used, one of which is Data Petri nets (DPNs) [8], which we consider in this paper. Data Petri Nets extend place/transition nets with variables, adding guards that depend on the values of these variables and allowing transitions to update them. Consequently, each

state in a DPN is represented as a pair that includes a marking and a variable valuation (values assigned to all model variables).

DPNs may incorporate both data and resource perspectives. The resources, as in classical Petri nets, could be represented as individual places that may be unbounded and contain extra tokens in a final state (since any DPN with arithmetic conditions is Turing complete, as was proved in [9], it is a reasonable choice to represent resources as places but not as variables). For DPNs, an algorithm to verify the classical soundness (whether a process always terminates properly and whether each process activity occurs in at least one of its executions) was proposed in [10]. However, this property is too strong, making the model incorrect unless it is bounded and does not allow for additional tokens in the final state. This makes this definition not suitable for checking correctness of resource-oriented DPNs.

An example of a resource-oriented DPN is shown in Fig. 1. Here, the model represents a gambling process and contains a resource place p_3 that stores game tokens obtained by a gambler. The idea of gambling is simple: A gambler rolls a number from 0 to 100 and based on this number, he/she may win or lose tokens or leave them unchanged. A gambler may quit the game whenever he/she wants. The initial number of tokens is 3. This DPN is unbounded, since p_3 may store any number of game tokens. Thus, it is not sound according to the definition of classical soundness proposed in [9]. However, the model can be considered correct for representing a real gambling process. Note that the following properties hold for this model: (i) it has at most one token in the sink place o, (ii) there are some number of feasible executions that end at o (albeit with possibly several tokens left in p_3), (iii) each transition occurs in at least one feasible execution. This means that the model is relaxed [5] and lazy [6] sound.

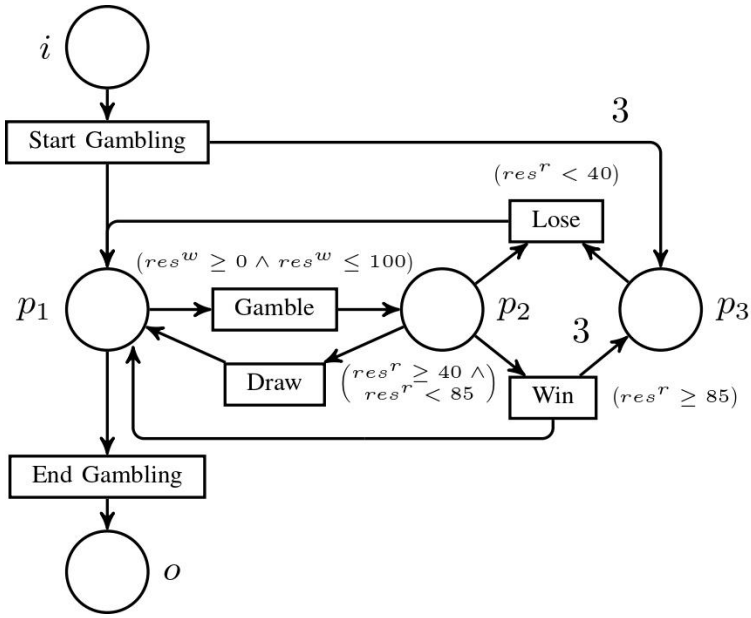

Fig. 1. Resource-oriented DPN $\mathcal{N}$ representing a process of gambling. p_3 is a resource place representing game tokens.

Deciding relaxed and lazy soundness properties requires solving a reachability problem [7,11] that is known to have high time complexity even for classical Petri nets (a lower bound is a tower of exponentials [12]). Thus, these soundness properties can barely be checked in practice even for small models. For DPNs with variables of infinite domains, the task becomes even more complex. In this paper, we propose a new soundness property called relaxed lazy soundness, which combines the main features of relaxed and lazy soundness and deciding which can be reduced to solving a

coverability problem, which is significantly less computationally complex that the reachability one. Our property captures a possibility of termination, support for lazy activities [6], and a requirement of participation of all activities in process executions. Compared with lazy soundness, our property allows models to have deadlocks (as in relaxed soundness). Compared with relaxed soundness, our property supports lazy activities (as in lazy soundness). We propose an algorithm to verify relaxed lazy soundness, which, according to our experimental evaluation, can be used in practice for moderately sized process models.

2. Related Works

Much effort has been put into validating data-aware process models, both in terms of model checking and soundness verification. The approaches differ not only in the algorithms themselves but also in the formalisms used to represent such process models.

The main current standardized approach to represent data-aware process models is to use the Business Process Model and Notation (BPMN) together with the Decision Model and Notation (DMN) to achieve the separation of concerns while representing both control and data flows [13]. Some authors propose other BPMN extensions, such as BPMN models with conditions on arcs [14]. However, due to the complexity and ambiguity of the BPMN notation, simpler mathematical formalisms are usually used to represent data-aware process models and verify their correctness. A classical formal representation of a business process is a workflow net [15]. A workflow net is a Petri net that has a single source place (a place without input arcs), denoted i, and a single sink place (a place without output arcs), denoted o, where each node is on a path from i to o. Representing data-aware process models using workflow nets is a cumbersome task, as shown in [16]. The reason is that, for each variable value, a separate place should be generally added. This makes this approach applicable only for small process models with rather limited domains of variable values, as for larger models or domains such construction results in huge models that cannot be analyzed either manually or automatically.

Another approach is to use Petri net extensions. The authors in [17] propose a conceptual workflow model extended with data operations (WFD-net), where each model state is represented by a marking and a subset of satisfied guards. Here, the activities actually read/write entire guards, which may be not granular enough to represent real process models [18]. Paper [19] introduces a workflow model extended with SQL operations on tables (WFT-net) designed to overcome this limitation. In [20], this formalism is extended by adding domain constraints that must hold in each state of the model. Each state of the proposed formalism is represented as a tuple that includes a marking, a state of data elements, a state of tables, and a subset of satisfied guards. The number of elements that represent a state makes the state sufficiently difficult to perceive, while the data perspective appears rather abstract.

Our paper focuses on Data Petri nets proposed in [8]. In DPNs, the data perspective is represented by variable valuations, while each transition is accompanied by a constraint that includes input and output conditions on variables. Compared with WFD and WFT nets, most of the behavioral properties of DPNs are generally undecidable. Thus, to consider correctness properties, such as soundness, the authors usually consider a subclass of DPNs, where each variable is real-typed and conditions are composed of variable-operator-constant and variable-operator-variable atoms.

Regarding soundness properties, data-aware process models are verified mainly against classical soundness (for example, all soundness verification algorithms for DPNs, namely [18,21,22], check only classical soundness). Paper [7] proves that a Petri net is classical sound *if and only if* its closure (a workflow net, where a transition is added from i to o) is live (from any reachable marking it is possible to enable any transition) and bounded. However, in many scenarios, weaker correctness criteria are determined for models. For instance, in resource-oriented models, model unboundedness is not a sign of incorrectness and, thus, checking them against classical soundness is not meaningful. For such cases, weaker notions of soundness have been proposed. Let us consider some of them.

Probably, the most well-known property after classical soundness is *weak soundness* [3]. A model is weakly sound if and only if the process always properly terminates (i.e., leads to marking $[o]$). This notion allows dead transitions, which are never executed, but still requires the model to be bounded. For cases when it is important not to have dead transitions in the model, *relaxed soundness* [5] can be applied. A model is relaxed sound if and only if each model activity is present in some execution that leads to the final state. A relaxed sound model may be unbounded and may include deadlocks and livelocks. In resource-oriented models, it is often the case that proper termination (having a single token in o) is not required. This is the case where *lazy soundness* [6] can be applied: a process model is lazy sound if and only if the process always 'adequately' terminates. Here, the 'adequate' termination means reaching a marking with a single token in o and any number of tokens in other places, whereas markings having more than one token in o should not be reachable. Lazy soundness requires an `adequate' termination from each reachable state, and thus deadlocks and livelocks are not allowed. Some authors also add other notions of soundness: for instance, [9] proposes the property (which they call 'relaxed soundness') that only requires the model to contain at least one proper execution. The last three soundness properties fit the case of resource-oriented models (indeed, all of them hold for the model from Fig. 1), but deciding each of them requires solving a reachability problem, which may take an unreasonable amount of time for unbounded models even of small sizes.

For DPNs, the first attempt to verify soundness was made in [10]. Here, the authors propose to convert a DPN into a colored Petri net (CPN) and then verify the soundness of the CPN using existing techniques. The authors restrict the DPN to contain only variable-operator-constant conditions. In all subsequent works, namely [21-23], the authors construct a finite abstraction of the state space, called either a Constraint Graph or a Labeled Transition System (LTS), to verify the classical soundness. Since, as proved in [22], such an abstraction of the state space is not sufficient by itself to verify the soundness property, various special techniques have been developed to verify soundness. In [23], the authors make an LTS as granular as possible by considering a graph node as a pair that includes a marking and the smallest set of variable assignments that can be described by a combination of atomic formulas present in the net. In [21], the authors construct a separate abstraction for each reachable DPN marking and then combine them. [22] modifies the original DPN by splitting transitions that belong to cycles and adding silent transitions in such a way that the resulting DPN is still equivalent to the original one, but its LTS can now be directly used to verify soundness.

All of the above algorithms are designed to verify the classical soundness. As mentioned in [23], they can also be used to verify weak soundness by skipping the step of detecting dead transitions. Checking classical and weak soundness can be done quick enough for small and medium-sized models as both of these properties require model boundedness. If a model is unbounded, solving a reachability problem becomes a significantly more complex task [24]. However, our investigation has shown that the property that combines the main features of relaxed and lazy soundness could be decided by a single coverability graph construction, which makes verification of this combined property a viable option for checking correctness of resource-oriented DPNs.

3. Data Petri Nets

A Data Petri net is a place/transition net that includes transition constraints, represented as logical expressions, that define input and output conditions over the data variables.

Each constraint φ over a set X of variables is an expression of the form

$$\varphi := \top \mid x \odot y \mid x \odot c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2,$$

where: (i) $\top$ is the logical "true"; (ii) $x, y \in X$; (iii) $c \in R$; (iv) $\odot \in \{<, =, >\}$.

By $\Phi(X)$, we denote the language of constraints. For example, for $X = \{y, z\}$, expressions $y > z$, $z < 2$ and $(z > 1) \vee ((z < 2) \wedge (y = 1))$ are in $\Phi(X)$.

Let X be a set of variables. A constraint $\varphi \in \Phi(X)$ is satisfied by an assignment $\theta: X \to \mathbb{R}$, written $\theta \models \varphi$, according to the following definitions:

- $\theta \models x \odot c$ if and only if $\theta(x)$ is defined, and $\theta(x) \odot c$ is true;
- $\theta \models x \odot y$ if and only if both $\theta(x)$ and $\theta(y)$ are defined, and $\theta(x) \odot \theta(y)$ is true;
- $\theta \models \neg\varphi$ if and only if $\theta \not\models \varphi$;
- $\theta \models \varphi_1 \wedge \varphi_2$ if and only if $\theta \models \varphi_1$ and $\theta \models \varphi_2$.

By $[[\varphi]]$, we denote the set of all assignments that satisfy $\varphi \in \Phi(X)$. We say that two formulas $\varphi_1, \varphi_2 \in \Phi(X)$ are logically equivalent (denoted $\varphi_1 \sim \varphi_2$)if and only if $[[\varphi_1]] = [[\varphi_2]]$.

Let V be some set of variables. Since in DPNs, variable values may be changed by a transition firing, for each variable $v \in V$, we use variables v^r (r – for 'read') and v^w (w – for 'write') to address its input and output (w.r.t. transition firings) values. Define $V^r \doteq \{v^r | v \in V\}$ and $V\hat{}w \doteq \{v^w | v \in V\}$. Then each transition constraint in a DPN is an expression in $\Phi(V^r \cup V^w)$.

Now we can define a DPN:

Definition 1 (Data Petri net). A *data Petri net* (DPN) is a tuple $\mathcal{N} = \langle P, T, F, V, guard \rangle$, where:

- P and T are disjoint sets of places and transitions, respectively;
- $F: (P \times T) \cup (T \times P) \to N$ is a flow relation;
- V is a finite set of variables;
- $guard: T \to \Phi(V^r \cup V^w)$ is a guard assignment function that labels transitions with arithmetic constraints.

Given $t \in T$, we also define $read(t)$ and $write(t)$ to denote, respectively, the set of variables V^r and V^w that occur in $guard(t)$. In this work, we consider DPNs with distinguished input (denoted i) and output (denoted o) places.

A state of a DPN $\mathcal{N}$ is a pair (M, α), where

- $M: P \to N$ is a marking function that assigns a number of tokens to each place $p \in P_{\mathcal{N}}$, and
- $\alpha: V \to \mathbb{R}$ is a variable valuation function that assigns a value to each variable in V.

We use $A_{\mathcal{N}}$ to denote the set of all possible variable valuations in $\mathcal{N}$ and $\mathcal{M}_{\mathcal{N}}$ to denote the set of all markings in $\mathcal{N}$. Given two markings M' and M'' of a DPN $\mathcal{N}$, we write $M'' \geq M'$if and only if for all $p \in P_{\mathcal{N}}$, we have $M''(p) \geq M'(p)$, and we write $M'' > M'$if and only if $M'' \geqslant M'$ and there exists $p \in P_{\mathcal{N}}$ s.t. $M''(p) > M'(p)$.

A DPN changes from one state to another by firing transitions. Given a DPN $\mathcal{N}$ and a state (M, α), we say that transition $t \in T$ may fire at (M, α) yielding a new state (M', α')if and only if:

- $M(p) \geq F(p, t)$ and $M'(p) = M(p) - F(p, t) + F(t, p)$, for all $p \in P$;
- $\beta \models guard(t)$, where $\beta: V^r \cup V^w \to \mathbb{R}$ and, for every $v \in V$, it holds that $\beta(v^r) = \alpha(v)$ and $\beta(v^w) = \alpha'(v)$;
- $\alpha(v) = \alpha'(v)$, for every $v \in V$ such that $v^w \neq write(t)$.

We denote transition firing as $(M, \alpha)[t\rangle(M', \alpha')$.

This is naturally extended to finite sequences of transition firings $\sigma = t_1 \cdots t_n$, called traces, while each trace induces a run denoted as $(M_0, \alpha_0)[t_1\rangle \ldots [t_n\rangle(M_n, \alpha_n)$ (or, equivalently, as $(M_0, \alpha_0)[\sigma\rangle(M_n, \alpha_n)$). Given two states (M, α) and (M', α'), we write $(M, \alpha)[*\rangle(M', \alpha')$ to denote zero or more transition firings leading from (M, α) to (M', α'). By (M_I, α_I) we denote the initial DPN state. As we consider resource-oriented nets, we assume that $M_I \geqslant [i]$ and $M_I[i] = 1$. For DPN $\mathcal{N}$ with initial state (M_I, α_I) we define runs of $\mathcal{N}$ and traces of $\mathcal{N}$ as the set of runs and traces as above, of any length, such that $(M_I, \alpha_I)[\sigma\rangle(M, \alpha)$ for some marking M and variable valuation α.

Consider DPN $\mathcal{N}$ in Fig. 1. For this net, $M_I = [i]$ and $\alpha_I(res) = 0$. Firing *Start Gambling* consumes a token from i and produces one token in p_1 and three tokens in p_3. If now *End Gambling* fires, the process terminates. If *Gamble* fires, it transfers a token to p_2 and sets a value from 0 to 100 to variable *res*, based on which the choice between *Lose*, *Draw*, and *Win* is done on the next step. Note that *Lose* consumes one token from p_3, Win produces three tokens in p_3, and *Draw* does not change the number of tokens in p_3. Firing any of these transitions adds a token to p_1 and another gambling attempt can be made. The process terminates when *End Gambling* fires or when a gambler loses all his game tokens.

Definition 2 (Reachability set, reachability graph). Let $\mathcal{N}$ be a DPN with an initial state (M_I, α_I). The reachability set of $\mathcal{N}$, denoted by $Reach_{\mathcal{N}}$, is the smallest set of states, which is inductively defined as follows:

- $(M_I, \alpha_I) \in Reach_{\mathcal{N}}$;
- if $(M, \alpha)[t\rangle(M', \alpha')$ for $t \in T$ and $(M, \alpha) \in Reach_{\mathcal{N}}$, then $(M', \alpha') \in Reach_{\mathcal{N}}$.

The reachability graph of $\mathcal{N}$, denoted as $RG_{\mathcal{N}}$, is a graph $\langle V, E\rangle$, where:

- $V = Reach_{\mathcal{N}}$ is the set of reachable states of $\mathcal{N}$;
- $E \subseteq V \times T \times V$ is the set of edges such that $(v, t, v') \in E$ if and only if $v[t\rangle v'$, for some $t \in T$.

Fig. 2 shows a fragment of $RG_{\mathcal{N}}$ for $\mathcal{N}$ from Fig. 1. The fragment includes an execution that leads to the situation, when all tokens are lost (a path from s_0 to s_{10}), and a feasible execution that includes all DPN activities and ends in o (a path from s_0 to s_{16}). A special case where a gambler starts and immediately ends gambling is also added (a path from s_0 to s_2).

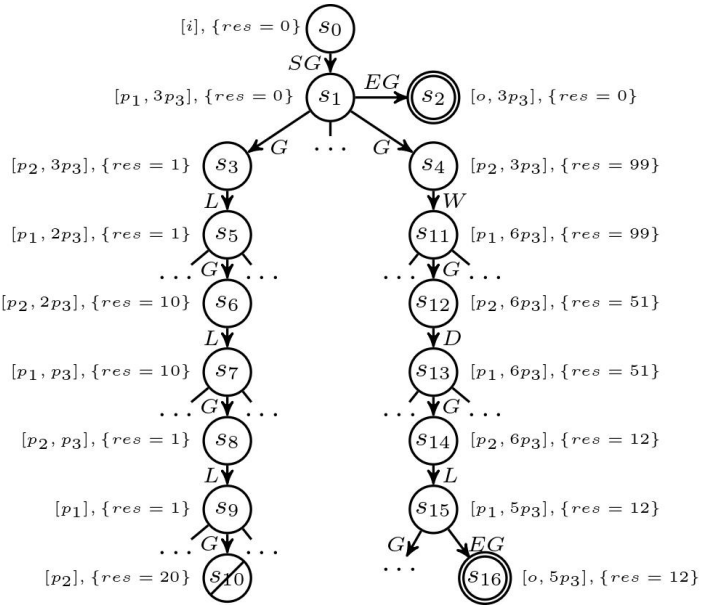

Fig. 2. A fragment of the reachability graph for $\mathcal{N}$ from Fig. 1. Arcs are labeled with the initial letters of the transition names. Square brackets denote markings. Curly brackets denote variable valuations. Double circles denote final nodes. Forbidden signs denote deadlocks.

3.1 Relaxed Lazy Soundness

We pose relaxed lazy soundness as an intersection of relaxed and lazy soundness properties. Consequently, a model is relaxed lazy sound if and only if (i) in each reachable state, there is no

more than one token in o, (ii) it contains some feasible executions that end at $M \succeq [o]$, and (iii) each model activity is present in at least one feasible execution. We formally define relaxed lazy soundness for DPNs as follows:

Definition 3. Let $\mathcal{N}$ be a DPN with initial state (M_I, α_I) and distinguished sink place o. Let $\mathcal{M}_F = \{M_F | M_F \in \mathcal{M}_\mathcal{N} \wedge M_F(o) = 1 \wedge M_F \succeq [o]\}$. $\mathcal{N}$ is relaxed lazy sound if and only if the following properties hold:

1) for each $M \in \mathcal{M}_\mathcal{N}$, $M(o) \leq 1$;
2) for each $t \in T$, there exist $(M, \alpha), (M', \alpha') \in Reach_\mathcal{N}$ and $M_F \in \mathcal{M}_F$, such that $(M, \alpha)[t\rangle(M', \alpha')$ and $(M', \alpha')[*\rangle(M_F, \alpha_F)$ for some α_F.

The first condition states that place o should be bounded and contain no more than one token. The second condition verifies that each DPN transition may fire leading to a state from which the sink place is reachable (potentially, with some other tokens in the net). DPN $\mathcal{N}$ from Fig. 1 is relaxed lazy sound, since both these conditions hold, although the DPN is unbounded and can enter a deadlock if a gambler always loses. Compared with lazy-only and relaxed-only soundness, relaxed lazy soundness can potentially be used for a wider range of resource-oriented models, since it captures the basic properties that should typically hold for such models.

4. Verification Algorithm

In this section, we introduce the algorithm to verify relaxed lazy soundness of a DPN. The algorithm is based on constructing and investigating a state space abstraction called an abstract coverability graph (ACG) that we define in the next subsection.

4.1 State Space Abstraction

An ACG of a DPN is a generalization of a classical coverability graph such that each node represents not a single state but a set of states having the same marking but different variable valuations. Compared with a classical coverability graph, the abstract version is always finite for the DPN setting that we consider in this paper (real-typed variables, variable-operator-constant/variable-operator-variable conditions).

To represent a set of variable valuations, we use the language of constraints defined in the previous section. In [22], we have proved that any set of variable valuations in DPN abstract state space structures can be described by some formula in $\Phi(V)$. The formula of a new state is then computed using operator $\oplus$ defined in [18]. Given transition constraint $guard(t)$ and node formula $\varphi_n \in \Phi(V)$, $[[\varphi_n \oplus guard(t)]]$ is the union of all possible variable valuations that can be obtained by firing t at any variable valuation from $[[\varphi_n]]$. The result of this operation is computed using the concept of quantifier elimination, which is decidable for real arithmetic [25], but may be undecidable for other domains.

To define an ACG, we first define a coverability relation. Let (M, φ), (M', φ') be two nodes. We say that (M', φ') covers (resp., strictly covers) (M, φ), denoted as $(M, \varphi) \sqsubseteq (M', \varphi')$ (resp., $(M, \varphi) \sqsubset (M', \varphi')$), if and only if $[[\varphi]] = [[\varphi']]$ and $M \preceq M'$ (resp., $M \prec M'$). To operate with unbounded nets, we use the special symbol ω, as in [22], which represents an unbounded number of tokens. For each integer n, $\omega > n$, $\omega \pm n = \omega$ and $\omega \geq \omega$. Now we can define an ACG:

Definition 4 (Abstract Coverability Graph). Let $\mathcal{N} = \langle P, T, F, V, guard\rangle$ be a DPN with initial state (M_I, α_I). Let $\Phi(V)$ be the language of constraints, as in Section 3. Abstract Coverability Graph $ACG_\mathcal{N}$ of $\mathcal{N}$ is a tuple $\langle S, E, s_0\rangle$, where:

- $s_0 = (M_I, \phi_I) \in S$ is the initial node with $\phi_I = \bigwedge_{v \in V}\{v = \alpha_I(v)\}$;
- $S \subseteq \mathcal{M}_N \times \Phi(V)$ is the least set that contains s_0 and is closed under the transition relation.
- $E \subseteq S \times T \times S$ is a set of arcs labeled with transitions, s.t. $\big((M, \phi), t, (M', \phi')\big) \in E$ if and only if:

- $\phi' = \phi \oplus guard(t)$ and $[[\phi]] \neq \emptyset$.
- for each $p \in P$, $M(p) \geq F(p,t)$.
- given $M^*(p) = M(p) - F(p,t) + F(t,p)$, $M'(p) = \omega$ if there exists a node $(M'', \phi'') \in S_{CG}$ along the path from s_0 to (M, ϕ), s.t. $(M'', \phi'') \sqsubset (M^*, \phi')$, and $M^*(p) > M''(p)$, otherwise $M'(p) = M^*(p)$.

If $(s,t,s') \in E$, we say that (s,t,s') is a transition firing in $ACG_{\mathcal{N}}$. We denote a transition firing by writing $s[t\rangle s'$. We extend this definition to sequences $\sigma = \langle t_1, \ldots, t_n \rangle$ of n transition firings, called traces, and denote the corresponding run by $s_0[t_1\rangle s_1[t_2\rangle \ldots [t_n\rangle s_n$ or equivalently by $s_0[\sigma\rangle s_n$. Fig. 3 illustrates an ACG constructed for DPN $\mathcal{N}$ from Fig. 1.

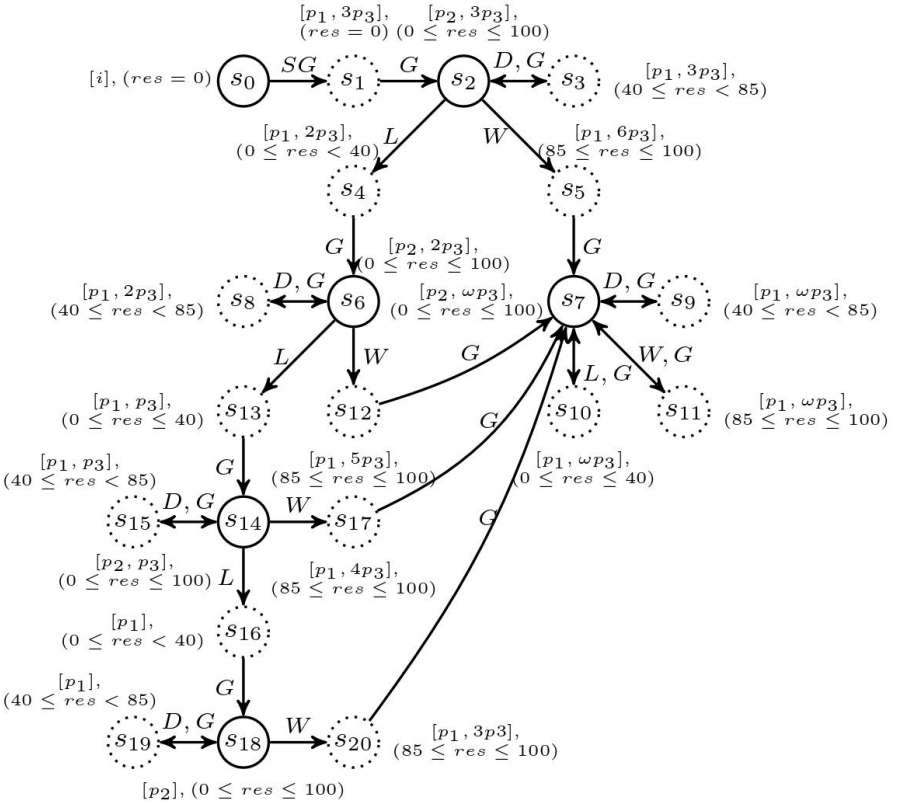

Fig. 3. An abstract coverability graph for $\mathcal{N}$ from Fig. 1. Arcs are labeled with the initial letters of the transition names. Square brackets denote markings. Dotted circles denote nodes from which it is possible to fire End Gambling and reach o (nodes with a token in o are omitted for brevity).

Proposition 1. Let $\mathcal{N} = \langle P, T, F, V, guard \rangle$ be a DPN with initial state (M_I, α_I) and $RG_{\mathcal{N}}$ be a reachability graph of $\mathcal{N}$. Let $ACG_{\mathcal{N}}$ be an abstract coverability graph of $\mathcal{N}$. Then, $(M_I, \phi_I)[\sigma\rangle(M, \phi)$ is in $ACG_{\mathcal{N}}$ if for each $\alpha \in [[\phi]]$, $(M_I, \alpha_I)[\sigma\rangle(M^*, \alpha)$ is in $RG_{\mathcal{N}}$, where $M^*(p) = M(p)$ if $M(p) \neq \omega$.

Proof. The statement is a corollary to Definition 4 (specifically, from the consequent application of the $\oplus$ operator and from the marking construction that is based on the coverability relation). □

Proposition 2. Let $\mathcal{N} = \langle P, T, F, V, guard \rangle$ be a DPN with initial state (M_I, α_I) and $RG_{\mathcal{N}}$ be the reachability graph of $\mathcal{N}$. Let $ACG_{\mathcal{N}}$ be the abstract coverability graph of $\mathcal{N}$. If run $(M_I, \phi_I)[\sigma\rangle(M, \phi)$ is in $ACG_{\mathcal{N}}$, then for each $\alpha \in [[\phi]]$, there exists run $(M_I, \alpha_I)[\sigma^*\rangle(M^*, \alpha)$ in

$RG_{\mathcal{N}}$, where the set of transitions in σ^* coincides with the set of transitions in σ, and $M(p) = M^*(p)$ if $M(p) \neq \omega$.

Proof. If run $(M_I, \phi_I)[\sigma\rangle(M, \phi)$ is in $ACG_{\mathcal{N}}$ and for each p, $M(p) \neq \omega$, then for each $\alpha \in [[\phi]]$, $RG_{\mathcal{N}}$ includes run $(M_I, \alpha_I)[\sigma\rangle(M, \alpha)$ (follows from Definition 4). Assume that a run in $ACG_{\mathcal{N}}$ends with (M, ϕ), where for some p, $M(p) = \omega$. Then, trace σ may not be executable in $RG_{\mathcal{N}}$ (due to the nature of ω-abstraction). Let us represent σ as a sequence $\sigma_1, t_1, \ldots, t_{n-1}, \sigma_n$, so that the execution of each σ-part in $ACG_{\mathcal{N}}$preserves the same set of places with ω in all the markings, while transitions $t_1, \ldots, t_{n-1}$ transform some places to ω-places.

Consider a base case. Let $(M_I, \phi_I)[\sigma_1\rangle(M_1, \phi_1)$ be a run in $ACG_{\mathcal{N}}$. For brevity, we further denote (M, A) as a set of DPN states (M, α), where $\alpha \in A$. Let (M_1, A_1) be a set of states resulting from executing σ_1 from (M_I, α_I) in $RG_{\mathcal{N}}$. By Definition 4, $A_1 = [[\phi_1]]$. Let $(M_1, \phi_1)[t_1\rangle(M_1', \phi_1')$ be a transition firing in $ACG_{\mathcal{N}}$. Then, there exists a sequence of transition firings in $\mathcal{N}$ that includes only transitions from σ_1 and transition t_1 and that can be executed any number of times by adding tokens in places for which $M_1'(p) = \omega$. From each node (M_1, α_1) with $\alpha_1 \in A_1$, execute t and this sequence of transitions as many times as necessary to obtain the number of tokens in each $p \in P$, where $M_1'(p) = \omega$, greater than any fixed but arbitrary natural number. Following Definition 4, there must exist a set of runs that include only transitions from σ_1 and transition t_1 and reach $(M_1'^*, \alpha_1')$, where $M_1'(p) = M_1'^*(p)$ if $M_1'(p) \neq \omega$, so that the union of such $(M_1'^*, \alpha_1')$ can be described as $(M_1'^*, A_1')$ with $A_1' = [[\phi_1']]$.

Induction step. Let for $k < n$, it is true. Then, we have run $(M_k', \phi_k')[\sigma_{k+1}\rangle$ (M_{k+1}, ϕ_{k+1}) in $ACG_{\mathcal{N}}$ and there exists a set of states $(M_k'^*, A_k')$ such that $A_k' = [[\phi_k']]$ and $M_k'(p) = M_k'^*(p)$ if $M_k'(p) \neq \omega$. Since for each $p \in P$, where $M_k'(p) = \omega$, we have a number of tokens greater than any fixed but arbitrary natural number, trace σ_{k+1} can be executed on $(M_k'^*, A_k')$ leading to the set of states (M_{k+1}^*, A_{k+1}), where $A_{k+1} = [[\phi_{k+1}]]$ and $M_{k+1}(p) = M_{k+1}^*(p)$ if $M_{k+1}(p) \neq \omega$. If σ_{k+1} is the last part of σ, the proposition is proved. Otherwise, there must exist transition firing $(M_{k+1}, \phi_{k+1})[t_{k+1}\rangle(M_{k+1}', \phi_{k+1}')$ in $ACG_{\mathcal{N}}$. Then, there exists a sequence of transitions that consists of transitions from $\sigma_1, \ldots, \sigma_{k+1}$, transitions $t_1, \ldots, t_{k+1}$, that ends with t_{k+1}, and that can be executed any number of times putting tokens in places, for which $M_{k+1}'(p) = \omega$ and $M_{k+1}(p) \neq \omega$. From (M_{k+1}^*, A_{k+1}), let execute t_{k+1} and this sequence of transitions as many times as necessary to obtain the number of tokens in each $p \in P$, where $M_{k+1}'(p) = \omega$ and $M_{k+1}(p) \neq \omega$, greater than any fixed but arbitrary natural number. After that, we obtain the set of states $(M_{k+1}'^*, A_{k+1}')$, such that $A_{k+1}' = [[\phi_{k+1}']]$ and $M_{k+1}'(p) = M_{k+1}'^*(p)$ if $M_{k+1}'(p) \neq \omega$. □

Note that it is impossible to tell from a coverability graph alone whether the net contains deadlocks and/or livelocks. This was proved in [24] for classical coverability graphs, and since an ACG is a generalization of this state-transition structure, the same limitation holds for ACG. For bounded DPNs, the problem of detecting deadlocks and livelocks is known to be decidable and was solved in [18,22,23]. For unbounded DPNs, there is currently no solution for this task. Note that the task of detecting deadlocks is challenging even for classical Petri nets. Although deadlock- and livelock-freedom can be reduced to the reachability problem that is known to be decidable [11], existing works, such as [24-25], allow one to verify the absence of deadlocks only for rather limited classes of nets (e.g., [25] considers nets with only one unbounded place). For similar reasons, the coverability graphs alone cannot be used to check whether the sink place is reachable in a 'clean way' (when $M = [o]$). Since coverability graphs allow ω-markings, we cannot determine whether it is possible to consume all the tokens from ω-places and thus reach the output place with no remaining tokens. There are some workarounds for this, but most of them use other representations of business processes, such as π-calculus in [28].

Nevertheless, coverability graphs could be used to verify some important properties that should usually hold for resource-oriented models. It is possible to check whether the model could terminate having one token in o and potentially extra tokens in the net (we could call this as 'adequate' termination), whether each model transition can actually fire and be present in executions with

'adequate' termination, and whether all the model executions do not put more than one token in o. These properties could be seen as a combination of main relaxed and lazy soundness features, and all of them are actually included in the proposed above the relaxed lazy soundness property.

4.2 Checking Relaxed Lazy Soundness

The procedure of checking relaxed lazy soundness is based on constructing and studying the ACG. Algorithm 1 illustrates the procedure of constructing an ACG according to Definition 4. Sets S and E represent the nodes and arcs of $ACG_{\mathcal{N}}$, s_0 – the initial node of $ACG_{\mathcal{N}}$. Set N stores the nodes that need to be expanded. For each node in $\mathcal{N}$, we try to fire DPN transitions for the marking and variable valuations it represents (lines 6-11). If some transition may fire, we try to find preceding strictly covering nodes (lines 12-13). If such nodes are found, we define places where the number of tokens should be replaced with ω (lines 13-16). If the resulting node already exists in the graph, we add an arc to this node from the current one; otherwise, a new node is added to the graph and to the set of nodes to be expanded (lines 17-20).

Regarding the implementation of the algorithm, the simplest approach is to construct the graph using a depth-first search and use a stack to represent set N. For each new node in the graph, we can determine the set of parents using the information obtained from the previous node, and thus skip the graph traversal procedure when checking node coverage. As for checking the equivalence of formulas, we can perform it by checking the satisfiability: to check $\phi \sim \phi'$, we can verify whether $\phi \wedge \neg\phi' \vee \neg\phi \wedge \phi'$ is satisfiable (if not, then $\phi \sim \phi'$ is true) using known satisfiability modulo theories (SMT) solvers such as Z3 [29]. Note that for our constraint language, satisfiability is decidable, as well as the quantifier elimination needed to compute the result of the $\oplus$-procedure [22].

Algorithm 1 *ConstructACG*$(\mathcal{N}, (M_I, \alpha_I))$

Input: A DPN $\mathcal{N} = \langle P, T, F, V, guard\rangle$ with initial state (M_I, α_I).
Result: Abstract Coverability Graph of $\mathcal{N}$.

$\phi_I \leftarrow \bigwedge_{v\in V}\{v = \alpha_I(v)\}$
$s_0 \leftarrow (M_I, \alpha_I)$
$S \leftarrow \{s_0\}$
$E \leftarrow \emptyset$
$N \leftarrow \{s_0\}$
while $N \neq \emptyset$ **do**
    $(M, \phi) \leftarrow Pick(N)$ // Take a node from N
    $N \leftarrow N\ \{(M, \phi)\}$
    foreach $t \in T$ *s.t.* $M[t\rangle M'$ **do**
        $\phi' \leftarrow \phi \oplus guard(t)$
        if $\neg(\phi' \sim false)$ then
            foreach *state* (M_o, ϕ_o), *from which* (M, ϕ) *is reachable* **do**
                if $(M' \succ M_o) \wedge (\phi' \sim \phi_o)$ **then**
                    foreach $p \in P$ **do**
                        if $M'(p) > M_o(p)$ **then**
                            $M'(p) = \omega$
            $E \leftarrow E \cup \{\langle (M, \phi), t, (M', \phi')\rangle\}$
            if $\forall(\bar{M}, \bar{\phi}) \in S: \bar{M} \neq M' \vee \neg(\phi' \sim \bar{\phi})$ **then**
                $S \leftarrow S \cup \{(M', \phi')\}$
                $N \leftarrow N \cup \{(M', \phi')\}$
return $\langle S, E, s_0\rangle$

Algorithm 2 describes the procedure for checking relaxed lazy soundness that consists of two steps:

(i) constructing an ACG (line 1),

(ii) exploring the ACG using graph-traversing techniques to verify the absence of markings that have more than one token in o (lines 2-3) and the ability to reach the sink place after each of the DPN transitions (lines 4-7).

Some of these checks can be included in the ACG construction procedure to answer the soundness of the model more quickly.

Algorithm 2 *CheckRelaxedLazySoundness*$(\mathcal{N}, (M_I, \alpha_I))$

Input: A DPN $\mathcal{N} = \langle P, T, F, V, guard \rangle$ with initial state (M_I, α_I).
Result: Whether or not $\mathcal{N}$ is relaxed lazy sound.
$\{S, E, s_0\} \leftarrow$ *ConstructACG*$(\mathcal{N}, M_I, \alpha_I)$
if $\exists (M, \phi) \in S: M(o) > 1$ then
    return $false$
$S_{feasible} \leftarrow \{s \in S \mid \exists (M_F, \phi): s[*\rangle (M_F, \phi) \wedge M_F > [o]\}$
foreach $t \in T$ **do**
    if $\forall (s, t, s'): s' \notin S_{feasible}$ **then**
        return $false$
return $true$

Proposition 3. Let $\mathcal{N} = \langle P, T, F, V, guard \rangle$ be a DPN with initial state (M_I, α_I). Procedure *CheckRelaxedLazySoundness*$(\mathcal{N}, (M_I, \alpha_I))$ terminates.

Proof. In paper [22], it was proved that a labeled transition system for a DPN with quasi-ordering $\sqsubseteq$ defined there is a well-structured transition system (WSTS) [30]. The ACG defined in our paper is actually a coverability graph for a labeled transition system defined in [22], where sets of states are replaced with formulas of the constraint language. In our case, relation $\sqsubseteq$ is decidable, and the procedure of computing node successors is also decidable (follows from the decidability of $\sqsubseteq$). From this, according to the WSTSs theory, it follows that the ACG is finite and effectively constructible. Since the ACG is finite, graph traversal methods on this structure are guaranteed to terminate. This proves that the procedure *CheckRelaxedLazySoundness* terminates. □

Proposition 4. Let $\mathcal{N} = \langle P, T, F, V, guard \rangle$ be a DPN with initial state (M_I, α_I). Let $ACG_{\mathcal{N}} = \langle S, E, s_0 \rangle$ be the abstract coverability graph of $\mathcal{N}$. Then for each $M \in \mathcal{M}_N$, $M(o) \leq 1$ if and only if $M(o) \leq 1$ for each $(M, \phi) \in S$.

Proof. (⇒) It follows from Proposition 1 that in $ACG_{\mathcal{N}}$, $M(o)$ may be either 0, 1, or ω. If $M(o) = \omega$, then o is unbounded and $RG_{\mathcal{N}}$ contains an execution that produces M' with $M'(o) > 1$, which contradicts the precondition $\forall M \in \mathcal{M}_N, M(o) \leq 1$.
(⇐) Follows from Proposition 2. □

Proposition 5. Let $\mathcal{N} = \langle P, T, F, V, guard \rangle$ be a DPN with initial state (M_I, α_I). Let $ACG_{\mathcal{N}} = \langle S, E, s_0 \rangle$ be the abstract coverability graph of $\mathcal{N}$. Then for each $t \in T$, $RG_{\mathcal{N}}$ contains an arc (v, t, v') included in a path to some node with $M_F \geqslant [o]$ if and only if there exists an arc $(s, t, s') \in E$ with a path from s' to a node with $M_F^{\omega} \geqslant [o]$.

Proof. (⇒) $RG_{\mathcal{N}}$ contains an execution that includes t and produces $M_F \geqslant [o]$. According to Proposition 1, the trace that represents this execution must be in $ACG_{\mathcal{N}}$. The trace execution ends at (M', ϕ'), where $M' \geqslant M_F$ (follows from Definition 4). Since $M' \geqslant M_F$, we have $M' \geqslant [o]$.
(⇐) $ACG_{\mathcal{N}}$ contains an execution that includes t and produces $M_F^{\omega} \geqslant [o]$. By Proposition 2, $RG_{\mathcal{N}}$ must exist a corresponding trace σ^* that includes all transitions of the trace in $ACG_{\mathcal{N}}$. Let this trace execution end at (M', α'). If $M_F^{\omega}(o) < \omega$, then $M'(o) = M_F^{\omega}(o) \geq 1$. If $M_F^{\omega}(o) = \omega$, then $M'(o)$ contains at least one token (follows from Definition 4). Thus, $M' \geqslant [o]$. □

The following is a corollary of Propositions 4 and 5:

Corollary 1. Let $\mathcal{N} = \langle P, T, F, V, guard \rangle$ be some DPN with initial state (M_I, α_I). $\mathcal{N}$ is relaxed lazy soundif and only if *CheckRelaxedLazySoundness*$(\mathcal{N}, (M_I, \alpha_I))$ returns $true$.

5. Implementation and Experiments

The proposed algorithm has been implemented as a module in the existing DPN soundness verification toolkit implemented on .NET WPF. The application with the added module is available for download on *https://github.com/SuvorovNM/DPN-Soundness-Verification*. Fig. 4 demonstrates the outputs provided by the application for DPN $\mathcal{N}$ from Fig. 1. A custom coloring function is implemented to ease the interpretation of ACGs, which can be huge in some cases.

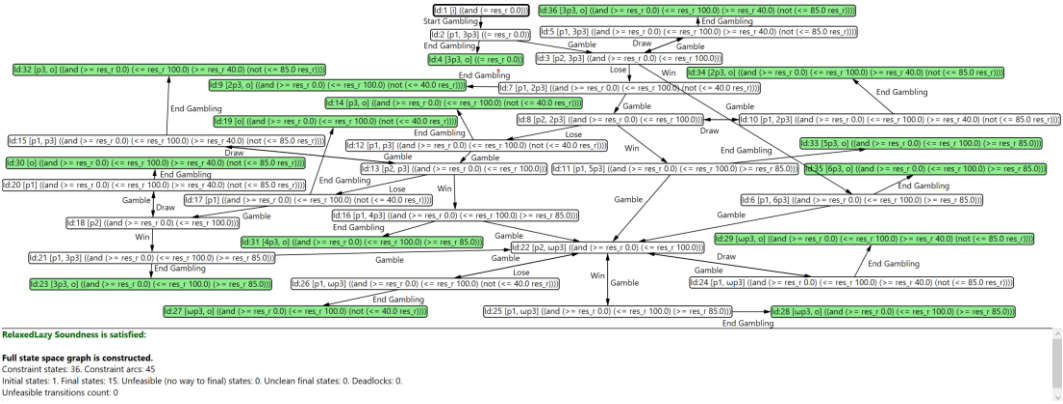

Fig. 4. Screenshot of the tool that implements the relaxed lazy soundness verification algorithm. The tool demonstrates $ACG_{\mathcal{N}}$ for DPN $\mathcal{N}$ from Fig. 1.

The tool takes as input a DPN in an extended PNML-based file format. All operations over formulas are conducted with the help of Z3 Solver [29]. At the implementation level, we have also made a small adjustment to the algorithm: we decided to check for $M(o) \leq 1$ when constructing an ACG and immediately return false when a node with $M(o) > 1$ is obtained. For large models, the ACG can be huge: even for a DPN with 10 transitions, an ACG may have more than 100000 nodes (see the `HugeACG.pnmlx` example in the repository). Thus, building the entire ACG when it is already known to be unsound may not be reasonable in some situations.

Table 1. Relaxed lazy soundness verification time for sample DPNs. Column 'Size' defines the net size in the number of places, denoted P, the number of transitions, denoted T, and the number of variables, denoted V.

Model	Size	Classic Sound	Relaxed Lazy Sound	Verification Time: Classic	Verification Time: Relaxed Lazy
Gambling Example (Fig. 1)	5P + 6T + 1V	False	True	34ms	89ms
Livelock Example [22]	3P+3T+2V	True	True	130ms	18ms
Digital Whiteboard: Transfer [9]	7P+6T+3V	False	True	40ms	15ms
Package Handling [23]	16P+28T+5V	False	False	1010ms	213ms
Road Fines Mined [9]	9P+19T+8V	False	False	491ms	194ms
Simple Auction [31]	4P+4T+2V	False	True	193ms	90ms

Table 1 reports on how much time the proposed algorithm takes to verify the relaxed lazy soundness of different DPN models presented in the literature. For bounded DPNs, as expected, verifying relaxed lazy soundness is usually faster than classical soundness, since classical soundness requires constructing more refined state space structures. If a net is unbounded (as $\mathcal{N}$ in Fig. 1), then classical soundness can be checked much faster, since we can immediately return $false$ when a node with ω is obtained.

We have also tested the performance of the algorithm on synthetically generated data. Given $n \in \mathbb{N}$, we considered DPNs parameterized according to the following setup:

- $1.2n$ places,
- n transitions,
- $0.25n$ variables, and
- $0.5n$ conditions.

For each $n \in \mathbb{N}$ from 3 to 90, we generated 10 DPNs that have at least one trace leading to $M \geqslant [o]$ and at least 40% feasible transitions (i.e., transitions occurring in executions leading to some $M \geqslant [o]$) using the tool introduced in [22], refined for the current study. In our experiments, we have considered nets with a low and moderate level of inherent concurrency (a limitation of the DPN generation tool). The obtained results are visualized in Fig. 5. The plot shows that our algorithm typically takes less than a minute to verify the relaxed lazy soundness of a DPN with fewer than 100 transitions.

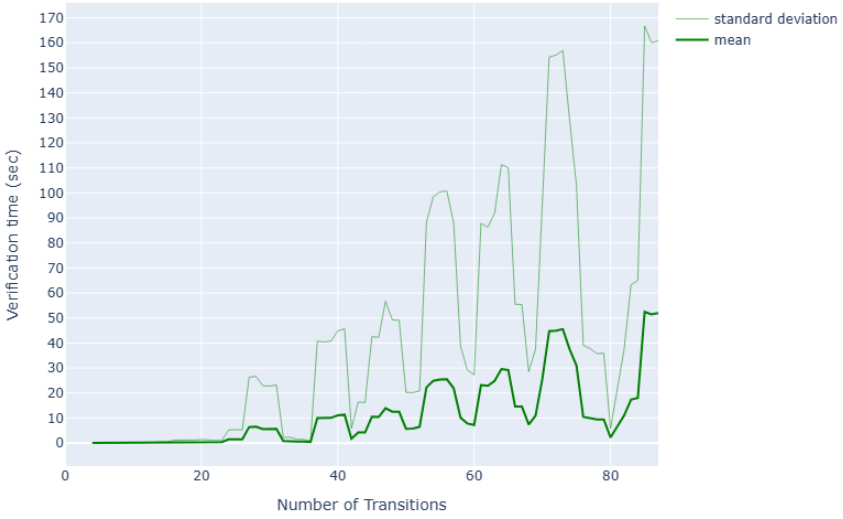

Fig. 5. CheckRelaxedLazySoundnes procedure execution time on DPNs of different sizes.

Note that in the worst-case scenarios, the verification can take an unreasonable amount of time. This may be the case for DPNs with a high inherent level of concurrency, which can lead to state space explosion, and for DPNs with long constraints on transitions, which can seriously impact the speed of formula operations. However, the results of our preliminary experimental evaluation show that the algorithm is promising and can be used in practice to verify the correctness of resource-oriented data-aware process models. Interestingly, the algorithm could also be used for preliminary verification of arbitrary process models, since for bounded models, it terminates sufficiently faster than the algorithm for classical soundness verification.

6. Conclusion

In this paper, we have proposed the notion of relaxed lazy soundness that can be used to check correctness of resource-oriented process models. We have shown that this property is decidable both for classical Petri nets and for Data Petri nets and could be evaluated using a single coverability graph construction compared to separate relaxed and lazy soundness properties that require solving a reachability problem to be decided. Performance evaluation of the algorithm implementation shows its practical applicability for process models of small and medium size.

The proposed algorithm can be used immediately after building a process model to verify that the model actually contains executions that terminate `adequately' and that each process activity can be present in such executions. One of practical applications of the proposed algorithm could be a plug-in for tools for constructing data-aware process models. The algorithm could run in the background and warn if the user's model is ill-structured.

In the future, we plan to estimate our verification algorithm on real industrial process models both in terms of speed and in terms of model errors found. This could help to identify the overall practical applicability of the relaxed lazy soundness property in industry. Another direction for investigation is finding an abstract state space structure that would be smaller than the proposed ACG but that would be sufficient for verifying relaxed lazy soundness. In its current state, the algorithm provides rather poor results for large models with a high level of concurrency, and it is interesting whether this issue can be addressed. Lastly, a promising direction could be to define the rules to reduce an ACG in order to make its visualization perceivable even for models of medium and large sizes, where an ACG could contain hundreds or thousands of nodes.

References

[1]. Weske M. Business Process Management: Concepts, Languages, Architectures. Introduction. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–23.
[2]. van der Aalst W. M. P. Verification of workflow nets. In Application and Theory of Petri Nets 1997, 1997, pp. 407–426.
[3]. Martens A. Analyzing web service based business processes. In Fundamental Approaches to Software Engineering, 2005, pp. 19–33.
[4]. van Hee K. M., Sidorova N., Voorhoeve M. Soundness and separability of workflow nets in the stepwise refinement approach. In Applications and Theory of Petri Nets, 2003, pp. 337–356.
[5]. Dehnert J., Rittgen P. Relaxed soundness of business processes. In Advanced Information Systems Engineering, 2001, pp. 157–170.
[6]. Puhlmann F., Weske M. Investigations on soundness regarding lazy activities. In Business Process Management, 2006, pp. 145–160.
[7]. van der Aalst W. M. P., van Hee K. M., ter Hofstede A. H. M., Sidorova N., Verbeek H. M. W., Voorhoeve M., Wynn M. T. Soundness of workflow nets: classification, decidability, and analysis. Formal Aspects of Computing, 23(3), 2011, pp. 333–363.
[8]. de Leoni M., van der Aalst W. M. P. Data-aware process mining: Discovering decisions in processes using alignments. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, 2013, pp. 1454–1461.
[9]. Mannhardt F. Multi-perspective process mining. Ph.D. dissertation, Eindhoven University of Technology, 2018.
[10]. de Leoni M., Felli P., Montali M. A holistic approach for soundness verification of decision-aware process models. In Conceptual Modeling, 2018, pp. 219–235.
[11]. Esparza J., Nielsen M. Decidability issues for Petri Nets. Information Processing and Cybernetics, *30*(3), 1994, pp. 143-160.
[12]. Czerwinski W., Lasota S., Lazic R., Leroux J., Mazowiecki F. The reachability problem for Petri nets is not elementary. In STOC 2019, 2019, pp. 24–33.
[13]. Bazhenova E., Zerbato F., Oliboni B., Weske M. From bpmn process models to dmn decision models. Information Systems, vol. 83, 2019, pp. 69–88.
[14]. Knuplesch D., Ly L. T., Rinderle-Ma S., Pfeifer H., Dadam P. On enabling data-aware compliance checking of business process models. In Conceptual Modeling, 2010, pp. 332–346.
[15]. van der Aalst W. M. The application of petri nets to workflow management. Journal of Circuits, Systems and Computers, vol. 8, 1998, pp. 21–66.
[16]. Awad A., Decker G., Lohmann N. Diagnosing and repairing data anomalies in process models. In Business Process Management Workshops, 2010, pp. 5–16.
[17]. Sidorova N., Stahl C., Trcka N. Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible. Information Systems, 36(7) , 2011, pp. 1026–1043.
[18]. Felli P., de Leoni M., Montali M. Soundness verification of decision-aware process models with variable-to-variable conditions. In ACSD 2019, 2019, pp. 82–91.

[19]. Tao X., Liu G., Yang B., Yan C., Jiang C. Workflow nets with tables and their soundness. IEEE Transactions on Industrial Informatics, 16(3), 2020, pp. 1503–1515.
[20]. Song J., Liu G, Wang M. Model checking of workflow nets with tables and constraints. ACM Transactions on Autonomous and Adaptive Systems, 2025, 37p.
[21]. Felli P., Montali M., Winkler S. Soundness of data-aware processes with arithmetic conditions. In Advanced Information Systems Engineering, 2022, pp. 389–406.
[22]. Suvorov N. M., Lomazova I. A. Verification of data-aware process models: Checking soundness of data petri nets. Journal of Logical and Algebraic Methods in Programming, vol. 138, 100953, 2024.
[23]. Felli P., de Leoni M., Montali M. Soundness verification of data-aware process models with variable-to-variable conditions. Fundamental Informaticae, 182(1), 2021, pp. 1–29.
[24]. Murata T. Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4), pp. 541–580, 1989.
[25]. Tarski A. A decision method for elementary algebra and geometry. Journal of Symbolic Logic, 14(3), pp. 188–188, 1949.
[26]. Lu F., Tao R., Du Y., Zeng Q., Bao Y. Deadlock detection-oriented unfolding of unbounded petri nets. Information Sciences, vol. 497, 2019, pp. 1–22.
[27]. Ding Z., Jiang C., Zhou M. Deadlock checking for one-place unbounded petri nets based on modified reachability trees. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 38(3), pp. 881–883, 2008.
[28]. Puhlmann F. Soundness verification of business processes specified in the pi-calculus. In On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, 2007, pp. 6–23.
[29]. de Moura L., Bjørner N. Z3: An efficient smt solver. In TACAS, 2008, pp. 337–340.
[30]. Finkel A., Schnoebelen P. Well-structured transition systems everywhere! Theoretical Computer Science, 256(1), 2001, pp. 63–92.
[31]. Felli P., Montali M., Winkler S. Repairing soundness properties in data-aware processes. In 2023 5th International Conference on Process Mining (ICPM), 2023, pp. 41–48.

Информация об авторах / Information about authors

Николай Михайлович СУВОРОВ– аспирант аспирантской школы по компьютерным наукам НИУ ВШЭ, стажер-исследователь лаборатории процессно-ориентированных информационных систем (ПОИС) факультета компьютерных наук НИУ ВШЭ. Сфера научных интересов: теория автоматов, распределенные процессы, верификация и исправление моделей.

Nikolai Mikhailovich SUVOROV – postgraduate student at Doctoral School of Computer Science, HSE University, research fellow at Laboratory of Process-Aware Information Systems, Faculty of Computer Science, HSE University. Research interests: automata theory, distributed processes, and model verification and repair.

Ирина Александровна ЛОМАЗОВА – доктор физико-математических наук, профессор факультета компьютерных наук НИУ ВШЭ, научный руководитель научно-учебной лаборатории процессно-ориентированных информационных систем (ПОИС) НИУ ВШЭ. Область научных интересов: анализ и моделирование бизнес-процессов, сети Петри, вложенные сети Петри, процессно-ориентированные информационные системы, формальные модели распределённых систем.

Irina Alexandrovna LOMAZOVA – Dr. Sci. (Phys.-Math.), professor of the faculty of computer science in HSE University, and laboratory head of the Laboratory for Process-Aware Information Systems (PAIS Lab), HSE University. Doctor of Sciences in Theoretical Foundations of Computer Science Russian Academy of Sciences Dorodnitsyn Computation Center since 2002. Research interests mainly include analysis and modeling of business processes, Petri nets, process-oriented information systems, formal models of distributed systems.