Fast Termination and Workflow Nets*

Piotr Hofman[1] [0000-0003-2914-2734], Filip Mazowiecki[1] [0000-0002-4535-6508], and Philip Offtermatt[1,2] [0000-0001-8477-2849]

[1] University of Warsaw, Poland piotr.hofman@uw.edu.pl
f.mazowiecki@mimuw.edu.pl
[2] Université de Sherbrooke, Sherbrooke, Canada
Philip.Offtermatt@usherbrooke.ca

Abstract. Petri nets are an established model of concurrency. A Petri net is terminating if for every initial marking there is a uniform bound on the length of all possible runs. Recent work on the termination of Petri nets suggests that, in general, practical models should terminate fast, *i.e.* in polynomial time. In this paper we focus on the termination of workflow nets, an established variant of Petri nets used for modelling business processes. We partially confirm the intuition on fast termination by showing a dichotomy: workflow nets are either non-terminating or they terminate in linear time.

The central problem for workflow nets is to verify a correctness notion called soundness. In this paper we are interested in generalised soundness which, unlike other variants of soundness, preserves desirable properties like composition. We prove that verifying generalised soundness is coNP-complete for terminating workflow nets.

In general the problem is PSPACE-complete, thus intractable. We utilize insights from the coNP upper bound to implement a procedure for generalised soundness using MILP solvers. Our novel approach is a semi-procedure in general, but is complete on the rich class of terminating workflow nets, which contains around 90% of benchmarks in a widely-used benchmark suite. The previous state-of-the-art approach for the problem is a different semi-procedure which is complete on the incomparable class of so-called free-choice workflow nets, thus our implementation improves on and complements the state-of-the-art.

Lastly, we analyse a variant of termination time that allows parallelism. This is a natural extension, as workflow nets are a concurrent model by design, but the prior termination time analysis assumes sequential behavior of the workflow net. The sequential and parallel termination times can be seen as upper and lower bounds on the time a process represented as a workflow net needs to be executed. In our experimental section we show that on some benchmarks the two bounds differ significantly, which agrees with the intuition that parallelism is inherent to workflow nets.

Keywords: Workflow · Soundness · Fast termination · generalised Soundness · Polynomial time.

* This work was partially supported by ERC grant INFSYS: 501-D110-60-0196287. P. Offtermatt is now at Informal Systems, Munich, Germany.

1 Introduction

Petri nets are a popular formalism to model problem in software verification [23], business processes [1] and many more (see [44] for a survey). One of the fundamental problems for such models is the *termination problem*, *i.e.* whether the lengths of all runs are universally bounded. There are two natural variants of this problem. First, if the initial configuration is fixed then the problem is effectively equivalent to the boundedness problem, known to be EXPSPACE-complete for Petri nets [37,43]. Second, if termination must hold for all initial configurations the problem known to be in polynomial time [31], and such nets are known as *structurally terminating*. In this paper we are interested in the latter variant.

Termination time is usually studied for *vector addition system with states* (VASS), an extension of Petri nets that allows the use of control states. In particular, the aforementioned EXPSPACE and polynomial time bounds work for VASS. In 2018, a deeper study of the termination problem for VASS was initiated [13]. This study concerns the asymptotics of the function $f(n)$ bounding the length of runs, where n bounds the size of the initial configuration. The focus is particularly on classes where $f(n)$ is a polynomial function, suggesting that such classes are more relevant for practical applications. This line of work was later continued for variants of VASS involving probabilities [12] and games [32].

For VASS the function $f(n)$ can asymptotically be as big as $F_i(n)$ in the Grzegorczyk hierarchy for any finite i (recall that $F_3(n)$ is nonelementary and $F_\omega(n)$ is Ackermann) [45,36]. It was known that for terminating Petri nets many problems are considerably simpler [42]. However, to the best of our knowledge, the asymptotic behaviour of $f(n)$ was not studied for Petri nets.

Our contributions. In this paper we focus on *workflow nets*, a class of Petri nets widely used to model business processes [1]. Our first result is the following dichotomy: any workflow net is either non-terminating or $f(n)$ is linear. This confirms the intuition about fast termination of practical models [13]. In our proof, we follow the intuition of applying linear algebra from [42] and rely on recent results on workflow nets [10]. We further show that the optimal constant $a_\mathcal{N}$ such that $f(n) = a_\mathcal{N} \cdot n$ can be computed in polynomial time. The core of this computation relies on a reduction to continuous Petri nets [20], a well known relaxation of Petri nets. Then we can apply standard tools from the theory of continuous Petri nets, where many problems are in polynomial time [20,7].

For workflow nets, the central decision problems are related to soundness. There are many variants of this problem (see [2] for a survey). For example k-*soundness* intuitively verifies that k started processes eventually properly terminate. We are interested in *generalised soundness*, which verifies whether k-soundness holds for all k [27,28,26]. The exact complexity of most popular soundness problems was established only recently in 2022 [10]. Generalised soundness is surprisingly PSPACE-complete. Other variants, like k-soundness, are EXPSPACE-complete, thus computationally harder, despite having a seemingly less complex definition. Moreover, unlike k-soundness and other variants, generalised soundness preserves desirable properties like composition [27]. Building on our first result, *i.e.* the dichotomy between non-terminating and linearly terminating workflow nets, our second result is that generalised soundness is coNP-complete for terminating workflow nets.

Finally, we observe that the asymptotics of $f(n)$ are defined with the implicit assumption that transitions are fired sequentially. Since workflow nets are models for parallel executions it is natural to expect that runs would also be performed in parallel. Our definition of parallel executions is inspired by similar concepts for time Petri nets, and can be seen as a particular case [5]. We propose a definition of the optimal running time of runs exploiting parallelism and denote this time $g(n)$, where n bounds the size of the initial marking. We show that the asymptotic behaviour of $g(n)$, similar to $f(n)$, can be computed in polynomial time, for workflow nets with mild assumptions. Together, the two functions $f(n)$ and $g(n)$ can be seen as (pessimistic) upper bound and (optimistic) lower bound on the time needed for the workflow net to terminate.

Experiments. Based on our insights, we implement several procedures for problems related to termination in workflow nets. Namely, we implement our algorithms for checking termination, for deciding generalised soundness of workflow nets and for computing the asymptotic behaviour of $f(n)$. We additionally implement procedures to compute $f(k), g(k)$ and decide k-soundness for terminating workflow nets. To demonstrate the efficacy of our procedures, we test our implementation on a popular and well-studied benchmark suite of 1382 workflow nets, originally introduced in [19]. It turns out that the vast majority of instances (roughly 90%) is terminating, thus the class of terminating workflow nets seems highly relevant in practice. Further, we positively evaluate our algorithm for generalised soundness against a recently proposed state-of-art approach [11] which semi-decides the property in general, and is further exact on the class of *free-choice workflow nets* [3]. Interestingly, our novel approach for generalised soundness is also a semi-procedure in general, but precise on terminating workflow nets. The approach from [11] is implemented as an $\exists\forall$-formula from $\mathrm{FO}(\mathbb{Q}, <, +)$, while our approach manages to avoid any quantifier alternations. It turns out that our approach is faster on over 95% of benchmark instances, and thus significantly improves upon the state-of-art. The mean analysis time for our approach is just $12.8ms$, while it is about $2s$ for the previous state-of-the-art. In addition, the classes of free-choice and terminating workflow nets are incomparable, thus our approach complements the state-of-the-art.

Related work. For general Petri nets and VASS the most well-known problem is reachability, recently shown to be Ackermann-complete [35,15,34]. Despite its high complexity, there are tools for the problem [17,47], including some based on integer and continuous relaxations [9,6,22]. Reachability was also studied in the context of terminating models. In particular, it is PSPACE-complete for structurally terminating Petri nets [42] and EXPSPACE-complete for polynomially terminating VASS [33].

Most algorithms for soundness are based on reductions to reachability [1], this is the case for the first algorithms for generalised soundness [28,26]. However, such reductions only imply Ackermannian upper bounds on the problem, while a direct study yielded elementary complexities [10].

A different class of approaches for soundness relies on *reduction rules*, which can be applied iteratively to reduce the size of a net while exactly preserving soundness [41,4]. These approaches are not precise in general, but can be for subclasses, *e.g.* for *live and bounded* free-choice workflow nets [16]. We use a certain set of reduction rules [14] for generalised soundness in our experimental evaluation.

There exist many established tools and frameworks for the analysis of workflow nets, for example Woflan [46], WoPeD [21], and ProM [18]. However, when it comes to soundness problems, these tools typically focus on k-soundness, with a particular focus on $k = 1$ (except for the discussed tool in [11]).

Organisation. In Section 2 we define the models, problems and basic notation. In Section 3 we prove the dichotomy between non-terminating and linear workflow nets. Then, we show how to compute the linear constants for terminating workflow nets in Section 4. Building on the dichotomy we show that generalised soudness is coNP-complete in Section 5. In Section 6 we define and compute a variant of termination time that takes into account parallelism. We present our experimental results in Section 7. Most proofs can be found in the appendix.

2 Preliminaries

We write $\mathbb{N}, \mathbb{N}_{>0}, \mathbb{Z}, \mathbb{Q}$ and $\mathbb{Q}_{\geq 0}$ for the naturals (including 0), the naturals without 0, the integers, the rationals, and the nonnegative rationals, respectively.

Let N be a set of numbers, *e.g.* $N = \mathbb{N}$. For $d, d_1, d_2 \in \mathbb{N}_{>0}$ we write N^d for the set of vectors with elements from N in dimension d. Similarly, $N^{d_1 \times d_2}$ is the set of matrices with d_1 rows and d_2 columns and elements from N. We use bold font for vectors and matrices. For $a \in \mathbb{Q}$ and $d \in \mathbb{N}_{>0}$, we write $\boldsymbol{a}^d \coloneqq (a, a, \ldots, a) \in \mathbb{Q}^d$ (or $\boldsymbol{a}$ if d is clear from context). In particular $\mathbf{0}^d = \mathbf{0}$ is the zero vector.

Sometimes it is more convenient to have vectors with coordinates in a finite set. Thus, for a finite set S, we write $\mathbb{N}^S$, $\mathbb{Z}^S$, and $\mathbb{Q}^S$ for the set of vectors over naturals, integers and rationals. Given a vector $\boldsymbol{v}$ and an element $s \in S$, we write $\boldsymbol{v}(s)$ for the value $\boldsymbol{v}$ assigns to s.

Given $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{Q}^S$, we write $\boldsymbol{v} \leq \boldsymbol{w}$ if $\boldsymbol{v}(s) \leq \boldsymbol{w}(s)$ for all $s \in S$, and $\boldsymbol{v} < \boldsymbol{w}$ if $\boldsymbol{v} \leq \boldsymbol{w}$ and $\boldsymbol{v}(s) < \boldsymbol{w}(s)$ for some $s \in S$. The *size* of S, denoted $|S|$, is the number of elements in S. We define the *norm* of a vector $\|\boldsymbol{v}\| \coloneqq \max_{s \in S} |\boldsymbol{v}(s)|$, and the norm of a matrix $\boldsymbol{A} \in \mathbb{Q}^{m \times n}$ as $\|\boldsymbol{A}\| \coloneqq \max_{1 \leq j \leq m, 1 \leq i \leq n} |A(i, j)|$. For a set $S \in \mathbb{Q}^d$, we denote by $\overline{S} \in \mathbb{R}^d$ the closure of S in the euclidean space.

2.1 (Integer) Linear Programs

Let $n, m \in \mathbb{N}_{>0}$, $\boldsymbol{A} \in \mathbb{Z}^{m \times n}$, and $\boldsymbol{b} \in \mathbb{Z}^m$. We say that $G \coloneqq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ is a *system of linear inequalities* with m inequalities and n variables. The *norm* of a system G is defined as $\|G\| \coloneqq \|\boldsymbol{A}\| + \|\boldsymbol{b}\| + m + n$. An $(m \times n)$*-ILP*, short for *integer linear program*, is a system of linear inequalities with m inequalities and n variables, where we are interested in the integer solutions. An $(m \times n)$*-LP* is such a system where we are interested in the rational solutions. We use the term MILP, short for *mixed integer linear program*, for a system where some variables are allowed to take on rational values, while others are restricted to integer values.

We allow syntactic sugar in ILPs and LPs, such as allowing constraints $x \geq y$, $x = y$, $x < y$ (in the case of ILPs). Sometimes we are interested in finding optimal solutions. This means we have a objective function, formally a linear function on the variables of the system, and look for a solution that either maximizes or minimizes the value of that function. For LPs, finding an optimal solution can be done in polynomial time, while this is NP-complete for ILPs and MILPs.

2.2 Petri nets

A *Petri net* $\mathcal{N}$ is a triple (P, T, F), where P is a finite set of *places*; T is a finite set of *transitions* such that $T \cap P = \emptyset$; and $F : ((P \times T) \cup (T \times P)) \to \mathbb{N}$ is a function describing its *arcs*. A *marking* is a vector $\boldsymbol{m} \in \mathbb{N}^P$. We say that $\boldsymbol{m}(p)$ is the number of *tokens* in place $p \in P$ and p is *marked* if $\boldsymbol{m}(p) > 0$. To write markings, we list only non-zero token amounts. For example, $\boldsymbol{m} = \{p_1 : 2, p_2 : 1\}$ is the marking $\boldsymbol{m}$ with $\boldsymbol{m}(p_1) = 2, \boldsymbol{m}(p_2) = 1$ and $\boldsymbol{m}(p) = 0$ for all $p \in P \setminus \{p_1, p_2\}$.

Let $t \in T$. We define the vector ${}^\bullet t \in \mathbb{N}^P$ by ${}^\bullet t(p) \coloneqq F(p, t)$ for $p \in P$. Similarly, the vector $t^\bullet \in \mathbb{N}^P$ is defined by $t^\bullet(p) \coloneqq F(t, p)$ for $p \in P$. We write the *effect* of t as $\Delta(t) \coloneqq t^\bullet - {}^\bullet t$. A transition t is *enabled* in a marking $\boldsymbol{m}$ if $\boldsymbol{m} \geq {}^\bullet t$. If t is enabled in the marking $\boldsymbol{m}$, we can *fire* it, which leads to the marking $\boldsymbol{m}' \coloneqq \boldsymbol{m} + \Delta(t)$, which we denote $\boldsymbol{m} \to^t \boldsymbol{m}'$. We write $\boldsymbol{m} \to \boldsymbol{m}'$ if there exists some $t \in T$ such that $\boldsymbol{m} \to^t \boldsymbol{m}'$.

A sequence of transitions $\pi = t_1 t_2 \ldots t_n$ is called a *run*. We denote the *length* of π as $|\pi| \coloneqq n$. A run π is *enabled* in a marking $\boldsymbol{m}$ iff $\boldsymbol{m} \to^{t_1} \boldsymbol{m}_1 \to^{t_2} \boldsymbol{m}_2 \to^{t_3} \ldots \boldsymbol{m}_{n-1} \to^{t_n} \boldsymbol{m}'$ for some markings $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}' \in \mathbb{N}^P$. The set of all runs is denoted $\text{Runs}_{\mathcal{N}}^{\boldsymbol{m}}$, *i.e.* $\pi \in \text{Runs}_{\mathcal{N}}^{\boldsymbol{m}}$ if π is enabled in $\boldsymbol{m}$. The *effect* of π is $\Delta(\pi) \coloneqq \sum_{i \in [1..n]} \Delta(t_i)$. *Firing* π from $\boldsymbol{m}$ leads to a marking $\boldsymbol{m}'$, denoted $\boldsymbol{m} \to^\pi \boldsymbol{m}'$, iff $\boldsymbol{m} \in \text{Runs}_{\mathcal{N}}^{\boldsymbol{m}}$ and $\boldsymbol{m}' = \boldsymbol{m} + \Delta(\pi)$. We denote by $\to^*$ the reflexive, transitive closure of $\to$. Given two runs $\pi = t_1 t_2 \ldots t_n$ and $\pi' = t'_1 t'_2 \ldots t'_{n'}$ we denote $\pi\pi' \coloneqq t_1 t_2 \ldots t_n t'_1 t'_2 \ldots t'_{n'}$.

The size of a Petri net is defined as $|\mathcal{N}| = |P| + |T| + |F|$. We define the *norm* of $\mathcal{N}$ as $\|\mathcal{N}\| \coloneqq \|F\| + 1$, where we view F as a vector in $\mathbb{N}^{(P \times T) \cup (T \times P)}$.

We also consider several variants of the firing semantics of Petri nets which we will need throughout the paper. In the *integer semantics*, we consider markings over $\mathbb{Z}^P$, and transitions can be fired without being enabled. To denote the firing and reachability relations, we use the notations $\to_{\mathbb{Z}}$ and $\to_{\mathbb{Z}}^*$. In the *continuous semantics* [20], we consider markings over $\mathbb{Q}_{\geq 0}^P$. Given $t \in T$ and a *scaling factor*

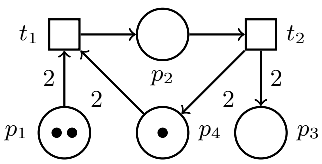


Fig. 1. A Petri net with places p_1, p_2, p_3, p_4 and transitions t_1, t_2. Marking $\{p_1\colon 2, p_4\colon 1\}$ is drawn. No transition is enabled.

$\beta \in \mathbb{Q}_{\geq 0}$[3], the effect of firing βt is $\Delta(\beta t) \coloneqq \beta \cdot \Delta(t)$. Further, βt is enabled in a marking $\boldsymbol{m}$ iff $\beta \cdot {}^{\bullet}t \leq \boldsymbol{m}$. We use $\rightarrow_{\mathbb{Q}_{\geq 0}}$ for the continuous semantics, that is, $\boldsymbol{m} \rightarrow^{\beta t}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}$ means βt is enabled in $\boldsymbol{m}$ and $\boldsymbol{m'} = \boldsymbol{m} + \Delta(\beta t)$. A *continuous run* π is a sequence of factors and transitions $\beta_1 t_1 \beta_2 t_2 \dots \beta_n t_n$. Enabledness and firing are extended to continuous runs: $\boldsymbol{m} \rightarrow^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}$ holds iff there exist $\boldsymbol{m}_1, \dots, \boldsymbol{m}_{n-1}$ such that $\boldsymbol{m} \rightarrow^{\beta_1 t_1}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}_1 \rightarrow^{\beta_2 t_2}_{\mathbb{Q}_{\geq 0}} \cdots \boldsymbol{m}_{n-1} \rightarrow^{\beta_n t_n}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}$. The *length* of π is $|\pi|_c \coloneqq \sum_{i=1}^{n} \beta_i$. Given $\alpha \in \mathbb{Q}_{\geq 0}$ and a run $\pi = \beta_1 t_1 \beta_2 t_2 \dots \beta_n t_n$ we write $\alpha\pi$ to denote the run $(\alpha\beta_1)t_1(\alpha\beta_2)t_2 \dots (\alpha\beta_n)t_n$. We introduce a lemma stating that continuous runs can be rescaled.

Lemma 1 (Lemma 12(1) in [20]). *Let $\alpha \in \mathbb{Q}_{\geq 0}$. Then $\boldsymbol{m} \rightarrow^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}$ if and only if $\alpha\boldsymbol{m} \rightarrow^{\alpha\pi}_{\mathbb{Q}_{\geq 0}} \alpha\boldsymbol{m'}$.*

Each run under normal semantics or integer semantics is *equivalent* to a continuous run i.e. $t_1 t_2 \dots t_n \approx 1t_1 1t_2 \dots 1t_2$. Given $\pi \in \mathrm{Runs}_{\mathcal{N}}^{\boldsymbol{m}}$ (*i.e.* a standard run) we define $\alpha\pi = \alpha\pi_c$ where $\pi_c \approx \pi$ is a continuous run. If $\pi_c = \beta_1 t_1 \dots \beta_n t_n$ with $\beta_i \in \mathbb{N}$ for all $i \in \{1, \dots, n\}$ then we also call π a (standard) run, *i.e.* the run where every transition t_i is repeated β_i times.

We define the set of continuous runs enabled from $\boldsymbol{m} \in \mathbb{N}^P$ in $\mathcal{N}$ as $\mathrm{CRuns}_{\mathcal{N}}^{\boldsymbol{m}}$. The *Parikh image* of a continuous run $\pi = \beta_1 t_1 \beta_2 t_2 \dots \beta_n t_n$ is the vector $\boldsymbol{R}_\pi \in \mathbb{Q}^T$ such that $\boldsymbol{R}_\pi(t) = \sum_{i | t_i = t} \beta_i$. For a (standard) run π we define its Parikh image $\boldsymbol{R}_\pi \coloneqq \boldsymbol{R}_{\pi_c}$ where $\pi_c \approx \pi$. Given a vector $\boldsymbol{R} \in \mathbb{Q}^T_{\geq 0}$, we define $\Delta(\boldsymbol{R}) \coloneqq \sum_{t \in T} \boldsymbol{R}(t) \cdot \Delta(t)$, ${}^{\bullet}\boldsymbol{R} \coloneqq \sum_{t \in T} {}^{\bullet}t \cdot \boldsymbol{R}(t)$, $\boldsymbol{R}^{\bullet} \coloneqq \sum_{t \in T} t^{\bullet} \cdot \boldsymbol{R}(t)$. Note that $\boldsymbol{R}$ is essentially a run without imposing an order on the transitions. For ease of notation, we define $\Delta(T)$ as a matrix with columns indexed by T and rows indexed by P, where $\Delta(T)(t)(p) \coloneqq \Delta(t)(p)$. Then $\Delta(\boldsymbol{R}) = \Delta(T)\boldsymbol{R}$.

Example 1. Consider the Petri net drawn in Figure 1. Marking $\boldsymbol{m} \coloneqq \{p_1\colon 2, p_4\colon 1\}$ enables no transitions. However, we have $\boldsymbol{m} \rightarrow^{t_1 t_2}_{\mathbb{Z}} \{p_3\colon 2\}$. We also have $\boldsymbol{m} \rightarrow^{t_2 t_1}_{\mathbb{Z}} \{p_3\colon 2\}$, since the transition order does not matter under the integer semantics. Thus, when we take $R = \{t_1\colon 1, t_2\colon 1\}$, we have $\boldsymbol{m} \rightarrow^{R}_{\mathbb{Z}} \{p_3\colon 2\}$.

Under the continuous semantics we can fire $1/2 t_1$, which is impossible under the normal semantics. For example, we have $\boldsymbol{m} \rightarrow^{1/2 t_1}_{\mathbb{Q}_{\geq 0}} \{p_1\colon 1, p_2\colon 1/2\} \rightarrow^{1/2 t_2}_{\mathbb{Q}_{\geq 0}} \{p_1\colon 1, p_3\colon 1, p_4\colon 1\} \rightarrow^{1/3 t_1}_{\mathbb{Q}_{\geq 0}} \{p_1\colon 1/3, p_2\colon 1/3, p_3\colon 1, p_4\colon 2/3\}$.

[3] Sometimes scaling factors are defined to be at most 1. The definitions are equivalent: Scaling larger than 1 can be done by firing the same transition multiple times.

2.3 Workflow Nets

A *workflow net* is a Petri net $\mathcal{N}$ such that:

- There exists an *initial* place i with $F(t, \mathsf{i}) = 0$ for all $t \in T$ (*i.e.* no tokens can be added to i);
- there exists a *final* place f with $F(\mathsf{f}, t) = 0$ for all $t \in T$ (*i.e.* no tokens can be removed from f); and
- in the graph (V, E) with $V = P \cup T$ and $(u, v) \in E$ iff $F(u, v) \neq 0$, each $v \in V$ lies on at least one path from i to f.

We say that $\mathcal{N}$ is *k-sound* iff for all $\boldsymbol{m}$, $\{\mathsf{i}\colon k\} \rightarrow^* \boldsymbol{m}$ implies $\boldsymbol{m} \rightarrow^* \{\mathsf{f}\colon k\}$. Further, we say $\mathcal{N}$ is *generalised sound* iff it is k-sound for all k.

A place $p \in P$ is *nonredundant* if $\{\mathsf{i}\colon k\} \rightarrow^* \boldsymbol{m}$ for some $k \in \mathbb{N}$ and marking $\boldsymbol{m}$ with $\boldsymbol{m}(p) > 0$, and *redundant* otherwise. We accordingly say that $\mathcal{N}$ is *nonredundant* if all $p \in P$ are nonredundant, otherwise $\mathcal{N}$ is *redundant*. A redundant workflow net can be made nonredundant by removing each redundant place $p \in P$ and all transitions such that ${}^\bullet t(p) > 0$ or $t^\bullet(p) > 0$. Note that this does not impact behaviour of the workflow, as the discarded transitions could not be fired in the original net. A polynomial-time saturation procedure can identify redundant places, see [28, Thm. 8, Def. 10, Sect. 3.2] and [10, Prop. 5.2].

If $\mathcal{N}$ is a workflow net, we write $\text{Runs}_{\mathcal{N}}^k$ for the set of runs that are enabled from the marking $\{\mathsf{i}\colon k\}$, and $\text{CRuns}_{\mathcal{N}}^k$ for the same for continuous runs. Lemma 1 implies that if $\pi \in \text{Runs}_{\mathcal{N}}^k$ then $\frac{1}{k}\pi \in \text{CRuns}_{\mathcal{N}}^1$. The converse does not need to hold as the rescaled continuous run need not have natural coefficients.

Example 2. The Petri net in Figure 1 can be seen as a workflow net with initial place p_1 and final place p_3. The workflow is not k-sound for any k. Further, the net is redundant: $\{\mathsf{i}\colon k\}$ is a deadlock for every k, so places p_2, p_3 and p_4 are redundant. ◁

2.4 Termination Complexity

Let $\mathcal{N}$ be a workflow net. Let us define as $\mathit{MaxTime}_{\mathcal{N}}(k)$ the supremum of lengths among runs enabled in $\{\mathsf{i}\colon k\}$, that is, $\mathit{MaxTime}_{\mathcal{N}}(k) = \sup\{|\pi| \mid \pi \in \text{Runs}_{\mathcal{N}}^k\}$. We say that $\mathcal{N}$ is *terminating* if $\mathit{MaxTime}_{\mathcal{N}}(k) \neq \infty$ for all $k \in \mathbb{N}_{>0}$, otherwise it is *non-terminating*.

We say that $\mathcal{N}$ has *polynomial termination time* if there exist $d \in \mathbb{N}$, $\ell \in \mathbb{R}$ such that for all k,

$$\mathit{MaxTime}_{\mathcal{N}}(k) \leq \ell \cdot k^d. \tag{1}$$

Further $\mathcal{N}$ has *linear termination time* if Eq. (1) holds with $d = 1$. Even more fine-grained, $\mathcal{N}$ has *a-linear termination time* if Eq. (1) holds for $\ell = a$ and $d = 1$. Note that any net with a-linear termination time also has $(a + b)$-linear termination time for all $b \geq 0$. For ease of notation, we call workflow nets that have linear termination time *linear workflow nets*, and similarly for a-linear.

We define $a_{\mathcal{N}} \coloneqq \inf\{a \in \mathbb{R} \mid \mathcal{N} \text{ is } a\text{-linear}\}$. Note that in particular $\mathcal{N}$ is $a_{\mathcal{N}}$-linear (because the inequality in Eq. (1) is not strict) and that $a_{\mathcal{N}}$ is the smallest constant with this property.

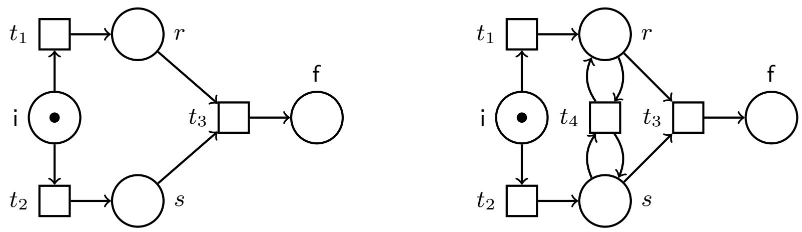


Fig. 2. Two workflow nets with the initial marking $\{\mathsf{i}\colon 1\}$. The workflow net on the left-hand side is terminating in linear time. The workflow net on the right-hand side is the same as the one on the left, but with one extra transition t_4. It is non-terminating.

Example 3. The net on the left-hand side of Figure 2 is terminating. For example, from $\{\mathsf{i}\colon 2\}$ all runs have length at most 3. It is easy to see that from $\{\mathsf{i}\colon k\}$ all runs have length at most $\frac{3}{2}k$ (*e.g.* the run $(t_1t_2t_3)^{\lfloor \frac{k}{2} \rfloor}$). The net has $a_{\mathcal{N}} = 3/2$.

The net on the right-hand side is non-terminating. From $\{\mathsf{i}\colon 2\}$, all runs of the form $t_1t_2t_4^*$ are enabled. Note that while the net is non-terminating, all runs from $\{\mathsf{i}\colon 1\}$ have length at most 1 (because t_3 and t_4 are never enabled). ◁

Our definition of termination time is particular to workflow nets, as there it is natural to have only i marked initially. It differs from the original definition of termination complexity in [13]. In [13] VASS are considered instead of Petri nets, and the initial marking is arbitrary. The termination complexity is measured in the size of the encoding of $\boldsymbol{m}$. The core difference is that in [13] it is possible to have a fixed number of tokens in some places, but arbitrarily many tokens in other places. In Section 3 we show an example that highlights the difference between the two notions. Our definition is a more natural fit for workflow nets, and will allow us to reason about soundness. Indeed, our particular definition of termination time allows us to obtain the coNP-completeness result of generalised soundness for linear workflow nets in Section 5.

3 A Dichotomy of Termination Time in Workflow Nets

Let us exhibit behaviour in Petri nets that cannot occur in workflow nets. Consider the net drawn in black in Figure 3 and a family of initial markings $\{\{p_1 : 1, s_1 : 1, b : n\} \mid n \in \mathbb{N}\}$. From the marking $\{p_1 : 1, s_1 : 1, b : n\}$, all runs have finite length, yet a run has length exponential in n. From the marking $\{p_1 : k, s_1 : 1, b : n\}$, the sequence $(t_1t_2)^k t_4 (t_3)^{2k} t_5$ results in the marking $\{p_1 : 2k, s_1 : 1, b : n-1\}$. Thus, following this pattern n times leads from $\{p_1 : 1, s_1 : 1, b : n\}$ to $\{p_1 : 2^n, s_1 : 1\}$. This behaviour crucially requires us to keep a single token in s_1, while having n tokens in b.
We can transform the net into a workflow net, as demonstrated by the colored part of Figure 3. However, observe that then

$$\{\mathsf{i} : 2\} \rightarrow^{t_i t_i t_4} \{p_1 : 2, s_1 : 1, s_2 : 1, b : 1\} \rightarrow^{t_1 t_2 t_3} \{p_1 : 2, s_1 : 1, s_2 : 1, b : 1, p_3 : 1\}.$$

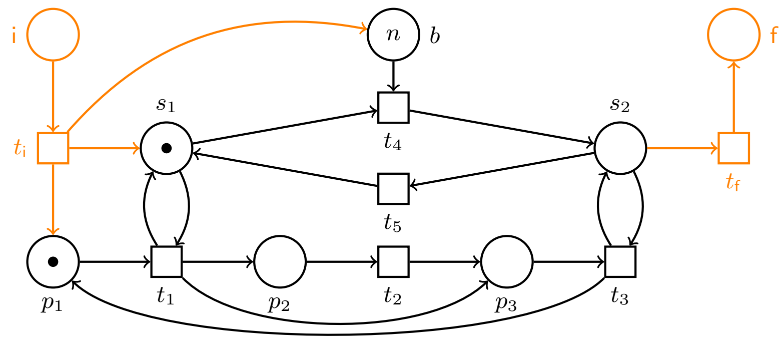


Fig. 3. In black: A Petri net $\mathcal{N}$ adapted from [29, Lemma 2.8]. It enables a run with length exponential in n from marking $\{p_1 : 1, s_1 : 1, b : n\}$. In color: Additional places and transitions, which make $\mathcal{N}$ a workflow net.

Note that the sequence $t_1t_2t_3$ strictly increased the marking. It can thus be fired arbitrarily many times, and the workflow net is non-terminating.

It turns out that, contrary to standard Petri nets, there exist no workflow nets with exponential termination time.[4] Instead, there is a dichotomy between non-termination and linear termination time.

Theorem 1. *Every workflow net $\mathcal{N}$ is either non-terminating or linear. Moreover, $MaxTime_{\mathcal{N}}(k) \leq ak$ for some $a \leq \|\mathcal{N}\|^{poly(|\mathcal{N}|)}$.*

As explained in Section 2.3 we can assume that $\mathcal{N}$ is nonredundant, *i.e.* for all $p \in P$ there exists $k \in \mathbb{N}$ such that $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m}$ with $\boldsymbol{m}(p) > 0$. The first important ingredient is the following lemma.

Lemma 2. *Let $\mathcal{N} = (P, T, F)$ be a nonredundant workflow net. Then $\mathcal{N}$ is non-terminating iff there exists a nonzero $\boldsymbol{R} \in \mathbb{N}^T$ such that $\Delta(\boldsymbol{R}) \geq \mathbf{0}$.*

Proof (sketch). The first implication follows from the fact that if we start from a big initial marking, then it is possible to fill every place with arbitrarily many tokens. In such a configuration any short run is enabled, so if there is a run with non-negative effect then it is further possible to repeat it infinitely many times. For the other implication we reason as follows. If there is an infinite run then by Dickson's lemma there are $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{N}^P$ such that for some k, it holds that $\{\mathsf{i}\colon k\} \to^{\pi} \boldsymbol{m} \to^{\rho} \boldsymbol{m}'$ and $\boldsymbol{m}' \geq \boldsymbol{m}$. But then $\Delta(\boldsymbol{R}_\rho) = \boldsymbol{m}' - \boldsymbol{m} \geq \mathbf{0}$. □

We define $\text{ILP}_{\mathcal{N}}$ with a $|T|$ dimensional vector of variables $\boldsymbol{x}$ as the following system of inequalities: $\boldsymbol{x} \geq \mathbf{0}$ and $\Delta(T)\boldsymbol{x} \geq \mathbf{0} - \{\mathsf{i}\colon \infty\}$.[5] The next lemma follows immediately from the definition of $\to_{\mathbb{Z}}$.

[4] This is caused by the choice of the family of initial configurations. Fixing the number of initial tokens in some places can be simulated by control states in the VASS model.

[5] This ∞ is syntactic sugar to omit the inequality for the place i. Formally $\Delta(T)$ and $\boldsymbol{x}$ should be projected to ignore i.

Lemma 3. *[Adapted from Claim 5.7 in [10]] For every $k \in \mathbb{N}$, $\boldsymbol{m} \in \mathbb{N}^P$, and a run π, it holds that $\{\mathsf{i}\colon k\} \rightarrow^{\pi}_{\mathbb{Z}} \boldsymbol{m}$ iff $\boldsymbol{R}_\pi$ is a solution to ILP$_\mathcal{N}$ with the additional constraint $\sum_{i=1}^{|T|} \Delta(t_i)(\mathsf{i}) \cdot \boldsymbol{R}_\pi(t_i) \geq -k$.*

Proof (Sketch for Theorem 1). Because of Lemma 3 the Parikh image of every run (in $\bigcup_{k\in\mathbb{N}} \mathrm{Runs}^k_\mathcal{N}$) is a solution $\boldsymbol{R} \in \mathbb{N}^T$ of $\Delta(T)\boldsymbol{R} \geq -\{\mathsf{i}\colon \infty\}$. So, we consider a set of solutions of the system of inequalities $\Delta(T)\boldsymbol{R} \geq -\{\mathsf{i}\colon \infty\}$. It is a linear set, so the sum of two solutions is again a solution and any solution can be written as a sum of small solutions with norm smaller than some $c \in \mathbb{N}$. For such small solutions, the length of any corresponding run is at most $|T| \cdot c$. Now observe that if the workflow is terminating then there is no $\boldsymbol{R} \in \mathbb{N}^T$ such that $\Delta(T)\boldsymbol{R} \geq \boldsymbol{0}$, because of Lemma 2. But it holds that $\Delta(\boldsymbol{R})(\mathsf{i}) \leq -1$ for any solution $\boldsymbol{R}$, so in particular for all small solutions. Let us take a run $\pi \in \mathrm{Runs}^k_\mathcal{N}$. We decompose $\boldsymbol{R}_\pi$ as a finite sum $\sum_i^\ell \boldsymbol{R}_i$ where $\boldsymbol{R}_i$ are from the set of small solutions. We have $-k \leq \Delta(\boldsymbol{R}_i)(\mathsf{i}) = \sum_i^\ell \Delta(\boldsymbol{R}_i)(\mathsf{i}) \leq \sum_i^\ell -1 = -\ell$. Recall that the norm of small solutions is bounded by c. It follows that the length of the run π is bounded by $\ell \cdot |T| \cdot c \leq k \cdot |T| \cdot c$. So the workflow is $|T| \cdot c$-linear.

4 Refining Termination Time

Recall that $a_\mathcal{N}$ is the the smallest constant such that $\mathcal{N}$ is $a_\mathcal{N}$-linear. In this section, we are interested in computing $a_\mathcal{N}$. This number is interesting, as it can give insights into the shape and complexity of the net, *i.e.* a large $a_\mathcal{N}$ implies complicated runs firing transitions several times, while a small $a_\mathcal{N}$ implies some degree of choice, where not all transitions can be fired for each initial token.

The main goal of this section is to show an algorithm for computing $a_\mathcal{N}$. Our algorithm handles the more general class of *aggregates* on workflow nets, and we can compute $a_\mathcal{N}$ as such an aggregate. More formally, let $\mathcal{N} = (P, T, F)$ be a workflow net. An aggregate is a linear map $f : \mathbb{Q}^T \to \mathbb{Q}$. The aggregate of a (continuous) run is the aggregate of its Parikh image, that is $f(\pi) \coloneqq f(\boldsymbol{R}_\pi)$.

Example 4. Consider the aggregate $f_{all}(\pi) \coloneqq \sum_{t\in T} \boldsymbol{R}_\pi(t) = |\pi|$, which computes the number of occurrences of all transitions. Let us consider two other natural aggregates. The aggregate $f_t(\pi) \coloneqq \boldsymbol{R}_\pi(t)$ computes the number of occurrences of transition t, and $f_p(\pi) \coloneqq \sum_{t\in T} \Delta(t)(p)\cdot\boldsymbol{R}_\pi(t)$ computes the number of tokens added to place p. Another use for aggregates is counting transition, but with different weights for each transition, thus simulating *e.g.* different costs. ◁

Given a workflow net $\mathcal{N}$ and an aggregate f we define

$$\sup_{f,\mathcal{N}} = \sup\left\{\frac{f(\pi)}{k} \mid k \in \mathbb{N}_{>0},\ \pi \in \mathrm{Runs}^k_\mathcal{N}\right\}. \tag{2}$$

Let us justify the importance of this notion by relating it to $a_\mathcal{N}$.

Proposition 1. *Let $\mathcal{N}$ be a linear workflow net. Then $a_\mathcal{N} = sup_{f_{all},\mathcal{N}}$.*

Proof. Recall that $a_{\mathcal{N}}$ is the smallest number a such that $|\pi| \leq a \cdot k$ for all $k \in \mathbb{N}_{>0}$ and $\pi \in \text{Runs}^k_{\mathcal{N}}$. Equivalently, $\frac{|f_{all}(\pi)|}{k} \leq a$. Thus by definition $\sup_{f_{all},\mathcal{N}} \leq a_{\mathcal{N}}$, and the inequality cannot be strict since $a_{\mathcal{N}}$ is the smallest number with this property. □

Theorem 2. *Consider a workflow net $\mathcal{N}$ and an aggregate f. The value $sup_{f,\mathcal{N}}$ can be computed in polynomial time.*

Corollary 1. *Let $\mathcal{N} = (P, T, F)$ be a linear workflow net. The constant $a_{\mathcal{N}}$ can be computed in polynomial time.*

In practice, we can use an LP solver to compute the constant $a_{\mathcal{N}}$. The algorithm is based on the fact that continuous reachability for Petri nets is in polynomial time [20,7]. We formulate a lemma that relates the values of aggregates under the continuous and standard semantics.

Lemma 4. *Let $\mathcal{N}$ be a Petri net and f be an aggregate.*

1. *Let $\pi \in Runs^k_{\mathcal{N}}$. Then $1/k \cdot \pi \in CRuns^1_{\mathcal{N}}$ and $f(1/k \cdot \pi) = f(\pi)/k$.*
2. *Let $\pi_c \in CRuns^1_{\mathcal{N}}$. There are $k \in \mathbb{N}$ and $\pi \in Runs^k_{\mathcal{N}}$ with $f(\pi_c) = f(\pi)/k$.*

Proof. Both items are simple consequences of Lemma 1 and the linearity of aggregates. Note that for (2), if $\pi_c = \beta_1 t_1 \dots \beta_n t_n$ then it suffices to define k such that $\beta_i \cdot k \in \mathbb{N}$ for all $i \in \{1, \dots, n\}$. □

From the above lemma we immediately conclude the following.

Corollary 2. *It holds that $sup_{f,\mathcal{N}} = sup\{f(\pi_c) \mid \pi_c \in CRuns^1_{\mathcal{N}}\}$.*

Proof (The proof of Theorem 2). We use Corollary 2 and conclude that we have to compute $sup\{f(\pi_c) \mid \pi_c \in \text{CRuns}^1_{\mathcal{N}}\}$. Let $S = \{\boldsymbol{R}_{\pi_c} \mid \pi_c \in \text{CRuns}^1_{\mathcal{N}}\}$. As $f(\pi)$ is defined as $f(\boldsymbol{R}_\pi)$, we reformulate our problem to compute $sup\{f(\boldsymbol{v}) \mid \boldsymbol{v} \in S\}$. Since f is a continuous function, it holds that $sup\{f(\boldsymbol{v}) \mid \boldsymbol{v} \in S\} = sup\{f(\boldsymbol{v}) \mid \boldsymbol{v} \in \overline{S}\}$. Let us define $\text{LP}_{\text{f},\mathcal{N}}$ as an LP with variables $\boldsymbol{x} \coloneqq x_1, \dots, x_{|T|}$ and constraints $\Delta(T)\boldsymbol{x} \geq -\{\mathsf{i}\colon 1\}$ and $\boldsymbol{x} \geq \boldsymbol{0}$.

Claim 1. It holds that $\boldsymbol{v} \in \overline{S}$ if and only if $\boldsymbol{v}$ is a solution to $\text{LP}_{\text{f},\mathcal{N}}$.

We postpone the proof of Claim 1. Claim 1 allows us to rephrase the computation of $sup\{f(\boldsymbol{v}) \mid \boldsymbol{v} \in \overline{S}\}$ as an $\text{LP}_{\text{f},\mathcal{N}}$ where we want to maximise $f(\boldsymbol{v})$, which can be done in polynomial time. □

What remains is the proof of Claim 1. It constitutes the remaining part of this Section. The claim is a special case of the forthcoming Lemma 8. Its formulation and proof require some preparation.

Definition 1. *A workflow net is* good *for a set of markings $M \subseteq \mathbb{Q}^P_{\geq 0}$ if for every place p there are markings $\boldsymbol{m}, \boldsymbol{m}'$ and continuous runs π and π' such that $\boldsymbol{m}(p) > 0$, $\boldsymbol{m}' \in M$, and $\{\mathsf{i}\colon 1\} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m} \to^{\pi'}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}'$.*

The notion of being good for a set of markings is a refined concept of nonredundancy. The nonredundancy allow us to mark every place. But if, after marking the place, we want to continue the run and reach a marking in a specific set of markings $M \subseteq \mathbb{Q}_{\geq 0}^P$, then we don't know if the given place can be marked. This motivates Definition 1.

Example 5. Let us consider a workflow net depicted on Fig. 4. It is nonredundant, as every place can be marked. But it is not good for $\{\mathsf{f}\colon 1\}$ as there is no continuous run to the marking $\{\mathsf{f}\colon 1\}$. In the initial marking the only enabled transition is t_1 but firing βt_1 for any $\beta \in \mathbb{Q}_{\geq 0}$ reduce the total amount of tokens in the net. The lost tokens can not be recrated so it is not possible to reach $\{\mathsf{f}\colon 1\}$.

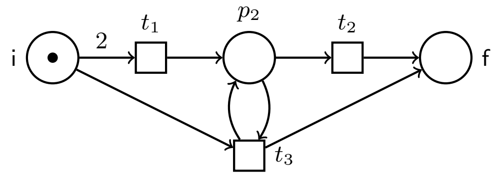


Fig. 4. A Petri net with places p_1, p_2, p_3 and transitions t_1, t_2, t_3. Marking $\{\mathsf{i}\colon 1\}$ is drawn.

The important fact is as follows:

Lemma 5. *Let $M \subseteq \mathbb{Q}_{\geq 0}^P$ be a set of solutions of some LP. Then testing if a net is good for M can be done in polynomial time.*

Lemma 6. *Suppose a workflow net $\mathcal{N}$ is good for $M \subseteq \mathbb{Q}_{\geq 0}^P$ and M is a convex set. Then there is a marking $\boldsymbol{m_+}$ such that $\boldsymbol{m_+}(p) > 0$ for every $p \in P$ and there are continuous runs π, π', and a marking $\boldsymbol{m_f} \in M$ such that $\{\mathsf{i}\colon 1\} \to_{\mathbb{Q}_{\geq 0}}^{\pi} \boldsymbol{m_+} \to_{\mathbb{Q}_{\geq 0}}^{\pi'} \boldsymbol{m_f}$.*

Informally, we prove it by taking a convex combination of a $|P|$ runs one for each $p \in P$. The last bit needed for the proof of Lemma 8 is the following lemma, shown in [20].

Lemma 7 ([20], Lemma 13). *Let $\mathcal{N}$ be a Petri net. Consider $\boldsymbol{m_0}$, $\boldsymbol{m} \in \mathbb{N}^P$ and $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^T$ such that:*

- $\boldsymbol{m} = \boldsymbol{m_0} + \Delta(\boldsymbol{v})$;
- $\forall p \in {}^\bullet\boldsymbol{v}\colon\ \boldsymbol{m_0}(p) > 0$;
- $\forall p \in \boldsymbol{v}^\bullet\colon\ \boldsymbol{m}(p) > 0$.

Then there exists a finite continuous run π such that $\boldsymbol{m_0} \to_{\mathbb{Q}_{\geq 0}}^{\pi} \boldsymbol{m}$ and $\boldsymbol{R}_\pi = \boldsymbol{v}$.

Lemma 8. *Suppose M is a convex set of markings over $\mathbb{Q}_{\geq 0}^P$ and that the workflow net is good for M. Let S be the set of Parikh images of continuous runs that start in $\{\mathsf{i}\colon 1\}$ and end in some marking $\boldsymbol{m'} \in M$ i.e.*

$$S \coloneqq \{\boldsymbol{R}_\pi \mid \exists_{\pi \in CRuns^1_{\mathcal{N}}} \exists_{\boldsymbol{m'} \in M} \text{ such that } \{\mathsf{i}\colon 1\} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}\}.$$

Then $\boldsymbol{v} \in \overline{S}$ if and only if there is a marking $\boldsymbol{m} \in M$ such that $\Delta(T)\boldsymbol{v} = \boldsymbol{m} - \{\mathsf{i}\colon 1\}$.

In one direction the proof of the lemma is trivial, in the opposite direction, intuitively, we construct a sequence of runs with Parikh images converging to $\boldsymbol{v}$. The Lemma 6 is used to put ε in every place (for $\varepsilon \to 0$) and Lemma 7 to show that there are runs with the Parihk image equal $\varepsilon\boldsymbol{x} + (1-\varepsilon)\boldsymbol{v}$ for some $\boldsymbol{x}$ witnessing Lemma 6. We are ready to prove Claim 1.

Claim 1. It holds that $\boldsymbol{v} \in \overline{S}$ if and only if $\boldsymbol{v}$ is a solution to $\mathrm{LP}_{\mathsf{f},\mathcal{N}}$.

Proof. Let M be the set of all markings over $\mathbb{Q}_{\geq 0}^P$, which clearly is convex. As $\mathcal{N}$ is nonredundant we know that every place can be marked via a continuous run, and because M is the set of all markings we conclude that $\mathcal{N}$ is good for M according to Definition 1. Thus M satisfies the prerequisites of Lemma 8. It follows that $\overline{S}$ is the set of solutions of a system of linear inequalities. Precisely, $\boldsymbol{v} \in \overline{S}$ if and only if there is $\boldsymbol{m} \in \mathbb{Q}_{\geq 0}^P$ such that $\Delta(T)\boldsymbol{v} \geq \boldsymbol{m} - \{\mathsf{i}\colon 1\}$ and $\boldsymbol{v} \geq 0$, which is equivalent to $\Delta(T)\boldsymbol{v} \geq -\{\mathsf{i}\colon 1\}$ and $\boldsymbol{v} \geq 0$, as required. ☐

5 Soundness in Terminating Workflow Nets

The dichotomy between linear termination time and non-termination shown in Section 3 yields an interesting avenue for framing questions in workflow nets. We know that testing generalised soundness is PSPACE-complete, but the lower bound in [10] relies on a reset gadget which makes the net non-terminating. Indeed, it turns out that the problem is simpler for linear workflow nets.

Theorem 3. *Generalised soundness is coNP-complete for linear workflow nets.*

A marking $\boldsymbol{m}$ is called a *deadlock* if $\mathrm{Runs}^{\boldsymbol{m}}_{\mathcal{N}} = \emptyset$. To help prove the coNP upper bound, let us introduce a lemma.

Lemma 9. *Let $\mathcal{N}$ be a terminating nonredundant workflow net. Then $\mathcal{N}$ is not generalised sound iff there exist $k \in \mathbb{N}$ and a marking $\boldsymbol{m} \in \mathbb{N}^P$ such that $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$, $\boldsymbol{m}$ is a deadlock and $\boldsymbol{m} \neq \{\mathsf{f}\colon k\}$. Moreover, if $\|\mathcal{N}\| \leq 1$ then $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$ can be replaced with $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Q}} \boldsymbol{m}$.*

The last part of the lemma is not needed for the theoretical results, but it will speed up the implementation in Section 7. We can now show Theorem 3.

Proof (of the coNP upper bound in Theorem 3). Let $\mathcal{N} = (P, T, F)$ and denote $T = \{t_1, \ldots, t_n\}$. By Lemma 9 $\mathcal{N}$ is not generalised sound iff there are $k \in \mathbb{N}$ and $\boldsymbol{m} \in \mathbb{N}^P$ such that $\{\mathsf{i}\colon k\} \rightarrow_{\mathbb{Z}}^* \boldsymbol{m}$, $\boldsymbol{m}$ is a deadlock and $\boldsymbol{m} \neq \{\mathsf{f}\colon k\}$. We can reduce the property to an ILP. First, the procedure guesses $|T|$ places $p_1, \ldots, p_n \in P$ (one for each transition). For each transition t_i, place p_i will prohibit firing t_i by not being marked with enough tokens. We create $\mathrm{ILP}_{\mathcal{N},p_1,\ldots,p_n}$, which is very similar to $\mathrm{ILP}_{\mathcal{N}}$ (see Section 3), but adds additional constraints. They state that $(\Delta(T)\boldsymbol{x})(p_j) - {}^\bullet t_j(p_j) < 0$ for every $1 \leq j \leq n$.

Let us show that there are $p_1, \ldots, p_n$ such that $\mathrm{ILP}_{\mathcal{N},p_1,\ldots,p_n}$ has a solution iff there exist k and a deadlock $\boldsymbol{m}$ such that $\{\mathsf{i}\colon k\} \rightarrow_{\mathbb{Z}}^* \boldsymbol{m}$. Indeed, let $x_1, \ldots, x_n$ be a solution of $\mathrm{ILP}_{\mathcal{N},p_1,\ldots,p_n}$. We denote $k = -\sum_{i=1}^{n} \Delta(t_i)(\mathsf{i}) \cdot x_i$ and $\boldsymbol{m} = \{\mathsf{i}\colon k\} + \sum_{i=1}^{n} \Delta(t_i) \cdot x_i$. It is clear that $\{\mathsf{i}\colon k\} \rightarrow_{\mathbb{Z}}^* \boldsymbol{m}$. The new constraints ensure that for each $t_i \in T$ there exists $p_i \in P$ such that ${}^\bullet t_i(p_i) > \boldsymbol{m}(p_i)$, thus $\boldsymbol{m}$ is a deadlock.

To encode the requirement that $\boldsymbol{m} \neq \{\mathsf{f}\colon k\}$, note that there are three cases, either $\boldsymbol{m}(k) \leq k - 1$, $\boldsymbol{m}(k) \geq k + 1$, or $\boldsymbol{m}(k) = k$ but $\boldsymbol{m} - \{\mathsf{f}\colon k\} \geq 0$. We guess which case occurs, and add the constraint for that case to $\mathrm{ILP}_{\mathcal{N},p_1,\ldots,p_n}$. □

The lower bound can be proven using a construction presented in [11, Theorem 2] to show a problem called continuous soundness on acyclic workflow nets is coNP-hard. We say that a workflow net is *continuously sound* iff for all $\boldsymbol{m}$ such that $\{\mathsf{i}\colon 1\} \rightarrow_{\mathbb{Q}_{\geq 0}}^* \boldsymbol{m}$, it holds that $\boldsymbol{m} \rightarrow_{\mathbb{Q}_{\geq 0}}^* \{\mathsf{f}\colon 1\}$. The reduction can be used as is to show that generalised soundness of nets with linear termination time is coNP-hard, but the proof differs slightly. See the appendix for more details.

6 Termination Time and Concurrent Semantics

Note that in Petri nets, transitions may be fired concurrently. Thus, in a sense, our definition of termination time may overestimate the termination time.

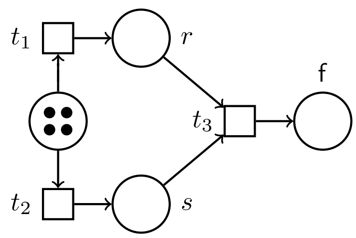
In this section we investigate parallel executions for workflow nets. Whereas the termination time is focused on the worst case sequential execution, now we are interested in finding the best case parallel executions. Thus, we provide an optimistic lower bound on the execution time to contrast the pessimistic upper bound investigated in Section 3 and Section 4.

Definition 2. *Given a Petri net $\mathcal{N}$ let $\pi = t_1 t_2 \ldots t_n \in Runs_{\mathcal{N}}^k$ for some $k \in \mathbb{N}$. A* block *in π is a subsequence of π, i.e. $t_a, \ldots, t_b$ for some $1 \leq a \leq b \leq n$. We define the* parallel execution *of π with respect to k as a decomposition of π into blocks $\pi = \pi_1 \pi_2 \ldots \pi_\ell$ such that*

1. *all transitions are pairwise different in a single block; and*
2. ${}^\bullet\boldsymbol{R}_{\pi_i} \leq \{\mathsf{i}\colon k\} + \sum_{j<i} \Delta(\pi_j)$ *for every $1 \leq i \leq \ell$.*

The execution time *of a parallel execution is denoted as $exec(\pi_1 \pi_2 \ldots \pi_\ell) \coloneqq \ell$.*

Example 6.



We consider parallel executions of the run $t_1t_2t_1t_2t_3t_3$ with respect to 4 initial tokens. The run can be decomposed into $(t_1t_2)(t_1t_2)(t_3)(t_3)$ but also into $(t_1)(t_2t_1)(t_2t_3)(t_3)$. Both executions have execution time 4. The parallel execution $(t_1t_2)(t_1t_2t_3)(t_3)$ has execution time 3. ◁

We are interested in finding the parallel executions of a run that minimise the execution time. It turns out that the so-called *greedy parallel execution* is such a minimal parallel execution. Given π and k it is defined inductively on the prefix of π. Suppose we already have some blocks $\pi_1 \dots \pi_{i-1}$. To construct block π_i, we simply choose the maximal sequence of transitions immediately following the last block π_{i-1} that satisfies the two conditions of Definition 2. In particular the last partition in Example 6 is the greedy parallel execution.

Lemma 10. *Consider a run π and $k \in \mathbb{N}$. The greedy parallel execution of π has the smallest execution time among all parallel executions of π with respect to k.*

Consider a workflow net $\mathcal{N}$ with the initial marking $\{\mathsf{i}\colon k\}$. Let $S_k \coloneqq \{\pi \mid \{\mathsf{i}\colon k\} \rightarrow^{\pi} \{\mathsf{f}\colon k\}\}$. We define $MinTime_{\mathcal{N}}(k)$ as the minimal execution time among parallel executions of runs in S_k. If $S_k = \emptyset$ then $MinTime_{\mathcal{N}}(k) = +\infty$.

Lemma 11. *Let $\mathcal{N}$ be a workflow net and let $k, x \in \mathbb{N}$. Deciding whether $MinTime_{\mathcal{N}}(k) \leq x$ is PSPACE-hard even if we fix $k = 1$.*

As computing $MinTime_{\mathcal{N}}(k)$ is computationally hard, we modify the question and ask about the asymptotic behaviour (similarly to Section 4). Thus, we are interested in computing $\lim_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k}$. The problem is well defined as the limit exists (Lemma 15 in Appendix D). This is interesting as $\lim_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k}$ corresponds to the average processing time of a single token when the workflow runs (informally speaking) on its maximal efficiency.

Theorem 4. *For a given nonredundant, generalised sound workflow net*[6] *$\mathcal{N}$ we can compute $lim_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k}$ in polynomial time.*

Proof (A sketch of the proof). The main idea relies on the continuous semantics, similarly to the proof of Theorem 2. We show that the limit is equal to the infimum over execution times[7] of continuous runs $\{\mathsf{i}\colon 1\} \rightarrow_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$. Then we prove the following claim.

[6] These assumptions can be relaxed to *a net good for* $\{\mathsf{f}\colon 1\}$, see Definition 1 in Appendix B.

[7] For a suitably defined parallel execution and execution time of continuous runs.

Claim 2. Let $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^T$. Let $S_{\boldsymbol{v}} = \{\pi \mid \{\mathsf{i}\colon 1\} \rightarrow^{\pi}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ and $\boldsymbol{R}_\pi = \boldsymbol{v}\}$. If $S \neq \emptyset$ then the infimum over optimal execution time of runs in $S_{\boldsymbol{v}}$ equals $\|\boldsymbol{v}\|$.

Let S be the set of Parikh images of continuous runs from $\{\mathsf{i}\colon 1\}$ to $\{\mathsf{f}\colon 1\}$. We define $f : \overline{S} \rightarrow \mathbb{Q}_{\geq 0}$ such that $f(\boldsymbol{v}) = \|\boldsymbol{v}\|$. Thus we can reformulate the problem as computing $inf\{f(\boldsymbol{v}) \mid \boldsymbol{v} \in S\}$. The function f is continuous, thus we can reformulate further as compute $inf\{f(\boldsymbol{v}) \mid \boldsymbol{v} \in \overline{S}\}$. The function f is not linear on $\overline{S}$, but it is piecewise linear. We define $\overline{S_t} \subseteq \overline{S}$ for $t \in T$ as follows $\overline{S_t} = \{\boldsymbol{v} \mid \boldsymbol{v} \in \overline{S}$ and $\boldsymbol{v}(t) \geq \boldsymbol{v}(t')$ for all $t' \in T\}$. Observe that f is linear over $\overline{S_t}$ for every $t \in T$ and that $\overline{S} = \bigcup_{t \in T} \overline{S_t}$. Thus we can rephrase our problem as computing the minimum over the set $\{inf\{\boldsymbol{v}(t) \mid \boldsymbol{v} \in \overline{S_t}\} \mid t \in T\}$.

Thus it is sufficient to show that $inf\{\boldsymbol{v}(t) \mid \boldsymbol{v} \in \overline{S_t}\}$ can be computed in polynomial time for any $t \in T$. Lemma 8 in Appendix B allows us to characterize $\overline{S}$ as follows: $\boldsymbol{v} \in \overline{S}$ iff $\Delta(T)\boldsymbol{v} = \{\mathsf{f}\colon 1\} - \{\mathsf{i}\colon 1\}$ and $\boldsymbol{v} \geq \boldsymbol{0}$. In consequence, $\overline{S_t}$ can be characterized as the set of solutions of the following system of inequalities

$$\Delta(T)\boldsymbol{v} = \{\mathsf{f}\colon 1\} - \{\mathsf{i}\colon 1\} \text{ and } \boldsymbol{v} \geq \boldsymbol{0} \text{ and } \boldsymbol{v}(t) \geq \boldsymbol{v}(t') \text{ for all } t' \in T.$$

This allows us to capture $\{inf\{\boldsymbol{v}(t) \mid \boldsymbol{v} \in \overline{S_t}\} \mid t \in T\}$ as an LP problem which can be solved in polynomial time. □

7 Experimental Evaluation

We have implemented prototypes of several procedures outlined in the paper, namely procedures to 1) decide termination; 2) decide soundness for terminating nets; 3) compute $a_{\mathcal{N}}$ for terminating nets; and 4) compute $MinTime_{\mathcal{N}}(1)$, $MaxTime_{\mathcal{N}}(1)$, and decide 1-soundness for nets with known $a_{\mathcal{N}}$. The idea behind all procedures is to use our results to encode the properties in LPs/ILPs. To solve these programs, we utilize the MILP solver *Gurobi* [25].

For 1), recall Lemma 2, which states that non-termination of a workflow net $\mathcal{N}$ is equivalent to the existence of a Parikh image $\boldsymbol{R} \in \mathbb{N}^T$ with $\Delta(\boldsymbol{R}) \geq \boldsymbol{0}$. We can instead search for $\boldsymbol{R} \in \mathbb{Q}^T$, as any solution could be scaled up to an integral one. Thus, we can encode this condition as an LP in a straightforward manner, and decide termination in polynomial time.[8]

For 2), we essentially use $\mathrm{ILP}_{\mathcal{N},p_1,\ldots,p_n}$, as defined in the proof of Theorem 3. A solution to $\mathrm{ILP}_{\mathcal{N},p_1,\ldots,p_n}$ yields a run π such that there exists $k \in \mathbb{N}$ with $\{\mathsf{i}\colon k\} \rightarrow^{\pi}_{\mathbb{Z}} \boldsymbol{m}$, where $\boldsymbol{m}$ is a deadlock.

We also consider continuous instead of integral variables. Then solutions relate to runs over $\rightarrow^*_{\mathbb{Q}}$ instead. As hinted at in the last sentence of Lemma 9, both variants yield equivalent results on nets without arc weights, *i.e.* $\|\mathcal{N}\| \leq 1$. However, continuous variables are generally easier to handle for MILP solvers. For brevity, by *integer deadlocks* we refer to the approach using integer variables, and by *continuous deadlocks* to the approach with continuous variables.

For 3), recall the LP given in Claim 1. We can use it to compute $\sup_{f,\mathcal{N}}$ for any aggregate $\mathcal{N}$, so in particular we can use it to compute $\sup_{f_{all},\mathcal{N}}$, which

[8] This observation and the general approach comes from [31].

is equal to $a_{\mathcal{N}}$ by Equation (2). Here, it only remains to mention that Gurobi allows not only checking feasibility of systems of linear inequalities, but further allows optimizing an objective value, as required by the LP.

For 4), note that if we have the bound $a_{\mathcal{N}}$ on the length of runs from $\{\mathsf{i}\colon 1\}$, we can check properties by unrolling runs. The intuition is as follows. We have $a_{\mathcal{N}} \cdot |T|$ integer variables. For step j of the run, we have variables $x_{1,j}, x_{2,j}, \ldots, x_{|T|,j}$. The variables for a step encode which transition(s) are fired in that step. We ensure that we encode a run by requiring $\sum_{i=1}^{|T|} x_{i,j} \leq 1$ for all $j \in [1..a_{\mathcal{N}}]$. We use integer variables, so either one or no transition is fired in each step.

Alternatively, we encode a parallel execution by imposing the requirements of Definition 2 on steps. By further specifying that for all $j \in [1..a_{\mathcal{N}}]$, it holds that $\{\mathsf{i}\colon 1\} + \sum_{j'=0}^{j} \sum_{i=1}^{|T|} \Delta(t_i) x_{i,j'} \geq \mathbf{0}$, thus the marking reached so far after each step is nonnegative. To compute $MinTime_{\mathcal{N}}(1)/MaxTime_{\mathcal{N}}(1)$, we minimise/maximise the number of blocks/steps with non-zero transition variables. For 1-soundness, we require reaching a deadlock different from $\{\mathsf{f}\colon 1\}$.

Our prototype is implemented in C#. All experiments were run on an 8-Core Intel® Core™ i7-7700 CPU @ 3.60GHz with Ubuntu 18.04. We limited memory to ~8GB. The time was limited to $60s$ for checking termination and generalised soundness as well as for computing $a_{\mathcal{N}}$. It was limited to $15s$ for computing $MinTime_{\mathcal{N}}(1), MaxTime_{\mathcal{N}}(1)$ and for checking 1-soundness.

7.1 Benchmark Suite

We use a popular benchmark suite of 1386 free-choice nets originating from models created in the IBM WebSphere Business Modeler. The instances were originally introduced in [19] and have frequently been studied since, see [14,40,39]. The nets use a slightly different formalisation of workflow nets that allow multiple final places, which can be transformed to standard workflow nets using a technique from [30]. This technique adds transitions, thus can increase $a_{\mathcal{N}}$, $MinTime_{\mathcal{N}}$ and $MaxTime_{\mathcal{N}}$. Unfortunately, 4 instances cannot be transformed to workflow nets with this technique, so we remove them. We also apply a set of well-known reduction rules from [14] that reduce the size of instances while keeping all types of soundness intact, and remove instances that are trivially sound after reduction. These rules never increase $a_{\mathcal{N}}$. While they in theory could increase $MinTime_{\mathcal{N}}$, this does not happen on our benchmarks. Due to the nature of the reduction rules, it may not be appropriate to run them before analyzing $MinTime_{\mathcal{N}}, MaxTime_{\mathcal{N}}(1)$ and $a_{\mathcal{N}}$, since these numbers then give no information about the original workflow. Thus we only run experiments on the reduced instances when we check soundness and termination.

In total, we are left with 1382 unreduced and 740 non-trivial reduced instances. Statistics about the sizes of the workflow nets can be seen in the columns under Net Size in Figure 5. The reduced nets are much smaller than the unreduced ones, even when the nets are not reduced to the trivial net.

		Net Size		Analysis Time (in ms)			
		$\|P\|$	$\|T\|$	Termination	Continuous Deadlock	Integer Deadlock	Continuous Soundness [11]
Unreduced instances	Mean	48.78	33.07	4.09	7.17	12.8	2022.54
	Median	37	26	3	5	11	88
	Max	274	285	23	85	88	55707
Reduced instances	Mean	7.43	5.49	2.99	2.3	8.88	44.51
	Median	6	5	3	2	8	33
	Max	33	22	5	18	39	99

		Total	Deadlocking (Not generalised sound)
Unreduced instances	Terminating	1262	523
	Nonterm.	120	53
Reduced instances	Terminating	694	536
	Nonterm.	46	23

Fig. 5. Top: Statistics on the net size, and analysis times for deciding termination, and checking generalised soundness via deadlocks and continuous soundness. Bottom: Statistics on the number of terminating/non-terminating and deadlocking/non-deadlocking (thus generalised unsound/generalised sound) nets.

7.2 Termination and Deadlocks

The time taken to decide termination is shown in the column labelled "Termination" in the top table of Figure 5. The numbers of nets that are terminating and non-terminating are shown in the bottom table of Figure 5. Among both the unreduced and reduced instances, the vast majority are terminating (about 90%). Note that the reduction rules can remove nontermination, even when they do not make the net nontrivial, thus the prevalence of terminating instances is even stronger among the reduced instances. In terms of analysis time, termination can be decided in under $25ms$ for all instances, with a median of $3ms$.

The top of Figure 5 shows the analysis times for generalised soundness. We use three algorithms. Columns "Continuous Deadlock" and "Integer Deadlock" show results for our two proposed approaches, and column "Continuous Soundness" shows the performance of a state-of-art approach [11] for deciding generalised soundness. Note that both approaches may claim an unsound workflow net to be sound, but they are precise on different classes of nets. The absence of integer deadlocks is equivalent to generalised soundness on terminating nets, see Lemma 9. Similarly, continuous soundness is equivalent to generalised soundness on free-choice nets [11].

In practice, it turns out that our approach for checking the absence of integer deadlocks is faster than the existing approach using continuous soundness on every single instance. Continuous soundness times out on 215 of the unreduced instances (not listed in the table), but neither of the approaches utilizing deadlocks times out on any instance. The performance of continuous soundness is not surprising: continuous soundness is checked by passing an ∃∀-formula from FO($\mathbb{Q}, <, +$) to an SMT solver. Quantifier alternation increases the complexity of validating such formulas [24]. In comparison, our check for integer deadlocks is implemented using standard ILP techniques, and thus an existential formula.

The bottom shows how many nets are non-terminating, as well as how many are deadlocking (thus not generalised sound). Recall that integer deadlocks and continuous deadlocks are equivalent for nets without arc weights, which all of our nets are. Both types of deadlocks are fast to compute, taking less than $90ms$ on each instance. In practice, checking for continuous deadlocks may be useful even for nets with arc weights, since their absence also proves the absence of integer deadlocks. About 50% of the unreduced instances and roughly 75% of the reduced instances are deadlocking. Note that the reduction rules can only make sound instances trivial, which are by definition not able to reach a deadlock.

7.3 $a_{\mathcal{N}}$, $MinTime_{\mathcal{N}}(1)$ and $MaxTime_{\mathcal{N}}(1)$

The top of Figure 6 the distribution of $a_{\mathcal{N}}$. This number depends on the number of transitions, so is hard to put into context. We instead display $\mathfrak{L} \coloneqq {}^{a_{\mathcal{N}}}/_{|T|}$. Intuitively, that number is an upper bound on the average of how many times each transition can be fired per initial tokens. For example, a net with $\mathfrak{L} = 1$ likely is linear, *i.e.* each transition can be fired only once per initial token, while nets with $\mathfrak{L} >> 1$ may exhibit more complex behaviour, and nets with $\mathfrak{L} << 1$ may exhibit high degrees of choice, where runs only visit a part of the net. We group nets with similar $\mathfrak{L}$ to give an idea of the distribution of the values of $\mathfrak{L}$ across instances. Our computation of $a_{\mathcal{N}}$ ran out of memory on 8 nets, so the figure displays only 1254 nets. Most nets have $\mathfrak{L} \leq 1$, with a significant number having in particular $\mathfrak{L} = 1$. The maximal $\mathfrak{L}$ is 5.83 among unreduced and 4.33 among reduced instances, while the minimal $\mathfrak{L}$ is 0.17 and 0.11 respectively.

To display $MinTime_{\mathcal{N}}(1)$ and $MaxTime_{\mathcal{N}}(1)$, we also divide them by the number of transitions, as we did for $a_{\mathcal{N}}$. We write $\mathfrak{T}_{Min} \coloneqq {}^{MinTime_{\mathcal{N}}(1)}/_{|T|}$ and $\mathfrak{T}_{Max} \coloneqq {}^{MaxTime_{\mathcal{N}}(1)}/_{|T|}$. We are mostly interested in their difference $\mathfrak{D} \coloneqq \mathfrak{T}_{Max} - \mathfrak{T}_{Min}$. For nets with large $\mathfrak{D}$, the difference between the pessimistic sequential and optimistic parallel execution time is large, thus they might allow a high degree of parallelism. On the contrary, if nets have very small $\mathfrak{D}$, they have a sequential structure. We again group nets with similar $\mathfrak{D}$, as we did for $\mathfrak{L}$ above. The results of the analysis are shown in the middle table of Figure 6.

As we divide by $|T|$ in the definition of $\mathfrak{D}$, it would be unusual for it to take on huge values, and indeed all nets have $\mathfrak{D} < 1$. Note that even $\mathfrak{D} = 0.5$ is significant, as it means that $MinTime_{\mathcal{N}}(1)$ and $MaxTime_{\mathcal{N}}(1)$ differ by half the number of transitions. The table totals only 700 nets. On 111 nets, computing $MinTime_{\mathcal{N}}(1)$ times out, while on 32 nets computing $MaxTime_{\mathcal{N}}(1)$ times out, and both time out on 51 nets. On the remaining 360 nets, there is no execution from $\{\mathsf{i}\colon 1\}$ to $\{\mathsf{f}\colon 1\}$, thus $MinTime_{\mathcal{N}}(1) = \infty$.

The analysis times for computing $a_{\mathcal{N}}$, $MinTime_{\mathcal{N}}(1)$ and $MaxTime_{\mathcal{N}}(1)$ are shown in the bottom table of Figure 6. We group nets by their size $|\mathcal{N}| = |P| + |T|$ to show how the analysis times depend on the instance size. We only

	Buckets B				
	$[0, 0.75)$	$[0.75, 1)$	$[1, 1]$	$(1, 1.75)$	$[1.75, \infty)$
Count with $\mathfrak{L} \in B$	303	274	422	173	82

	Buckets B				
	$[0, 0.05)$	$[0.05, 0.15)$	$[0.15, 0.3)$	$[0.3, 0.5)$	$[0.5, 1)$
Count with $\mathfrak{D} \in B$	29	222	295	120	34

		Buckets B			
		$[0, 20)$	$[20, 60)$	$[60, 150)$	$[150, 405)$
Count with $\lvert\mathcal{N}\rvert \in B$		241	391	388	40
Analysis time for computing $a_{\mathcal{N}}$ (in ms)	Mean	11.9	9.56	9.65	9.8
	Median	7	7	8	8
	Max	714	246	289	33
Analysis time for computing $MinTime_{\mathcal{N}}(1)$ (in ms)	Mean	8.29	120.52	1610.44	2128.83
	Median	8	36	307	1454
	Max	14	6599	14905	12160
Analysis time for computing $MaxTime_{\mathcal{N}}(1)$ (in ms)	Mean	3.99	44.23	669.66	5305.5
	Median	4	29	173	4934
	Max	8	2561	12370	14954

Fig. 6. Top: Statistics on the distribution of $\mathfrak{L}$. Middle: Statistics on the distribution of $\mathfrak{D}$. Bottom: Statistics on the analysis times for $a_{\mathcal{N}}$, $\mathfrak{I}_{Min}$ and $\mathfrak{I}_{Max}$.

list 1060 nets, as we omit those where the computation of $MinTime_{\mathcal{N}}(1)$ or $MaxTime_{\mathcal{N}}(1)$ timed out. One interesting observation is that for most instances, particularly small ones, $MinTime_{\mathcal{N}}(1)$ is harder to compute than $MaxTime_{\mathcal{N}}(1)$. However, both are very slow to compute compared to $a_{\mathcal{N}}$, which indeed never times out on our instances. In fact, $a_{\mathcal{N}}$ takes at most $714ms$ to compute for any instance. It is interesting that the time for computing $a_{\mathcal{N}}$ does not seem to depend highly on the net size. We suspect this might be partly due to the fact that $a_{\mathcal{N}}$ tends to be proportionally smaller for larger instances: Bucket $[0, 20)$ has a mean $\mathfrak{L}$ of 1.04, while the mean is 0.86 for bucket $[150, 405)$.

7.4 1-Soundness

Lastly, we briefly comment on the time for deciding 1-soundness via unrolling for nets with known $a_{\mathcal{N}}$. The procedure times out for 71 instances, among which $a_{\mathcal{N}}$ has a mean of 133.88 and a maximum of 256. It takes a mean of $612.66ms$ and a maximum of $14431ms$ to decide 1-soundness in this way. Unlike in the case for generalised soundness, our procedure for 1-soundness does not seem to be able to compete with the state-of-the-art. In [19], 1-soundness is decided for many of our instances in a few milliseconds per instance, which our approach does so only for instances with small $a_{\mathcal{N}}$ (up to about 25).

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: Proc. 18th International Conference on Application and Theory of Petri Nets (ICATPN). vol. 1248, pp. 407–426 (1997). https://doi.org/10.1007/3-540-63139-9_48
2. 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 Comput. **23**(3), 333–363 (2011). https://doi.org/10.1007/s00165-010-0161-4, https://doi.org/10.1007/s00165-010-0161-4
3. van der Aalst, W.M.: A class of petri net for modeling and analyzing business processes. Computing Science Reports **95**(26), 1–25 (1995)
4. van der Aalst, W.M., Hirnschall, A., Verbeek, H.: An alternative way to analyze workflow graphs. In: Advanced Information Systems Engineering: 14th International Conference, CAiSE 2002 Toronto, Canada, May 27–31, 2002 Proceedings 14. pp. 535–552. Springer (2002)
5. Bérard, B., Cassez, F., Haddad, S., Lime, D., Roux, O.H.: Comparison of different semantics for time petri nets. In: Peled, D.A., Tsay, Y. (eds.) Automated Technology for Verification and Analysis, Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4-7, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3707, pp. 293–307. Springer (2005). https://doi.org/10.1007/11562948_23, https://doi.org/10.1007/11562948_23
6. Blondin, M., Finkel, A., Haase, C., Haddad, S.: Approaching the coverability problem continuously. In: Proc. 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 480–496 (2016). https://doi.org/10.1007/978-3-662-49674-9_28
7. Blondin, M., Finkel, A., Haase, C., Haddad, S.: The logical view on continuous Petri nets. ACM Transactions on Computational Logic (TOCL) **18**(3), 24:1–24:28 (2017). https://doi.org/10.1145/3105908
8. Blondin, M., Haase, C.: Logics for continuous reachability in petri nets and vector addition systems with states. In: 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017. pp. 1–12. IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005068, https://doi.org/10.1109/LICS.2017.8005068
9. Blondin, M., Haase, C., Offtermatt, P.: Directed reachability for infinite-state systems. In: Groote, J.F., Larsen, K.G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12652, pp. 3–23. Springer (2021). https://doi.org/10.1007/978-3-030-72013-1_1, https://doi.org/10.1007/978-3-030-72013-1_1
10. Blondin, M., Mazowiecki, F., Offtermatt, P.: The complexity of soundness in workflow nets. In: Proc. 37th Symposium on Logic in Computer Science (LICS) (2022). https://doi.org/10.1145/3531130.3533341, https://doi.org/10.1145/3531130.3533341
11. Blondin, M., Mazowiecki, F., Offtermatt, P.: Verifying generalised and structural soundness of workflow nets via relaxations. In: Shoham, S., Vizel, Y. (eds.) CAV. Lecture Notes in Computer Science, vol. 13372, pp. 468–489. Springer (2022). https://doi.org/10.1007/978-3-031-13188-2_23, https://doi.org/10.1007/978-3-031-13188-2_23

12. Brázdil, T., Chatterjee, K., Kucera, A., Novotný, P., Velan, D.: Deciding fast termination for probabilistic VASS with nondeterminism. In: Chen, Y., Cheng, C., Esparza, J. (eds.) Automated Technology for Verification and Analysis - 17th International Symposium, ATVA 2019, Taipei, Taiwan, October 28-31, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11781, pp. 462–478. Springer (2019). https://doi.org/10.1007/978-3-030-31784-3_27, https://doi.org/10.1007/978-3-030-31784-3_27
13. Brázdil, T., Chatterjee, K., Kucera, A., Novotný, P., Velan, D., Zuleger, F.: Efficient algorithms for asymptotic bounds on termination time in VASS. In: Dawar, A., Grädel, E. (eds.) Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018. pp. 185–194. ACM (2018). https://doi.org/10.1145/3209108.3209191, https://doi.org/10.1145/3209108.3209191
14. Bride, H., Kouchnarenko, O., Peureux, F.: Reduction of workflow nets for generalised soundness verification. In: Proc. 18th International Conference Verification, Model Checking, and Abstract Interpretation (VMCAI). pp. 91–111 (2017). https://doi.org/10.1007/978-3-319-52234-0_6
15. Czerwinski, W., Orlikowski, L.: Reachability in vector addition systems is Ackermann-complete. In: Proc. 62^{nd} Annual IEEE Symposium on Foundations of Computer Science (FOCS) (2021), to appear
16. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press (1995). https://doi.org/10.1017/CBO9780511526558
17. Dixon, A., Lazic, R.: Kreach: A tool for reachability in petri nets. In: Biere, A., Parker, D. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12078, pp. 405–412. Springer (2020). https://doi.org/10.1007/978-3-030-45190-5_22, https://doi.org/10.1007/978-3-030-45190-5_22
18. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The prom framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) Applications and Theory of Petri Nets 2005. pp. 444–454. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
19. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous soundness checking of industrial business process models. In: Proc. 7th International Conference on Business Process Management (BPM). pp. 278–293 (2009). https://doi.org/10.1007/978-3-642-03848-8_19
20. Fraca, E., Haddad, S.: Complexity analysis of continuous Petri nets. Fundamenta Informaticae **137**(1), 1–28 (2015). https://doi.org/10.3233/FI-2015-1168
21. Freytag, T., Allgaier, P., Burattin, A., Danek-Bulius, A.: Woped-a" proof-of-concept" platform for experimental bpm research projects. In: 15th International Conference on Business Process Management (BPM 2017) (2017)
22. Geffroy, T., Leroux, J., Sutre, G.: Occam's razor applied to the petri net coverability problem. Theor. Comput. Sci. **750**, 38–52 (2018). https://doi.org/10.1016/j.tcs.2018.04.014, https://doi.org/10.1016/j.tcs.2018.04.014
23. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM **39**(3), 675–735 (1992). https://doi.org/10.1145/146637.146681, https://doi.org/10.1145/146637.146681
24. Grigoriev, D.: Complexity of deciding tarski algebra. J. Symb. Comput. **5**(1/2), 65–108 (1988). https://doi.org/10.1016/S0747-7171(88)80006-3, https://doi.org/10.1016/S0747-7171(88)80006-3

25. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023), https://www.gurobi.com
26. van Hee, K.M., Oanea, O., Sidorova, N., Voorhoeve, M.: Verifying generalized soundness of workflow nets. In: Proc. 6th International Andrei Ershov Memorial Conference on Perspectives of Systems Informatics (PSI). pp. 235–247 (2006). https://doi.org/10.1007/978-3-540-70881-0_21
27. van Hee, K.M., Sidorova, N., Voorhoeve, M.: Soundness and separability of workflow nets in the stepwise refinement approach. In: Proc. 24th International Conference on Applications and Theory of Petri Nets 2003 (ICATPN). vol. 2679, pp. 337–356 (2003). https://doi.org/10.1007/3-540-44919-1_22
28. van Hee, K.M., Sidorova, N., Voorhoeve, M.: Generalised soundness of workflow nets is decidable. In: Proc. 25th International Conference on Applications and Theory of Petri Nets (ICATPN). pp. 197–215 (2004). https://doi.org/10.1007/978-3-540-27793-4_12
29. Hopcroft, J., Pansiot, J.J.: On the reachability problem for 5-dimensional vector addition systems. Theoretical Computer Science **8**(2), 135–159 (1979)
30. Kiepuszewski, B., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Fundamentals of control flow in workflows. Acta Informatica **39**(3), 143–209 (2003). https://doi.org/10.1007/s00236-002-0105-4
31. Kosaraju, S.R., Sullivan, G.F.: Detecting cycles in dynamic graphs in polynomial time (preliminary version). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 398–406. ACM (1988). https://doi.org/10.1145/62212.62251, https://doi.org/10.1145/62212.62251
32. Kucera, A., Leroux, J., Velan, D.: Efficient analysis of VASS termination complexity. In: Hermanns, H., Zhang, L., Kobayashi, N., Miller, D. (eds.) LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020. pp. 676–688. ACM (2020). https://doi.org/10.1145/3373718.3394751, https://doi.org/10.1145/3373718.3394751
33. Leroux, J.: Polynomial vector addition systems with states. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic. LIPIcs, vol. 107, pp. 134:1–134:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.134, https://doi.org/10.4230/LIPIcs.ICALP.2018.134
34. Leroux, J.: The reachability problem for Petri nets is not primitive recursive. In: Proc. 62^{nd} Annual IEEE Symposium on Foundations of Computer Science (FOCS) (2021), to appear
35. Leroux, J., Schmitz, S.: Reachability in vector addition systems is primitive-recursive in fixed dimension. In: Proc. 34th Symposium on Logic in Computer Science (LICS) (2019). https://doi.org/10.1109/LICS.2019.8785796
36. Leroux, J., Schnoebelen, P.: On functions weakly computable by petri nets and vector addition systems. In: Ouaknine, J., Potapov, I., Worrell, J. (eds.) Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8762, pp. 190–202. Springer (2014). https://doi.org/10.1007/978-3-319-11439-2_15, https://doi.org/10.1007/978-3-319-11439-2_15
37. Lipton, R.: The Reachability Problem Requires Exponential Space. Department of Computer Science. Yale University **62** (1976)

38. Mayr, E.W., Weihmann, J.: A framework for classical petri net problems: Conservative petri nets as an application. In: Ciardo, G., Kindler, E. (eds.) Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8489, pp. 314–333. Springer (2014). https://doi.org/10.1007/978-3-319-07734-5_17, https://doi.org/10.1007/978-3-319-07734-5_17
39. Meyer, P.J., Esparza, J., Offtermatt, P.: Computing the expected execution time of probabilistic workflow nets. In: Vojnar, T., Zhang, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 154–171. Springer International Publishing, Cham (2019)
40. Meyer, P.J., Esparza, J., Völzer, H.: Computing the concurrency threshold of sound free-choice workflow nets. In: Beyer, D., Huisman, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 3–19. Springer International Publishing, Cham (2018)
41. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989). https://doi.org/10.1109/5.24143
42. Praveen, M., Lodaya, K.: Analyzing reachability for some petri nets with fast growing markings. Electron. Notes Theor. Comput. Sci. **223**, 215–237 (2008). https://doi.org/10.1016/j.entcs.2008.12.041, https://doi.org/10.1016/j.entcs.2008.12.041
43. Rackoff, C.: The covering and boundedness problems for vector addition systems. Theoretical Computer Science **6**, 223–231 (1978). https://doi.org/10.1016/0304-3975(78)90036-1
44. Schmitz, S.: The complexity of reachability in vector addition systems. ACM SIGLOG News **3**(1), 4–21 (2016). https://doi.org/10.1145/2893582.2893585, https://doi.org/10.1145/2893582.2893585
45. Valk, R., Vidal-Naquet, G.: Petri nets and regular languages. J. Comput. Syst. Sci. **23**(3), 299–325 (1981). https://doi.org/10.1016/0022-0000(81)90067-2, https://doi.org/10.1016/0022-0000(81)90067-2
46. Verbeek, E., van der Aalst, W.M.P.: Woflan 2.0: A Petri-net-based workflow diagnosis tool. In: Proc. 21^{st} Conference on Application and Theory of Petri Nets 2000, , (ICATPN). pp. 475–484 (2000). https://doi.org/10.1007/3-540-44988-4_28
47. Wolf, K.: Petri net model checking with LoLA 2. In: Application and Theory of Petri Nets and Concurrency. pp. 351–362 (2018)

A Missing proofs of Section 3

This section is devoted to prove Theorem 1 and Lemma 2. We start from some preliminary lemmas useful in the proof. Then we restate and prove Lemma 2. Finally, we restate and prove Theorem 1.

Lemma 12 (Reformulation of Lemma 5.5 in [10][9]). *Let* $\mathcal{N} = (P, T, F)$ *be a nonredundant workflow net. Let* $k \in \mathbb{N}$*,* $\boldsymbol{m} \in \mathbb{N}^P$*, and* π *be a run such that* $\{\mathsf{i}\colon k\} \rightarrow^{\pi}_{\mathbb{Z}} \boldsymbol{m}$*. There exist* $\ell \in \mathbb{N}$*,* $\boldsymbol{m'} \in \mathbb{N}^P$ *and a run* π' *with* $\boldsymbol{R}_\pi = \boldsymbol{R}_{\pi'}$ *such that* $\{\mathsf{i}\colon \ell\} \rightarrow^{\delta} \boldsymbol{m'}$ *and* $\{\mathsf{i}\colon \ell + k\} \rightarrow^{\delta\pi'} \boldsymbol{m} + \boldsymbol{m'}$*.*

Intuitively, the lemma states that any run under $\rightarrow^*_{\mathbb{Z}}$ can be turned into a run under $\rightarrow^*$ when starting at a large enough initial marking. The lemma follows from nonredundancy. Roughly, each place can be filled with arbitrarily many tokens, so with enough tokens to cause run π to remain positive in any place.

We say that a workflow net $\mathcal{N} = (P, T, F)$ is $\mathbb{Z}$-unbounded if there exists $\boldsymbol{R} \in \mathbb{N}^T$ such that $\Delta(\boldsymbol{R}) > \boldsymbol{0}$. The following lemma shows, intuitively, that $\mathbb{Z}$-boundedness guarantees a bound on the norm of reachable configurations.

Lemma 13 (Lemma 5.10 in [10]). *Let* $\mathcal{N} = (P, T, F)$ *be a workflow net. Suppose* $\{\mathsf{i}\colon k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}$*, where* $\boldsymbol{m} \in \mathbb{N}^P$*. There exists a constant* c *depending only on* k *and* $\mathcal{N}$ *such that if* $\|\boldsymbol{m}\| > c$ *then* $\mathcal{N}$ *is* $\mathbb{Z}$*-unbounded.*[10]

The following lemma can be shown using Lemmas 12 and 13.

Lemma 2. *Let* $\mathcal{N} = (P, T, F)$ *be a nonredundant workflow net. Then* $\mathcal{N}$ *is non-terminating iff there exists a nonzero* $\boldsymbol{R} \in \mathbb{N}^T$ *such that* $\Delta(\boldsymbol{R}) \geq \boldsymbol{0}$*.*

Proof. $\Longleftarrow$ Let $\boldsymbol{R}$ be a nonzero vector such that $\Delta(\boldsymbol{R}) \geq \boldsymbol{0}$ and denote $\boldsymbol{m} = \Delta(\boldsymbol{R})$. By Lemma 12 there exists ℓ such that $\{\mathsf{i}\colon \ell\} \rightarrow^* \boldsymbol{m'} \rightarrow^{\pi} \boldsymbol{m} + \boldsymbol{m'}$ for some $\boldsymbol{m'} \in \mathbb{N}^P$, and where the Parikh image of π is $\boldsymbol{R}$. Since $\boldsymbol{m} \geq \boldsymbol{0}$, it remains to observe that for every $e \in \mathbb{N}$ we have $e \cdot \boldsymbol{m} + \boldsymbol{m'} \rightarrow^{\pi} (e+1) \cdot \boldsymbol{m} + \boldsymbol{m'}$. Since $\boldsymbol{R}_\pi = \boldsymbol{R}$ and $\boldsymbol{R}$ is non-zero, it must hold that $|\pi| > 0$. Thus, $\mathcal{N}$ is non-terminating.

$\Longrightarrow$ Suppose $\mathcal{N}$ is non-terminating. Fix k such that $MaxTime_{\mathcal{N}}(k) = \infty$. Consider a run $\pi_n = t_1 \dots t_n$ and for every $1 \leq i \leq n$ let $\pi_i = t_1 \dots t_i$ be the first i transitions of π_n. We denote as $\boldsymbol{m}_i \in \mathbb{N}^P$ the marking such that $\{\mathsf{i}\colon k\} \rightarrow^{\pi_i} \boldsymbol{m}_i$. Let c be the constant from Lemma 13. If $\|\boldsymbol{m}_i\| > c$ for some $1 \leq i \leq n$ then by Lemma 13 $\mathcal{N}$ is $\mathbb{Z}$-unbounded and we are done. Otherwise $\|\boldsymbol{m}_i\| \leq c$ for all $1 \leq i \leq n$. We can assume that n is arbitrarily big as $MaxTime_{\mathcal{N}}(k) = \infty$. By the pigeonhole principle, there exist $i < j < n$ such that $\boldsymbol{m}_i = \boldsymbol{m}_j$. We are done since $\Delta(\boldsymbol{R}_{t_{i+1} \dots t_j}) = \boldsymbol{0}$. □

---

[9] Similar statements were already known in [28], but in the context of this work it will be more convenient to use the formulation from [10]. The formulation of Lemma 5.5 in [10] does not deal with π and π', but simply states that $\{\mathsf{i}\colon \ell + k\} \rightarrow^* \boldsymbol{m} + \boldsymbol{m'}$. The proof of the statement reorders the transitions in π, which implies the stronger statement that we need in Lemma 12. We require this to prove Lemma 2.

[10] In [10], an exponential bound on the size of c is given, but we will not need it.

Note that the condition $\Delta(\boldsymbol{R}) \geq \mathbf{0}$ is very similar to $\mathcal{N}$ being $\mathbb{Z}$-unbounded, except that the inequality is non-strict. Before we go further, we need one more relaxed semantics for Petri nets.

Definition 3 (Rational semantics). *Rational semantics for Petri nets is a combination of continuous and integer semantics. So, like in continuous semantics, whenever we fire a transition it is multiplied by a scalar $\beta \in \mathbb{Q}_{\geq 0}$ i.e. βt. From the integer semantics we take that we can fire βt even if it is not enabled. So markings are in $\mathbb{Q}^P$ where P is the set of places. We write $\boldsymbol{m} \rightarrow_{\mathbb{Q}}^{\beta t} \boldsymbol{m}'$ to denote that $\boldsymbol{m}' = \boldsymbol{m} + \beta \cdot \Delta(t)$. Like before $\rightarrow_{\mathbb{Q}}^{*}$ is a transitive closure of the single step relation.*

Let us relate $\rightarrow_{\mathbb{Z}}^{*}$ and $\rightarrow_{\mathbb{Q}}^{*}$ to solutions of LPs/ILPs. This is inspired by the definition of $\text{ILP}_{\mathcal{N}}$ in [10], but differs slightly as we leave the effect on the initial place implicit. Let $\mathcal{N} = (P, T, F)$ be a nonredundant workflow net. We define $\text{ILP}_{\mathcal{N}}$ with a $|T|$ dimensional vector of variables $\boldsymbol{x}$ and inequalities: $\boldsymbol{x} \geq \mathbf{0}$ and $\Delta(T)\boldsymbol{x} \geq \mathbf{0} - \{\mathsf{i} \colon \infty\}$.[11]
We recall.

Lemma 3. *[Adapted from Claim 5.7 in [10]] For every $k \in \mathbb{N}$, $\boldsymbol{m} \in \mathbb{N}^P$, and a run π, it holds that $\{\mathsf{i} \colon k\} \rightarrow_{\mathbb{Z}}^{\pi} \boldsymbol{m}$ iff $\boldsymbol{R}_\pi$ is a solution to $\text{ILP}_{\mathcal{N}}$ with the additional constraint $\sum_{i=1}^{|T|} \Delta(t_i)(\mathsf{i}) \cdot \boldsymbol{R}_\pi(t_i) \geq -k$.*

We also need another lemma.

Lemma 14 (Lemma 4.3 in [10]). *Let $G \coloneqq \mathbf{A} \cdot \boldsymbol{x} \geq \boldsymbol{b}$ be an $(m \times n)$-ILP, where $\boldsymbol{b} \geq \mathbf{0}$. There exists $c \leq \|G\|^{\mathcal{O}((m+n)\log(m+n))}$ such that for every $\boldsymbol{\mu} \in \mathbb{N}^T$ satisfying $\mathbf{A} \cdot \boldsymbol{\mu} \geq \boldsymbol{b}$, there exists $\boldsymbol{\nu} \in \mathbb{N}^T$ satisfying $\mathbf{A} \cdot \boldsymbol{\nu} \geq \boldsymbol{b}$ such that $\boldsymbol{\nu} \leq \boldsymbol{\mu}$, $\boldsymbol{\nu} \leq \boldsymbol{c}$, and $\mathbf{A} \cdot \boldsymbol{\nu} \leq \mathbf{A} \cdot \boldsymbol{\mu}$.*

Finally, we can restate Theorem 1 and prove it.

Theorem 1. *Every workflow net $\mathcal{N}$ is either non-terminating or linear. Moreover, $MaxTime_{\mathcal{N}}(k) \leq ak$ for some $a \leq \|\mathcal{N}\|^{poly(|\mathcal{N}|)}$.*

Proof. As explained in Section 2.3 we can assume that $\mathcal{N}$ is nonredundant, *i.e.* for all $p \in P$ there exists $k \in \mathbb{N}$ such that $\{\mathsf{i} \colon k\} \rightarrow^{*} \boldsymbol{m}$ with $\boldsymbol{m}(p) > 0$.

Let $\mathcal{N} = (P, T, F)$ and denote $T = \{t_1, \ldots, t_n\}$. By Lemma 2 we need to prove that if there is no $\boldsymbol{R} \in \mathbb{N}^T$ such that $\Delta(\boldsymbol{R}) \geq \mathbf{0}$, then $\mathcal{N}$ is linear. Indeed, let as assume that there is no such $\boldsymbol{R}$.

We denote $\boldsymbol{x} = (x_1, \ldots, x_{|T|})$. Let $\boldsymbol{A}$ be the matrix such that $\boldsymbol{A} \cdot \boldsymbol{x} \geq \mathbf{0}$ encodes $\text{ILP}_{\mathcal{N}}$. Precisely, the matrix $\boldsymbol{A}$ is a matrix $\Delta(T)$ from which the row for the place i is removed. Let $k \in \mathbb{N}$ and consider a run $\{\mathsf{i} \colon k\} \rightarrow_{\mathbb{Z}}^{\pi} \boldsymbol{m}$. Note that $\boldsymbol{R}_\pi$ is a solution to $\text{ILP}_{\mathcal{N}}$ by Lemma 3. Let c be the constant from Lemma 14 for $\text{ILP}_{\mathcal{N}}$. We will show that $|\pi| \leq c|T| \cdot k$, which will conclude the proof as $c \leq \|\mathcal{N}\|^{poly(|\mathcal{N}|)}$, thus $c|T|$ depends only on $\mathcal{N}$.

[11] The inequality indexed by the place i can be dropped, as trivially satisfied by any valuation of $\boldsymbol{x}$.

By Lemma 14 there is a solution $\boldsymbol{R}_1$ of ILP$_{\mathcal{N}}$ with $\boldsymbol{R}_1 \leq \boldsymbol{R}_\pi$ and $\boldsymbol{R}_1 \leq \boldsymbol{c}$. Since $\boldsymbol{R}_1 \leq \boldsymbol{R}_\pi$, we have $\boldsymbol{R}_2 \coloneqq \boldsymbol{R}_\pi - \boldsymbol{R}_1 \in \mathbb{N}^T$. By Lemma 14 we have $\boldsymbol{A} \cdot \boldsymbol{R}_1 \leq \boldsymbol{A} \cdot \boldsymbol{R}_\pi$. Thus $\boldsymbol{A} \cdot \boldsymbol{R}_2 = \boldsymbol{A} \cdot (\boldsymbol{R}_\pi - \boldsymbol{R}_1) \geq \boldsymbol{0}$. This proves that $\boldsymbol{R}_2$ is a solution to ILP$_{\mathcal{N}}$.

Note that $\Delta(\boldsymbol{R}_1) \geq \boldsymbol{0}$ or $\Delta(\boldsymbol{R}_2) \geq \boldsymbol{0}$ would contradict our initial assumption. Since both $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are solutions to ILP$_{\mathcal{N}}$ they can have negative effect only in i. By recursively applying Lemma 14 to $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ we get a $\boldsymbol{R}_\pi = \boldsymbol{R}_1 + \boldsymbol{R}_2 \ldots \boldsymbol{R}_l$, where $\boldsymbol{0} \leq \boldsymbol{R}_i \leq \boldsymbol{c}$. Moreover, $l \leq k$, since every $\boldsymbol{R}_i$ has a negative effect on i and the initial marking is $\{\mathsf{i}\colon k\}$. This shows that $|\pi| \leq c \cdot |T| \cdot l \leq c \cdot |T| \cdot k$. ☐

B Missing proofs of Section 4

Lemma 5. *Let $M \subseteq \mathbb{Q}_{\geq 0}^P$ be a set of solutions of some LP. Then testing if a net is good for M can be done in polynomial time.*

Proof. Continuous reachability relation $\boldsymbol{m} \rightarrow^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}$ can be expressed as a formula $\Phi(\boldsymbol{m}, \boldsymbol{m'})$ in a logic defined in [8], more precisely see [8, Theorem 3.6]. Roughly speaking, the logic is existentially quantified linear programming extended with implications of the following form: $x_i > 0 \implies x_j > 0$. For example in this logic one can write the formula

$$\begin{aligned} \Lambda(\boldsymbol{y}) = \exists_{\boldsymbol{x} \in \mathbb{Q}^d}\ \boldsymbol{A} \cdot \boldsymbol{x} + \boldsymbol{A'} \cdot \boldsymbol{y} \geq \boldsymbol{b}\ \wedge\ \boldsymbol{x}(5) > 0 \implies \boldsymbol{y}(7) > 0\ \wedge \\ \boldsymbol{x}(5) > 0 \implies \boldsymbol{x}(8) > 0\ \wedge\ \boldsymbol{x}(2) > 0 \implies \boldsymbol{y}(3) > 0 \end{aligned}$$

The satisfiability of formulas in this logic is in polynomial time.

To test if the net is good for M we write the formula:

$$\begin{aligned} \bigwedge_{p \in P} \exists_{\boldsymbol{m_p} \in \mathbb{Q}^d} \exists_{\boldsymbol{m'_p} \in \mathbb{Q}^d}\ \boldsymbol{m_p} \geq 0\ \wedge \boldsymbol{m_p}(p) > 0 \wedge\ \boldsymbol{m'_p} \in M \\ \wedge\ \Phi(\{\mathsf{i}\colon 1\}, \boldsymbol{m_p})\ \wedge\ \Phi(\boldsymbol{m_p}, \boldsymbol{m'_p}) \end{aligned}$$

☐

Lemma 6. *Suppose a workflow net $\mathcal{N}$ is good for $M \subseteq \mathbb{Q}_{\geq 0}^P$ and M is a convex set. Then there is a marking $\boldsymbol{m_+}$ such that $\boldsymbol{m_+}(p) > 0$ for every $p \in P$ and there are continuous runs π, π', and a marking $\boldsymbol{m_f} \in M$ such that $\{\mathsf{i}\colon 1\} \rightarrow^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m_+} \rightarrow^{\pi'}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m_f}$.*

Proof. As $\mathcal{N}$ is good for M we know that for every $p \in P$ there are runs π_p, π'_p and a marking $\boldsymbol{m_p}$ such that $\boldsymbol{m_p}(p) > 0$ and $\{\mathsf{i}\colon 1\} \rightarrow^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m} \rightarrow^{\pi'}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'_p}$ for some $\boldsymbol{m'_p} \in M$. Because of Lemma 1 we know that $\{\mathsf{i}\colon \frac{1}{|P|}\} \rightarrow^{\frac{1}{|P|}\pi_p}_{\mathbb{Q}_{\geq 0}} \frac{1}{|P|}\boldsymbol{m_p} \rightarrow^{\frac{1}{|P|}\pi'_p}_{\mathbb{Q}_{\geq 0}} \frac{1}{|P|}\boldsymbol{m'_p}$. Let $P = \{p_1 \ldots p_l\}$. We define

- $\pi = \frac{1}{|P|}\pi_{p_1} \frac{1}{|P|}\pi_{p_2} \cdots \frac{1}{|P|}\pi_{p_l}$,
- $\pi' = \frac{1}{|P|}\pi'_{p_1} \frac{1}{|P|}\pi'_{p_2} \cdots \frac{1}{|P|}\pi'_{p_l}$,

- $\boldsymbol{m_+} = \frac{1}{|P|}\boldsymbol{m_{p_1}} + \frac{1}{|P|}\boldsymbol{m_{p_2}} \dots \frac{1}{|P|}\boldsymbol{m_{p_l}}$,
- $\boldsymbol{m_f} = \frac{1}{|P|}\boldsymbol{m'_{p_1}} + \frac{1}{|P|}\boldsymbol{m'_{p_2}} \dots \frac{1}{|P|}\boldsymbol{m'_{p_l}}$

We have to check that $\boldsymbol{m_+}, \boldsymbol{m_f}, \pi, \pi'$ have all of the required properties. First, we observe that $\{\mathsf{i}\colon 1\} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m_+} \to^{\pi'}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m_f}$. Further, as $\boldsymbol{m_p}(p) > 0$ and $\boldsymbol{m_p} \geq \boldsymbol{0}$ for every $p \in P$ then $\boldsymbol{m_+}(p) > 0$ for every $p \in P$. Finally, as M is convex and $\boldsymbol{m'_p} \in M$ for every $p \in P$ we have that $\boldsymbol{m_f} = \frac{1}{|P|}\boldsymbol{m'_{p_1}} + \frac{1}{|P|}\boldsymbol{m'_{p_2}} + \ldots + \frac{1}{|P|}\boldsymbol{m'_{p_l}} \in M$. □

Lemma 8. *Suppose M is a convex set of markings over $\mathbb{Q}^P_{\geq 0}$ and that the workflow net is good for M. Let S be the set of Parikh images of continuous runs that start in $\{\mathsf{i}\colon 1\}$ and end in some marking $\boldsymbol{m'} \in M$ i.e.*

$$S \coloneqq \{\boldsymbol{R}_\pi \mid \exists_{\pi \in CRuns^1_{\mathcal{N}}} \exists_{\boldsymbol{m'} \in M} \text{ such that } \{\mathsf{i}\colon 1\} \to^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}\}.$$

Then $\boldsymbol{v} \in \overline{S}$ if and only if there is a marking $\boldsymbol{m} \in M$ such that $\Delta(T)\boldsymbol{v} = \boldsymbol{m} - \{\mathsf{i}\colon 1\}$.

Proof. ($\Longrightarrow$) If $\boldsymbol{v} \in S$ then this is trivial. Otherwise, we exploit the fact that the set of solutions of the program on the right is a closed set.

($\Longleftarrow$) To prove the opposite implication, it suffices to define a sequence of continuous runs $\{\mathsf{i}\colon 1\} \to^{\pi_i}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m_i}$ such that each $\boldsymbol{m_i} \in M$ and their Parikh images converge to $\boldsymbol{v}$. Let $\boldsymbol{m_+}$ be a marking such that $\boldsymbol{m_+}(p) > 0$ for all $p \in P$ and $\{\mathsf{i}\colon 1\} \to^{\rho}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m_+} \to^{\rho'}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m_f}$ for some $\boldsymbol{m_f} \in M$; we know it exists because of Lemma 6.

For any rational number $0 < \varepsilon \leq 1$ there is a run

$$\{\mathsf{i}\colon 1\} \to^{\varepsilon\rho}_{\mathbb{Q}_{\geq 0}} \{\mathsf{i}\colon 1 - \varepsilon\} + \varepsilon\boldsymbol{m_+} \to^{\pi_\varepsilon}_{\mathbb{Q}_{\geq 0}} (1-\varepsilon)\boldsymbol{m} + \varepsilon\boldsymbol{m_+} \to^{\varepsilon\rho'}_{\mathbb{Q}_{\geq 0}} (1-\varepsilon)\boldsymbol{m} + \varepsilon\boldsymbol{m_f},$$

such that $\boldsymbol{R}_{\pi_\varepsilon} = (1-\varepsilon)\boldsymbol{v}$. The first part is correct because of Lemma 1 applied to the run ρ. The second part is correct because of Lemma 7. Indeed, markings $\{\mathsf{i}\colon 1 - \varepsilon\} + \varepsilon\boldsymbol{m_+}$ and $(1-\varepsilon)\boldsymbol{m} + \varepsilon\boldsymbol{m_+}$ are positive on all places and $\Delta(T)\left((1-\varepsilon)\boldsymbol{v}\right) = (1-\varepsilon)(\boldsymbol{m} - \{\mathsf{i}\colon 1\})$. The third part is correct one more time because of Lemma 1 applied to the run ρ'.
Moreover the final marking $(1-\varepsilon)\boldsymbol{m} + \varepsilon\boldsymbol{m_f} \in M$ as the set M is convex.

So we define our sequence as $(\varepsilon\rho)\pi_\varepsilon(\varepsilon\rho')$ where $\varepsilon = \frac{1}{n}$ for $n \in \mathbb{N}$ and $n \to \infty$. □

C Missing proofs of Section 5

Missing intuition for the proof of coNP-hardness in Theorem 3. The reduction in [11, Theorem 2] is the problem of checking whether a given formula in DNF is a tautology, which is coNP-complete. The constructed acyclic workflow net $\mathcal{N}$ depends on the input formula φ. It is not important to know the formal definition of acyclic workflow nets, we only use the property that acyclic nets are terminating (which is the case). Thus, the workflow net constructed in [11, Theorem 2] is terminating and by Theorem 1 it is also linear. Without going into the details of $\mathcal{N}$ we highlight the important bits in the proof of [11, Theorem 2] showing that ϕ is generalised sound iff φ is a tautology.

$\Longrightarrow$ Suppose $\mathcal{N}$ is generalised sound. By [11, Theorem 1] it implies being continuous sound. The proof in [11, Theorem 2] shows that then φ is a tautology.

$\Longleftarrow$ Suppose φ is a tautology. Let $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m}$ for some $k \in \mathbb{N}$. Note that $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}$ trivially holds. By Lemma 1 we can also rescale the continuous run and obtain $\{\mathsf{i}\colon 1\} \to^*_{\mathbb{Q}_{\geq 0}} \boldsymbol{m} \cdot \frac{1}{k}$. The proof in [11, Theorem 2] implies that $\boldsymbol{m} \cdot \frac{1}{k} \to^{\rho}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ for some continuous run $\rho = \alpha_1 t_1 \dots \alpha_n t_n$. By Lemma 1 $\boldsymbol{m} \to^{k\rho}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon k\}$. It remains to observe that $k\rho$ is essentially a run (not just a continuous run). It suffices to prove that $k \cdot \alpha_i$ is a natural number for all $1 \leq i \leq n$. A close inspection of the proof in [11, Theorem 2] shows that α_i are defined as the number of tokens in $\boldsymbol{m} \cdot \frac{1}{k}$, which concludes the proof. □

Lemma 9. *Let $\mathcal{N}$ be a terminating nonredundant workflow net. Then $\mathcal{N}$ is not generalised sound iff there exist $k \in \mathbb{N}$ and a marking $\boldsymbol{m} \in \mathbb{N}^P$ such that $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$, $\boldsymbol{m}$ is a deadlock and $\boldsymbol{m} \neq \{\mathsf{f}\colon k\}$. Moreover, if $\|\mathcal{N}\| \leq 1$ then $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m}$ can be replaced with $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Q}} \boldsymbol{m}$.*

Proof. $\Longrightarrow$ Suppose $\mathcal{N}$ is generalised unsound. There exist k and $\boldsymbol{m'}$ such that $\{\mathsf{i}\colon k\} \to^* \boldsymbol{m'} \not\to^* \{\mathsf{f}\colon k\}$. Since $\mathcal{N}$ is terminating there exists $\boldsymbol{m'} \to^* \boldsymbol{m}$ such that $\boldsymbol{m}$ is a deadlock. Since $\boldsymbol{m'} \not\to^* \{\mathsf{f}\colon k\}$ we know that $\boldsymbol{m} \neq \{\mathsf{f}\colon k\}$. We have obtained k and $\boldsymbol{m}$ as required. In this case the additional statement of the lemma is trivial as we also have $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Q}} \boldsymbol{m}$.

$\Longleftarrow$ Let k and $\boldsymbol{m}$ be as in the lemma. By definition $\{\mathsf{i}\colon k\} \to^*_{\mathbb{Z}} \boldsymbol{m} \not\to^* \{\mathsf{f}\colon k\}$. Now, generalised unsoundness is a simple corollary of Lemma 12 (it follows directly from [10, Lemma 5.6]). We obtain that $\{\mathsf{i}\colon k + l\} \to^* \boldsymbol{m} + \boldsymbol{m'}$ for some $l \in \mathbb{N}$ and marking $\boldsymbol{m'}$ such that $\{\mathsf{i}\colon l\} \to^* \boldsymbol{m'}$. To conclude, it holds that either $\boldsymbol{m'} \not\to^* \{\mathsf{f}\colon l\}$ or $\boldsymbol{m} + \boldsymbol{m'} \to^* \boldsymbol{m} + \{\mathsf{f}\colon l\}$. In both cases, $\mathcal{N}$ is not generalised sound.

It remains to deal with the additional case, where we only assume that $\{\mathsf{i}\colon k\} \to^{\pi}_{\mathbb{Q}} \boldsymbol{m} \not\to^* \{\mathsf{f}\colon k\}$ for some rational run π. The assumption $\|\mathcal{N}\| \leq 1$ gives us that for every transition t there is a place p_t such that ${}^\bullet t = 1$ and $\boldsymbol{m}(p_t) = 0$. In other words the places that do not have enough tokens to enable a transition need to have 0 tokens. By Lemma 1[12] $\{\mathsf{i}\colon ak\} \to^{a\cdot\pi}_{\mathbb{Q}} a \cdot \boldsymbol{m}$ for every $a \in \mathbb{N}$. We can choose a such that $a \cdot \pi$ is an integer run (not just a rational run). It remains to observe that $a \cdot \boldsymbol{m}$ is a deadlock because $a \cdot \boldsymbol{m}(p_t) = 0$ for all transitions t. We have reduced the problem to the previous case as $\{\mathsf{i}\colon ak\} \to^{a\cdot\pi}_{\mathbb{Z}} a \cdot \boldsymbol{m}$ for the chosen a. □

D Missing proofs of Section 6

We start from restating Lemma 10.

[12] Lemma 1 of course holds for rational runs as well.

Lemma 10. *Consider a run π and $k \in \mathbb{N}$. The greedy parallel execution of π has the smallest execution time among all parallel executions of π with respect to k.*

Proof. The idea of the proof is that any parallel execution can be transformed into the greedy execution. The crucial observation is that every transformation step cannot increase (but might decrease) the execution time. Let $\rho_1 \dots \rho_m$ be the greedy parallel execution.

The transformation step is defined as follows. Suppose $\pi_1\pi_2 \dots \pi_l$ is not a greedy parallel execution. Let i be the a first block such that $\pi_i \neq \rho_i$. Note that it must be the case that π_i is strictly contained in ρ_i. Indeed, $\rho_{i-1} = \pi_{i-1}$ and ρ_i is the maximal possible block after π_{i-1}. Thus the block π_{i+1} is nonempty, *i.e.* we can write it as $t\pi'_{i+1}$, where $t \in T$ and π'_{i+1} is a (possibly empty) block.

We define the new parallel execution by moving t to the previous block, *i.e.* $\pi_1, \dots, \pi_{i-1}(\pi_i t)\pi'_{i+1}\pi_{i+2}\dots\pi_l$. To see that this is a parallel execution we need to prove that the two conditions in Definition 2 hold for the two new blocks $(\pi_i t)$ and π'_{i+1} (for the remaining blocks nothing has changed). Indeed, $(\pi_i t)$ is contained in ρ_i, thus it satisfies both conditions. For π'_{i+1} we need to check the second condition. We have

$$^\bullet\boldsymbol{R}_{\pi'_{i+1}} = {}^\bullet\boldsymbol{R}_{\pi_{i+1}} - {}^\bullet t \leq \{\mathsf{i}\colon k\} + \sum_{j\leq i} \Delta(\pi_j) - {}^\bullet t \leq \sum_{j\leq i} \Delta(\pi_j) + \Delta(t).$$

It remains to observe that the execution time of the new parallel execution is at most l (it could decrease if π'_{i+1} is empty). It is easy to see that after every transformation step the parallel execution agrees on a longer prefix with the greedy execution. Thus it has to terminate with the greedy execution. □

Now we restate and prove Lemma 11.

Lemma 11. *Let $\mathcal{N}$ be a workflow net and let $k, x \in \mathbb{N}$. Deciding whether $MinTime_{\mathcal{N}}(k) \leq x$ is PSPACE-hard even if we fix $k = 1$.*

Proof. We reduce from the reachability problem of conservative Petri nets, which is known to be PSPACE-hard [38]. A conservative Petri net is a Petri net $\mathcal{N} = (P, T, F)$ such that $\sum_{p \in P} \Delta(t)(p) = 0$ for every $t \in T$. In simpler words, every transition preserves the number of tokens in the Petri net. Let $\boldsymbol{m}$ and $\boldsymbol{m}'$ be the markings for which we ask whether $\boldsymbol{m} \to^* \boldsymbol{m}'$.

We define the workflow net $\mathcal{N}' = (P', T', F')$, where $P' = P \cup \{\mathsf{i}, \mathsf{f}, r\}$, *i.e.* there are three extra places, including the initial and final places. Slightly abusing the notation we write $\boldsymbol{m}$ and $\boldsymbol{m}'$ as markings over P', by fixing the values to 0 on the new places $\mathsf{i}, \mathsf{f}, r$. The set of transitions is $T' = T \cup \{t_\mathsf{i}, t_\mathsf{f}\}$. The arc function F' is the same as F when restricted to $P \times T \cup T \times P$. Additionally:

1. ${}^\bullet t_\mathsf{i}(\mathsf{i}) = 1$, ${}^\bullet t_\mathsf{i}(p') = 0$ for all $p' \in P' \setminus \{\mathsf{i}\}$ and $t_\mathsf{i}^\bullet = \boldsymbol{m} + \{r : 1\}$;
2. ${}^\bullet t_\mathsf{f}(\mathsf{f}) = \boldsymbol{m}' + \{r : 1\}$ and $t_\mathsf{f}^\bullet(\mathsf{f}) = 1$, $t_\mathsf{f}^\bullet(p') = 0$ for all $p' \in P' \setminus \{\mathsf{f}\}$;
3. ${}^\bullet t(r) = t^\bullet(r) = 1$ and t consumes and produces 0 tokens on places i and f for all $t \in T$.

It remains to observe that this is a workflow net. Indeed, the only nontrivial condition is whether all places and transitions are on a path from i to f. It is easy to see that the place r is on such a path. Moreover, all other places and transitions are connected with r through the transitions in T. (In the special case that for a node $p \in P$ no token is ever consumed or produced it is easy to see that p can be removed from $\mathcal{N}$ without changing the reachability question).

Let $c = \sum_{p \in P} \boldsymbol{m}(p)$, *i.e.* the number of tokens in the initial configuration of $\mathcal{N}$. We define $x = |P'|^c$, which is an upper bound on the size of all possible configurations in $\mathcal{N}$ (recall that it is a conservative Petri net). We claim that $\boldsymbol{m} \to^* \boldsymbol{m'}$ in $\mathcal{N}$ if and only if $MinTime_{\mathcal{N}}(1) \leq x$ for $\mathcal{N}'$.

Indeed, it suffices to observe that $\mathcal{N}'$ essentially simulates $\mathcal{N}$. The transition t_{i} initialises the configuration to $\boldsymbol{m}$ (plus one token in the special place r) and the transition t_{f} checks whether $\boldsymbol{m'}$ was reached (note that because $\mathcal{N}$ is conservative coverability and reachability are the same problems). The special place r is for two reasons. First, because of r a block in a parallel execution can be only of length one. Indeed, as an invariant after firing t_{i} and until firing t_{f} the place r has always 1 token. Since all other transitions consume a token from r no two transitions can be fired in parallel. Thus whether $MinTime_{\mathcal{N}}(1) = x$ is equivalent to the question whether there is a run of length at most x. To conclude it remains to observe that if there is a run then there is a run of length bounded by x (as in the shortest run no configuration can repeat). □

Now we move to the analysis of $\lim_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k}$. We start by proving that it exists.

Lemma 15. *It holds that*

$$\liminf_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k} = \limsup_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k} = \lim_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k}.$$

Proof. First notice that

$$MinTime_{\mathcal{N}}(a+b) \leq MinTime_{\mathcal{N}}(a) + MinTime_{\mathcal{N}}(b) \tag{3}$$

(simply because we can sequentially use the parallel execution of $MinTime_{\mathcal{N}}(a)$ and then $MinTime_{\mathcal{N}}(b)$). To prove the lemma it suffices to prove that

$$\limsup_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k} \leq \liminf_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k}. \tag{4}$$

Let $n_1 < n_2 < n_3 < \dots$ be the indices such that

$$\liminf_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k} = \lim_{k\to\infty} \frac{MinTime_{\mathcal{N}}(n_k)}{n_k}$$

and let $m_1 < m_2 < m_3 < \ldots$ be the indices such that

$$\limsup_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k} = \lim_{k\to\infty} \frac{MinTime_{\mathcal{N}}(m_k)}{m_k}.$$

Note that we can replace (n_i) or (m_i) with any of their infinite subsequences. In particular without loss of generality we can assume that $MinTime_{\mathcal{N}}(n_i) \cdot n_i < m_i$ for all $i \in \mathbb{N}_{>0}$.

For every i let $a_i \in \mathbb{N}_{>0}$ and $0 \leq r_i \leq n_i - 1$ be the unique decomposition of m_i modulo n_i, *i.e.* $m_i = a_i \cdot n_i + r_i$. By Eq. (3) we get

$$\begin{aligned}
\frac{MinTime_{\mathcal{N}}(m_i)}{m_i} &= \frac{MinTime_{\mathcal{N}}(a_i \cdot n_i + r_i)}{m_i} \\
&\leq \frac{a_i \cdot MinTime_{\mathcal{N}}(n_i)}{m_i} + \frac{MinTime_{\mathcal{N}}(r_i)}{m_i} \\
&\leq \frac{a_i \cdot MinTime_{\mathcal{N}}(n_i)}{a_i \cdot n_i} + \frac{MinTime_{\mathcal{N}}(r_i)}{m_i} \\
&\leq \frac{MinTime_{\mathcal{N}}(n_i)}{n_i} + \frac{MinTime_{\mathcal{N}}(n_i)}{m_i} \\
&\leq \frac{MinTime_{\mathcal{N}}(n_i)}{n_i} + \frac{MinTime_{\mathcal{N}}(n_i)}{MinTime_{\mathcal{N}}(n_i) \cdot n_i} \\
&\leq \frac{MinTime_{\mathcal{N}}(n_i)}{n_i} + \frac{1}{n_i}
\end{aligned} \tag{5}$$

So

$$\begin{aligned}
\limsup_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k} &= \lim_{i\to\infty} \frac{MinTime_{\mathcal{N}}(m_i)}{m_i} \\
\leq \lim_{i\to\infty} \frac{MinTime_{\mathcal{N}}(n_i)}{n_i} + \frac{1}{n_i} &= \lim_{i\to\infty} \frac{MinTime_{\mathcal{N}}(n_i)}{n_i} \\
&= \liminf_{k\to\infty} \frac{MinTime_{\mathcal{N}}(k)}{k}
\end{aligned} \tag{6}$$

as required by Equation (4), and we are done. □

The proof of Theorem 4. The proof of the theorem requires preparation. The key idea for the algorithm is similar to the concept from Section 4, i.e. the relaxation to the continuous semantics. First we have to define a continuous version of the parallel execution, and define its execution time.

Definition 4. *For a continuous run $\pi_c = \beta_1 t_1, \beta_2 t_2 \ldots \beta_n t_n$ executable from the initial marking* i *its* continuous parallel execution *is a partition of the run into blocks $\pi_c = \pi_{c1}\pi_{c2}\ldots\pi_{cl}$ such that*

1. *a single transition cannot appear twice in a single block;*
2. *for every $i \leqslant l$ holds ${}^{\bullet}\boldsymbol{R}_{\pi_{ci}} \leqslant \{\mathsf{i}\colon 1\} + \sum_{j<i} \Delta(\pi_{cj})$.*

The execution time of a single block $\pi_{ci} = \beta_{i,1}t_{i,1}\beta_{i,2}t_{i,2}\ldots\beta_{i,n_i}t_{i,n_i}$ *equals* $exec(\pi_{ci}) = max(\beta_{i,1}, \beta_{i,2}\ldots\beta_{i,n_i})$*. The execution time of the continuous parallel run* $exec(\pi_{c1}, \pi_{c2}, \ldots \pi_{cl}) = \sum_{j=1}^{l} exec(\pi_{cj})$*.*

The ideas behind this definition are as follows:

- If $\beta_i = 1$ for all $i \leqslant n$ it is equivalent to parallel execution with the normal semantics.
- The execution time of each block, corresponding to a single step of the system, is equal to the maximal execution time among firings of individual transitions.

Our next goal is to relate $lim_{k\to\infty}\frac{MinTime_{\mathcal{N}}(k)}{k}$ and execution times of continuous parallel executions. Suppose π_c is a continuous run of the workflow. Let $exec_{opt}(\pi_c)$ be the minimum of execution times among its continuous parallel executions.

Lemma 16. $lim_{k\to\infty}\frac{MinTime_{\mathcal{N}}(k)}{k} = inf\{exec_{opt}(\pi_c) \mid \{\mathsf{i}\colon 1\} \to^{\pi_c}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}\}$

Proof. We prove the lemma by showing two inequalities.

($\leq$) We know that $\frac{MinTime_{\mathcal{N}}(e)}{e} \geq \frac{MinTime_{\mathcal{N}}(i\cdot e)}{i\cdot e}$ for any $i, e \in \mathbb{N}$ (see Eq. (3)). Thus for any run π such that $\{\mathsf{i}\colon e\} \to^{\pi} \{\mathsf{f}\colon e\}$ we have $lim_{k\to\infty}\frac{MinTime_{\mathcal{N}}(k)}{k} \leqslant \frac{exec_{opt}(\pi)}{e}$. Thus to prove

$$lim_{k\to\infty}\frac{MinTime_{\mathcal{N}}(k)}{k} \leq inf\{exec_{opt}(\pi_c) \mid \{\mathsf{i}\colon 1\} \to^{\pi_c}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}\}$$

if suffices to show that for every π_c there are $e \in \mathbb{N}$ and a run π such that $\{\mathsf{i}\colon e\} \to^{\pi} \{\mathsf{f}\colon e\}$ and

$$\frac{exec_{opt}(\pi)}{e} \leqslant exec_{opt}(\pi_c).$$

To prove it, fix $\pi_c = \frac{e'_1}{e_1}t_1, \frac{e'_2}{e_2}t_2, \ldots \frac{e'_n}{e_n}t_n$ and let $\rho_1\rho_2\ldots\rho_\ell$ be its continuous parallel execution that minimises the execution time. Let $e = e_1 \cdot e_2 \cdot \ldots e_n$. It suffices to define $\{\mathsf{i}\colon e\} \to^{\pi} \{\mathsf{f}\colon e\}$ and its parallel execution of execution time at most $e \cdot exec_{opt}(\pi_c) = e \cdot \sum_{1\leq i\leq \ell} exec(\rho_i)$.

The run π will consists of ℓ parts $\pi_1 \ldots \pi_\ell$ such that $\boldsymbol{R}_{\pi_i} = \boldsymbol{R}_{e\rho_i}$ for all $1 \leq i \leq \ell$. Note that $\boldsymbol{R}_{e\rho_i} \in \mathbb{N}^T$ by definition of e. We will show that every part π_i can be split into $e \cdot exec(\rho_i)$ blocks, which will define the desired parallel execution. Recall that $exec(\rho_i)$ is the largest $\frac{e'_j}{e_j}$ among the scalings in the block ρ_j. Thus $e \cdot exec(\rho_i)$ is a natural number. We define π_i by its parallel execution, *i.e.* decomposing it into blocks.

Given a vector $\boldsymbol{v} \in \mathbb{N}^T$ we write $\breve{\boldsymbol{v}}$ for $\breve{\boldsymbol{v}}(t) = 1$ if $\boldsymbol{v} > 0$ and $\breve{\boldsymbol{v}}(t) = 0$ otherwise. Let $\boldsymbol{v} = \boldsymbol{R}_{e\rho_i}$. Then the block decomposition is defined by the recursive procedure $f(\boldsymbol{v}) = \breve{\boldsymbol{v}}f(\boldsymbol{v} - \breve{\boldsymbol{v}})$, where $f(\boldsymbol{0}) = \varepsilon$. Here by $\breve{\boldsymbol{v}}$ we understand any run using its transitions (the order does not matter). Observe that the number of blocks equals $\|\boldsymbol{v}\|$.

Since every block contains at most one transition it remains to check the second condition of Definition 2. This follows easily since we know that ρ_i is a block and thus ${}^\bullet\boldsymbol{R}_{\rho_i} \leqslant \{\mathsf{i}\colon 1\} + \sum_{j<i} \Delta(\rho_j)$. By scaling both sides of the inequality by e we get the desired inequality.

($\geq$) It is sufficient to show that for every run $\{\mathsf{i}\colon k\} \rightarrow^{\pi} \{\mathsf{f}\colon k\}$ there is a continuous run π_c such that $\frac{exec_{opt}(\pi)}{k} \geq exec_{opt}(\pi_c)$. Suppose $\pi = t_1 t_3 \dots t_n$. We define $\pi_c = \frac{1}{k}t_1, \frac{1}{k}t_2 \dots \frac{1}{k}t_n$. The inequality is trivial. □

Now, we focus on computing $inf\{exec_{opt}(\pi_c) \mid \{\mathsf{i}\colon 1\} \rightarrow^{\pi_c}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}\}$. As the next step we show that the minimal execution time of a run depends only on its Parikh image. This fact however requires more insight into the theory of continuous reachability.

We formulate two lemmas that are essentially proven in [20].

Lemma 17 (Proposition 17 in [20]). *Let $\boldsymbol{m}, \boldsymbol{m'}$ be two markings in $\mathbb{Q}^P_{\geq 0}$ such that $\boldsymbol{m} \rightarrow^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}$. Then there are two configurations $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ and three continuous runs ρ_1, ρ_2, ρ_3 such that:*

- $\boldsymbol{m} \rightarrow^{\rho_1}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m}_1 \rightarrow^{\rho_2}_{\mathbb{Q}_{\geq 0}} m_2 \rightarrow^{\rho_3}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}$,
- $\boldsymbol{R}_{\rho_1} + \boldsymbol{R}_{\rho_2} + \boldsymbol{R}_{\rho_3} = \boldsymbol{R}_{\pi}$,
- $\forall_{p \in {}^\bullet\boldsymbol{R}_\pi} \boldsymbol{m}_1(p) > 0$,
- $\forall_{p \in \boldsymbol{R}_\pi^\bullet} \boldsymbol{m}_2(p) > 0$.

Proof. The assumptions of Proposition 17 in [20] require that there is no empty siphon. This is implied by nonredundancy, which we can assume (see Section 2). In that formulation we additionally have $\boldsymbol{m}_1 = \boldsymbol{m}_2$.

Formally, Proposition 17 in [20] does not deal with ρ_3 but this can be easily adjusted using Lemma 1. □

To formulate the second lemma we need one transformation $run(\boldsymbol{v})$. Let $T = \{t_1, t_2 \dots t_l\}$.. The transformation takes a $\boldsymbol{v} \in \mathbb{Q}^T_{\geq 0}$ and returns a continuous run $run(\boldsymbol{v}) = \boldsymbol{v}(t_1)t_1, \boldsymbol{v}(t_2)t_2 \dots \boldsymbol{v}(t_l)t_l$.

Lemma 18 (Lemma 12 in [20]). *Let $\boldsymbol{m}, \boldsymbol{m'}$ be two markings in $\mathbb{Q}^P_{\geq 0}$ and a vector $\boldsymbol{R} \in \mathbb{Q}^T_{\geq 0}$ such that:*

- $\Delta(\boldsymbol{R}) = \boldsymbol{m'} - \boldsymbol{m}$,
- $\forall_{p \in {}^\bullet\boldsymbol{R}} \boldsymbol{m}(p) > 0$,
- $\forall_{p \in \boldsymbol{R}^\bullet} \boldsymbol{m'}(p) > 0$.

Then there is $n \in \mathbb{N}$ such that the run

$$\pi = \underbrace{\frac{1}{n}\, run(\boldsymbol{R}) \frac{1}{n} run(\boldsymbol{R}) \dots \frac{1}{n} run(\boldsymbol{R})}_{n}$$

is enabled at $\boldsymbol{m}$. Precisely $\boldsymbol{m} \rightarrow^{\pi}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}$.

Proof. The statement of Lemma 12 in [20] is that $\boldsymbol{m} \rightarrow^{*}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'}$. However, the proof is that there is a run like π. □

Lemma 19. *Let $\boldsymbol{R} \in \mathbb{Q}^T_{\geq 0}$ be a vector and $S = \{\pi \mid \{\mathsf{i}\colon 1\} \rightarrow^{\pi}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ and $\boldsymbol{R}_\pi = \boldsymbol{R}\}$ be a set of continuous runs with the Parikh image $\boldsymbol{R}$. Let $max = \|\boldsymbol{R}\|$ be the maximal coordinate of the vector $\boldsymbol{R}$. Let $min = inf\{exec_{opt}(\pi) \mid \pi \in S\}$. Suppose S is not empty. Then $max = min$.*

Proof. First observe that $max \leqslant min$ as we have to execute one of the transitions max many times and min is well defined as S is not empty. Formally, we show that $max \leq exec_{opt}(\pi)$ for any $\pi \in S$. Suppose that $\pi = \rho_1\rho_2 \ldots \rho_h$ is the optimal partition of π into blocks and that $max = \boldsymbol{R}(t)$. The $exec_{opt}(\pi) = \sum_{i=1}^{h} exec_{opt}(\rho_i) \geq \sum_{i=1}^{h} \boldsymbol{R}_{\rho_i}(t) = max$.

Now we show inequality in the opposite direction, *i.e.* $min \leq max$. It is sufficient to show a family of runs π_ε such that $exec_{opt}(\pi_\varepsilon) \leq max + \varepsilon$ for any $\varepsilon > 0$. First, because of Lemma 17 there are runs and markings $\{\mathsf{i}\colon 1\} \rightarrow^{\pi_1}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m_1} \rightarrow^{\pi_2}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m_2} \rightarrow^{\pi_3}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ where

- $\boldsymbol{R}_{\pi_1} + \boldsymbol{R}_{\pi_2} + \boldsymbol{R}_{\pi_3} = \boldsymbol{R}$,
- $\forall p \in {}^\bullet\boldsymbol{v}$ holds $\boldsymbol{m_1}(p) > 0$,
- $\forall p \in \boldsymbol{v}^\bullet$ holds $\boldsymbol{m_2}(p) > 0$.

Further, there is $\delta > 0$ such that if we take runs $\rho_1 = \delta\pi_1$, $\rho_2 = (1-\delta)\pi_1\pi_2(1-\delta)\pi_3$, and $\rho_3 = \delta\pi_3$ then $\{\mathsf{i}\colon 1\} \rightarrow^{\rho_1}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'_1} \rightarrow^{\rho_2}_{\mathbb{Q}_{\geq 0}} \boldsymbol{m'_2} \rightarrow^{\rho_3}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}$ where

- $exec_{opt}(\rho_1)$ from $\{\mathsf{i}\colon 1\}$ and $exec_{opt}(\rho_3)$ from $\boldsymbol{m'_2}$ are smaller than $\varepsilon/2$,
- $\boldsymbol{R}_{\rho_1} + \boldsymbol{R}_{\rho_2} + \boldsymbol{R}_{\rho_3} = \boldsymbol{R}$,
- $\forall p \in {}^\bullet\boldsymbol{R}$ holds $\boldsymbol{m'_1}(p) > 0$,
- $\forall p \in \boldsymbol{R}^\bullet$ holds $\boldsymbol{m'_2}(p) > 0$.

Now because of Lemma 18 we know that ρ_2 can be of the form

$$\rho_2 = \underbrace{\frac{1}{n}\; run(\boldsymbol{R}_{\rho_2})\frac{1}{n}run(\boldsymbol{R}_{\rho_2})\ldots\frac{1}{n}run(\boldsymbol{R}_{\rho_2})}_{n}.$$

We claim that for big enough n we can partition it into n blocks $\frac{1}{n}\; run(\boldsymbol{R}_{\rho_2})$, and that this partition is a parallel execution from $\boldsymbol{m}'_1$ to $\boldsymbol{m}'_2$. Indeed, it just suffices to observe that $\frac{1}{n}\; run(\boldsymbol{R}_{\rho_i})$ are all blocks.

Further, we have following equality $exec(\rho_2) = n \cdot exec(\frac{1}{n}run(\boldsymbol{R}_{\rho_2})) = n \cdot \frac{1}{n}\|\boldsymbol{R}_{\rho_2}\|$. We define $\pi_\varepsilon = \rho_1\rho_2\rho_3$. Because $\boldsymbol{R}_{\rho 2} \leq \boldsymbol{R}$ we can write

$$\begin{aligned} exec_{opt}(\pi_\varepsilon) \leq exec_{opt}(\rho_1) + exec(\rho_2) + exec_{opt}(\rho_3) = \\ exec_{opt}(\rho_1) + \|\boldsymbol{R}_{\rho_2}\| + exec_{opt}(\rho_3) \leq \frac{1}{2}\varepsilon + \|\boldsymbol{R}\| + \frac{1}{2}\varepsilon = max + \varepsilon \end{aligned} \tag{7}$$

□

Theorem 5. *For a given nonredundant, generalised sound workflow net $\mathcal{N}$ good for $\{\mathsf{f}\colon 1\}$ we can compute $lim_{k\rightarrow\infty}\frac{MinTime_{\mathcal{N}}(k)}{k}$ in polynomial time.*

Note that Theorem 4 follows from this theorem. Indeed, if the workflow net is nonredundant and generalized sound then the net is good for $\{\{\mathsf{f}\colon 1\}\}$. Indeed, every place can be marked because of nonredundancy and from any reachable marking it is possible to reach $\{\mathsf{f}\colon 1\}$.

Proof (of Theorem 5). First, we can change the problem to computing $inf\{exec_{opt}(\pi_c) \mid \{\mathsf{i}\colon 1\} \rightarrow^{\pi_c}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}\}$ because of Lemma 16. Next, we can change the problem to computing $inf\{\|\boldsymbol{R}_\pi\| \mid \{\mathsf{i}\colon 1\} \rightarrow^{\pi}_{\mathbb{Q}_{\geq 0}} \{\mathsf{f}\colon 1\}\}$ because of Lemma 19. Let S be the set of Parikh images of continuous runs from $\{\mathsf{i}\colon 1\}$ to $\{\mathsf{f}\colon 1\}$. So we rephrase the previous task and write $inf\{\|\boldsymbol{v}\| \mid \boldsymbol{v} \in S\}$. This is equal to $inf\{\|\boldsymbol{v}\| \mid \boldsymbol{v} \in \overline{S}\}$, as $\|\boldsymbol{v}\|$ is a continuous function.

Since $\mathcal{N}$ good for $\{\mathsf{f}\colon 1\}$, the set of markings $\{\{\mathsf{f}\colon 1\}\}$ satisfies the preconditions of Lemma 8. Thus $\overline{S} = \{\boldsymbol{v} \mid \Delta(T)\boldsymbol{v} = \{\mathsf{f}\colon 1\} - \{\mathsf{i}\colon 1\}\}$.

Now, we split $\overline{S}$ into $|T|$ sets

$$S_t = \{\boldsymbol{v} \mid \boldsymbol{v} \in \overline{S} \text{ and } \boldsymbol{v}(t) \geqslant \boldsymbol{v}(t') \text{ for any } t' \in T\}.$$

Observe that $\boldsymbol{v} \in S_t$ if the following system of inequalities Sys_t is satisfied

$$Sys_t \coloneqq \Delta(T)\boldsymbol{v} = \{\mathsf{f}\colon 1\} - \{\mathsf{i}\colon 1\} \text{ and } \boldsymbol{v}(t) \geqslant \boldsymbol{v}(t') \text{ for all } t' \in T.$$

Further, we know that if $\boldsymbol{v} \in S_t$ then $\|\boldsymbol{v}\| = \boldsymbol{v}(t)$. So

$$inf\{\|\boldsymbol{v}\| \mid \boldsymbol{v} \in \overline{S}\} = min\left\{inf\{\boldsymbol{v}(t) : \boldsymbol{v} \in \overline{S_t}\} : t \in T\right\}.$$

We conclude since $inf\{\boldsymbol{v}(t) : \boldsymbol{v} \in \overline{S_t}\}$ is a solution of the linear program that minimises the function $f(\boldsymbol{v}) = \boldsymbol{v}(t)$ subject to Sys_t. $\square$