A Framework for Efficiently Deciding Language Inclusion for Sound Unlabelled WF-Nets

D.M.M. Schunselaar*, H.M.W. Verbeek*, W.M.P. van der Aalst*, and H.A. Reijers*

Eindhoven University of Technology,
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
{d.m.m.schunselaar, h.m.w.verbeek, w.m.p.v.d.aalst, h.a.reijers}@tue.nl

Abstract. We present a framework to efficiently check language inclusion between two sound unlabelled WF-nets. That is, to efficiently check whether every successfully terminating transition sequence of one sound unlabelled WF-net also is a successfully terminating transition sequence of a second sound unlabelled WF-net. Existing approaches for checking language inclusion are typically based on the underlying transition systems of both nets, and hence are subject to the well-known state-explosion problem. As a result, these approaches cannot check language inclusion on sound unlabelled WF-nets in polynomial time. Our framework allows for efficient language inclusion checks even if parallelism is present, by comparing specific net abstractions that can be computed and compared in polynomial time. For sound unlabelled WF-nets that are free-choice and do not contain loops, we prove that our approach is complete.

1 Introduction

Language inclusion refers to the problem of deciding whether a first language is included in a second language, that is, whether every word of the first language is also a word in the second language. For this paper, we assume that a language is described by a sound unlabelled WF-net (a subclass of Petri nets), and that every word in the language corresponds to a successfully terminating trace in that WF-net. As a result, the problem can now be rephrased as deciding whether every successfully terminating trace of a first sound unlabelled WF-net is also a successfully terminating trace of a second sound unlabelled WF-net.

There already is a large body of work on equivalences and/or similarities between Petri nets, e.g., [1–3]. However, in some cases we do not need to decide on language equivalence as language inclusion is already sufficient to investigate relevant properties. For example, suppose we have a large repository of Petri nets that are all different. To reduce the number of Petri nets we need to maintain, we might want to remove any Petri net that has less behaviour than some other

* This research has been carried out as part of the Configurable Services for Local Governments (CoSeLoG) project (http://www.win.tue.nl/coselog/).

Petri net. This requires us to determine whether the language of a Petri net is included in the language of another Petri net.

Conformance checking is another interesting application area for language inclusion. In conformance checking, one seeks to validate an event log against a Petri net, i.e., does every trace in the event log correspond to a successfully terminating transition sequence of the Petri net [4]? If we can create a Petri net that precisely captures the traces from the event log, then this question corresponds to whether the language of this created Petri net is included in the language of the original, given, Petri net.

Another interesting application of language inclusion comes from the CoSeLoG project[1]. In this project, we are cooperating with municipalities of different sizes and with different characteristics. If, for instance, a first, small, municipality wants to cooperate with a second, large, municipality, then the second municipality might have more generic process models (that is, Petri nets) that also support the first municipality. Although, language equivalence is unlikely to hold, language inclusion might very well hold, which is already sufficient for a fruitful cooperation. Apart from comparing the Petri nets of these municipalities, we can also combine them into a more generic Petri net [5], and check whether a Petri net of a third municipality is already included in this generic Petri net. If so, then this third municipality can be supported by this generic Petri net as well, while it might not have been supported by the Petri net of the first or the second municipality only. In the context of [5], we already have standardised the Petri nets, i.e., same activities names, and same level of abstraction.

The main problem with Petri-net-based language inclusion is that it is defined on the state-space underlying the Petri net (encoded as a transition system) and is PSPACE-complete. Due to the state-space explosion problem, such a transition system may be extremely large. Therefore, in general, it is computationally expensive to decide on language inclusion using the underlying transition systems.

We present a framework for deciding whether and how Petri-net-based language inclusion can be decided in a more efficient way for sound unlabelled WF-nets. The framework is based on the general pattern: If a sound unlabelled WF-net adheres only to some characteristics (e.g., is acyclic), then we can map this net to an alternative representation using a mapping function λ. Using a comparator $\leq_\lambda$, we may then efficiently decide whether language inclusion holds or not. As an example, Fig. 1 depicts a mapping function that maps every net to its set of transitions. Using the subset operator as comparator, we can already conclude that language inclusion does not hold, i.e., that the transition set of the first net is not a subset of the transition set of the second net.

For the case, sound unlabelled WF-nets that are free-choice and contain no loops, we prove that our approach is complete. If the net is not free-choice or contains loops, we present mapping functions and corresponding comparators that can signal that language inclusion does *not* hold.

[1] http://www.win.tue.nl/coselog/

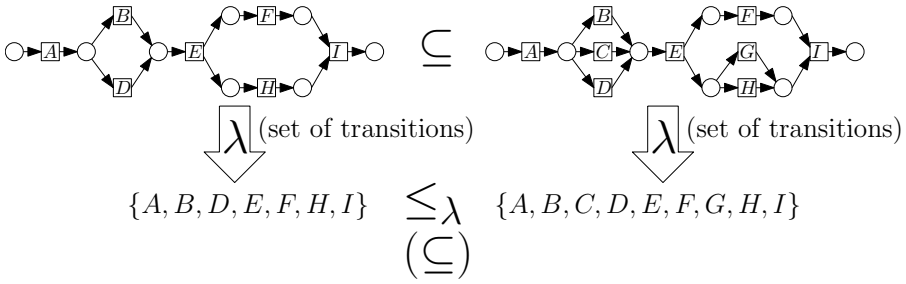

Fig. 1: General pattern of the framework: First, we map every net to an alternative representation (like its set of transitions); second, by efficiently comparing these alternative representations (like the standard subset comparator on the sets of transitions), language inclusion may be decided.

The structure of the paper is as follows. In Sect. 2, we present the approaches from literature and present the mappings found in literature. Preliminaries are provided in Sect. 3. Section 4 presents the general framework for our approach, which includes different mapping functions and comparators to tackle the language inclusion problem at hand. Finally, the conclusions and directions for future work are elaborated on in Sect. 5.

2 Related Work

There are solutions for checking whether the language of a first Petri net is included in the language of a second Petri net, see [6–8]. However, these solutions use the transition systems underlying both Petri nets. Unfortunately, these transition systems may become very large in the presence of parallelism. Therefore, as mentioned, we are designing a framework for deciding whether and how Petri-net-based language inclusion can be decided in a more efficient way. An important element of this framework is mapping a net to an alternative representation. In the remainder of this section, we present different candidates for such mappings as found in the literature. Note that this list is not exhaustive, but that our framework can easily incorporate other, new, mappings.

The first mapping we consider is based on the α-relation [9]. The α-relation denotes direct succession between activities and was originally designed for process discovery. In [10], a characterisation is given for the nets discoverable by the α-relation. Extensions have been made to the α-relation in [11] to be able to discover a larger set of nets.

The α-relation denotes direct succession, the relation used in behavioural profiles [3,12,13] denotes the eventual succession. Based on the behavioural profiles, work has been done for language equivalence, but language inclusion has not been addressed. A more generic approach based on behavioural profiles is presented in [1], where eventual succession is still considered, but the distance between occurrences of transition is bounded. Using behavioural profiles, [14]

considers an abstraction, but this abstraction is not language-equivalence preserving, i.e., the language of the abstraction is different from the language of the original net.

An approach similar to behavioural profiles is the approach using causal footprints [15]. Causal footprints denote causality between activities. In [15], causal footprints are used to deduce structural properties of sound WF-nets. Using causal footprints, [16] denotes the similarity between nets. This similarity can be used to deduce language equivalence (similarity of 1), but this approach is not tailored towards language inclusion.

Causal nets [4] allow us to model processes in such a way that causality between activities is made explicit. Language inclusion of causal nets has been defined in [17]. Furthermore, [4] presents an operational algorithm to transform a net to a causal net by transforming place and transition into activities.

Process nets [18] allow us to encode causal runs. Such a causal run is expressed as a marked graph. The approach was originally designed for the validation of nets. Therefore, no language equivalence or language inclusion is defined on process nets.

The last mapping we mention is the mapping from a net to its basis of semi-positive transition invariants [19]. The intuitive notion of such an invariant is that if we start at a given marking, and execute all the transitions from the invariant as many times as the invariant indicates, we return to the marking we started with. As such, every behavioural cycle in the Petri net is covered by such an invariant, but this not work the other way around: It may be possible that an invariant does not correspond to any behavioural cycle (due to the fact that we may not have sufficient tokens to execute exactly the transitions from the invariant).

In this paper, we consider the α-mapping, and the work that has been done on behavioural profiles, due to the fact they can be computed efficiently (in case of free-choice Petri nets [20]) on the structure of the net. The other abstractions can be used, but some do no always run in a time polynomial with respect to the size of the net. Causal footprints are closely related to behavioural profile. Therefore, using causal footprints yields similar results as using behavioural profiles. Hence, we do not use causal footprints. To use causal nets, we would need an approach to deal with the places introduced in the transformation as the same place might have different names in different process models. To use process nets, we would need an approach to deduce language inclusion between sets of sets of marked graphs. Furthermore, the number of process nets might be exponential in the number of choices, as there is no choice in a marked graph and every trace needs to be encoded in at least one marked graph. Finally, the main problem with semi-positive transition invariants is that the basis of invariant may be exponential in the size of the net.

3 Preliminaries

We use the following general notions: $\mathcal{A}$ denotes the set of actions, and $\mathbb{N}$ denotes the set of non-negative integers.

First, the notions of language, word of a language, and language inclusion are defined. A word is a sequence of actions from $\mathcal{A}$, a language is a set of words, and a language is included in another language if and only if all the words of the first language are part of the second language.

Next, we introduce the notion of a transition system, and the language of a transition system.

Definition 1 (Transition System). *A Transition System $TS = (S, \rightarrow, \alpha, \omega)$ is a tuple where:*

- *S is the set of states,*
- *$\rightarrow \subseteq S \times \mathcal{A} \times S$ is the set of transitions,*
- *$\alpha \in S$ is the initial state,*
- *$\omega \in S$ is the final state.*

A word of a transition system is a sequence of transitions starting from the initial state and ending in the final state. The language of a transition system is the set of all words which can be produced by the transitions system.

Now we introduce some general Petri net concepts.

Definition 2 ((Unlabelled) Petri net). *A Petri net $N = (P, T, F)$ is a tuple where:*

- *P is a set of places,*
- *$T \subseteq \mathcal{A}$ is a set of transitions, such that $P \cap T = \emptyset$,*
- *$F \subseteq (P \times T) \cup (T \times P)$ is a flow relation.*

The preset of a transition/place n, denoted by $\bullet n$, is the set of places/transitions in $\bigcup i \in P \cup T : (i, n) \in F$. The postset of a transition/place n, denoted by $n\bullet$, is the set of places/transitions in $\bigcup i \in P \cup T : (n, i) \in F$. A Petri net is free-choice if and only if for every two transitions, either the presets of every two transitions is disjoint, or they are the same.

Definition 3 (Bag over Set). *Let S be a set. A bag over S is a function from S to the natural numbers $\mathbb{N}$ such that only a finite number of elements from S is assigned a non-zero function value.*

Note that a finite set is also a bag, namely the function assigning 1 to every element in the set and 0 otherwise.

The set of all bags over S is denoted $\mathcal{B}(S)$. For a bag b over S and $s \in S$, $b(s)$ denotes the number of occurrences of s in b, often called the *cardinality* of s in b. We use brackets to explicitly enumerate a bag and superscripts to denote cardinalities. For example, $[a^2, b^3, c]$ is the bag with two a's, three b's and one c; the bag $[s^2 \mid P(s)]$, where P is a predicate on S, contains two elements s for

every s such that $P(s)$ holds. The sum of two bags b_1 and b_2, denoted $b_1 + b_2$, is defined as $[s^n \mid s \in S \wedge n = b_1(s) + b_2(s)]$. The difference of two bags b_1 and b_2, denoted $b_1 - b_2$, is defined as $[s^n \mid s \in S \wedge n = (b_1(s) - b_2(s)) \max 0]$. Bag b_1 is a subbag of b_2, denoted $b_1 \leq b_2$, if and only if $\forall s \in S : b_1(s) \leq b_2(s)$.

A marking of a Petri net is a bag over the set of places. A transition t is enabled in a marking M, denoted by $M[t\rangle$, if and only if the preset of t is a subbag of M. Firing an enabled transition t from a marking M resulting in another marking M' is denoted by $M[t\rangle M'$. Marking M' is obtained by taking the difference between M and the preset of t and summing it with the postset of t, i.e. $M' = M - {\bullet t} + t{\bullet}$.

Definition 4 (Transition System of a Petri net). *Let $N = (P, T, F)$ be a Petri net, let M_i be a marking of N, and let M_o be a marking of N, then the transition system $TS = TS(N, M_i, M_o)$, belonging to N, is constructed as follows:*

- $\alpha = M_i$,
- $\omega = M_o$,
- *S is the smallest set X, such that:*
  - $\alpha \in X$, $\omega \in X$, *and*
  - *if $s \in X \wedge s[t\rangle s'$ for some $t \in T$ and $s' \in \mathcal{B}(P)$, then $s' \in X$.*
- $\rightarrow = \{(s_1, t, s_2) \mid s_1, s_2 \in S \wedge t \in T \wedge s_1[t\rangle s_2\}$.

The language of a Petri net is the language of the transition system as constructed in Def. 4.

As mentioned, we limit ourselves to WF-nets [21]. A WF-net has a unique input and output place, i.e., the preset of the input place is empty and the postset of the output place is empty. Furthermore, every transition is on a path between the input place and the output place. With $TS(N)$, we denote the transition system belonging to the WF-net N with unique input place i and output place o (i.e., $TS(N) = TS(N, [i], [o])$).

We require a WF-net to be sound [21].

Definition 5 (Soundness [21]). *Let $N = (P, T, F)$ be a WF-net, let $TS = (S, \rightarrow, \alpha, \omega)$ be a transition system belonging to N, i.e., $TS = TS(N)$, then N is sound if and only if (note that α is $[i]$ and ω is $[o]$):*

- $\forall m \in S : (m, \omega) \in \rightarrow^*$, *where $\rightarrow^*$ is the transitive closure of $\rightarrow$,*
- $\forall t \in T : \exists s, s' \in S : (s, t, s') \in \rightarrow$.

In the remainder of this paper, $\mathcal{N}$ denotes the set of all sound unlabelled WF-nets. Note that whenever we say WF-net the unlabelled variant is intended unless stated differently.

As mentioned in the related work section, there exist approaches to compute language inclusion on the transition system. However, this computation does not need to be polynomial in the size of the Petri net. Therefore, we provide in Sect. 4 an approach to deduce language inclusion based on the structure of the Petri net.

4 Approach

In this section, we determine in which situations language inclusion can be efficiently computed on the transition system, and what can be done to avoid using the transition system in other situations. For the latter, we use our framework, i.e., when the net has certain characteristics, this net is mapped onto an alternative representation. Using a comparator, language inclusion can be decided, similar to the example given in the introduction.

Whether it is computationally efficient to use the transition system and, if not, which mapping can be used, depends heavily on the constructs used in the Petri nets, i.e., on the characteristics of both nets. For example, if both nets are sequential and acyclic, then language inclusion can be efficiently computed on the transition systems. If both nets are acyclic, then the *eventually-follows* relation between transitions may provide valuable information on language inclusion, where the eventually-follows relation between two transitions denote that the first can be followed by the second.

The remainder of this section is organised as follows: We first define an abstract mapping, then introduce characteristics, and finally define a comparator. Afterwards, we revisit the used (concrete) mappings from literature. Having the concrete mappings in place, the characteristics are defined, and a polynomial-time algorithm is provided to compute these characteristics on the free-choice WF-nets. Using the mappings and characteristics, we provide comparators and accompanying abstractions together, as they are tightly linked.

4.1 Abstract framework

In our abstract framework, an abstraction consists of three elements: (1) a mapping function, mapping a Petri net into an abstract representation, (2) a set of characteristics denoting when this abstraction can be applied, and (3) a comparator to compare the abstract representations (mapping to the booleans). The first and the last follow straightforward from the previous explanations, but the second requires some extra explanation and mainly on the *can be applied* part.

We say a Petri net adheres to certain characteristics if and only if these characteristics are valid for this Petri net. This allows abstractions to be used when characteristics are added, i.e., the addition of an characteristic does not invalidate the results presented in this paper.

Two different kinds of abstractions are considered: (1) a positive abstraction between N and N' denoting that: If the comparator yields true, N is guaranteed to be language included in N', and (2) a negative abstraction between N and N' denoting that: If the comparator yields false, N is guaranteed to be *not* language included in N'. This allows our framework to be more flexible.

4.2 Mappings

Having the abstract framework with abstractions in place, we now present the specific mappings used (based on the reasoning in Sect. 2) in the remainder of

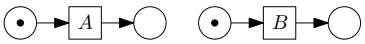

Fig. 2: Two WF-nets with different languages but with the same mapping.

this paper. Please note that we do not claim that these mappings are exhaustive: Other mappings may exist that allow for efficient positive and/or negative abstractions that are not covered by this paper. However, these mappings (and corresponding comparator) can be easily added to our framework.

The T-mapping provides us with the set of transitions from a WF-net, as explained in the introduction (we denote this set as $\lambda_T(N)$).

The α-mapping denotes that two transitions can follow each other directly in a Petri net N, i.e., if (t_1, t_2) is part of the mapping (denoted by $\lambda_\alpha(N)$), then there is a trace in which t_1 is *directly* followed by t_2.

The ∞-mapping denotes that two transitions can follow each other eventually in a Petri net N, i.e., if (t_1, t_2) is part of the mapping (denoted by $\lambda_\infty(N)$), then there is a trace in which t_1 is *eventually* followed by t_2.

The presence of loops may be problematic for the ∞-mapping. For this reason, we also include the k-mapping (generalisation of the α-mapping), which denotes that there are *at most* k other activities between two transitions in a Petri net N. As such, this mapping (denoted by $\lambda_k(N)$) corresponds to the eventually-follows-within-k-steps-relation between transitions. As a result, transitions that occur in a loop of length of at least k are not automatically related in both ways by this mapping, e.g., assume a and b are in a loop, then (a, b) in $\lambda_k(N)$ but (b, a) does not need to be in $\lambda_k(N)$ while (b, a) would have been included in $\lambda_\infty(N)$.

These latter three specific mappings are all based on relations between two transitions, which poses a possible problem if a net only contains traces that contain only a single transition. Clearly, for such a net these follows relations are all empty, which yields equivalent mappings for the nets in Fig. 2. For this reason, we add artificial start and end activities $(\top, \bot)$ to any net. Note that this does not limit the expressivity of the process models.

4.3 Characteristics

Since our framework depends on the exact definition of the characteristics, this section defines the characteristics of sound WF-nets, and shows that we can compute these characteristics efficiently in case of free-choice nets.

The *sequential characteristic* denotes that from a state firing one transition, means the other transitions enabled in the state are no longer enabled.

Definition 6 (Sequential Characteristic). *Let $N = (P, T, F) \in \mathcal{N}$ be a sound WF-net, let $TS = (S, \rightarrow, \alpha, \omega)$ be the transition system belonging to N, i.e., $TS = TS(N)$, then N is sequential if and only if:* $\forall t_1, t_2 \in T, s \in S : \neg(\bullet t_1 + \bullet t_2 \leq s)$

The *acyclic characteristic* denotes that it is not possible to reach a previously visited state.

Definition 7 (Acyclic Characteristic). *Let $N = (P, T, F) \in \mathcal{N}$ be a sound WF-net, let $TS = (S, \rightarrow, \alpha, \omega)$ be the transition system belonging to N, i.e., $TS = TS(N)$, then N is acyclic if and only if: $\forall n \in \mathbb{N}, s_0, \ldots, s_n \in S, t_1, \ldots, t_n \in T : n < 1 \vee s_0 \neq s_n \vee \exists 1 \leq k \leq n : (s_{k-1}, t_k, s_k) \notin \rightarrow$.*

In general, these characteristics are defined on the transition system, and may be inefficient to compute. However, for free-choice nets we can compute them in an efficient way, as the following theorems show.

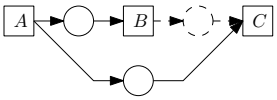

Fig. 3: With the dashed place and arrows, we have a sequential construct, else not.

In [1](Def. 26), a polynomial time algorithm is presented to compute the minimal structural successor between places and transitions (MSS) for a free-choice net, i.e., the minimal set of transitions and places between the execution of two transitions. For some characteristics, we require that the considered transitions can follow each other *directly*, thus a MSS has a size of 1, i.e., only a single place is between the transitions. Consider Fig. 3, without the dashed parts, B and C are non-sequential. However, with the dashed parts, B and C are sequential. In the first case, the size of the MSS between A and C is 1, in the latter case the size of the MSS between A and C is 3.

Theorem 1. *Let $N = (P, T, F)$ be a sound, free-choice WF-net, then N is sequential if and only if: $\forall u_1, u_2, u_3 \in T : u_1 \bullet \cap \bullet u_2 = \emptyset \vee u_1 \bullet \cap \bullet u_3 = \emptyset \vee \bullet u_2 \cap \bullet u_3 \neq \emptyset \vee |MSS(u_1, u_2)| \neq 1 \vee |MSS(u_1, u_3)| \neq 1$.*

Proof. We need to prove: $(\forall t_1, t_2 \in T, s \in S : \neg(\bullet t_1 + \bullet t_2 \leq s)) \Leftrightarrow (\forall u_1, u_2, u_3 \in T : u_1 \bullet \cap \bullet u_2 = \emptyset \vee u_1 \bullet \cap \bullet u_3 = \emptyset \vee \bullet u_2 \cap \bullet u_3 \neq \emptyset \vee |MSS(u_1, u_2)| \neq 1 \vee |MSS(u_1, u_3)| \neq 1)$.

- $\Rightarrow$ We prove this by contradiction, assume the left-hand side is true, but the right-hand side is false, thus $\exists u_1, u_2, u_3 \in T : u_1 \bullet \cap \bullet u_2 \neq \emptyset \wedge u_1 \bullet \cap \bullet u_3 \neq \emptyset \wedge \bullet u_2 \cap \bullet u_3 = \emptyset \wedge |MSS(u_1, u_2)| = 1 \wedge |MSS(u_1, u_3)| = 1$. From the right-hand side it follows that u_2 and u_3 are both enabled after executing u_1 (by soundness, we know u_1 can fire). Furthermore, u_2 and u_3 can fire directly after u_1 by the MSS, and because their presets are disjoint there is no ordering between u_2 and u_3. Hence, there is a state s such that $\bullet u_2 + \bullet u_3 \leq s$.
- $\Leftarrow$ Again we prove this by contradiction, assume the right-hand side is true, but the left-hand side is false, thus $\exists t_1, t_2 \in T, s \in S : \bullet t_1 + \bullet t_2 \leq s$. Since

we have a sound WF-net, we start with a single token in the input place. This means there is a sequence of transitions to reach a state s where two transitions are enabled say t_1 and t_2, let s be the first state. Then there is a transition t to reach state s, let t fire from state s', i.e., $s'[t\rangle s$. By the fact that t_1 and t_2 can fire directly from s, we know that $|MSS(t, t_1)| = 1$ and $|MSS(t, t_2)| = 1$. Now it remains to prove that the first 3 clauses are false, that is: $u_1 \bullet \cap \bullet u_2 \neq \emptyset$ and $u_1 \bullet \cap \bullet u_3 \neq \emptyset$ and $\bullet u_2 \cap \bullet u_3 = \emptyset$.

- Assume $t \bullet \cap \bullet t_1 = \emptyset$, this means that $s'[t\rangle$ and $s'[t_1\rangle$. Then there are two options: $\bullet t + \bullet t_1 \leq s'$, this means s' is an earlier state in which two transitions are enabled (not possible by assumption), and $\neg(\bullet t + \bullet t_1 \leq s')$. We now obtain by free-choiceness that $\bullet t = \bullet t_1$. From the fact that t_1 is still enabled after firing t, and the fact that $t \bullet \cap \bullet t_1 = \emptyset$, we obtain that $\bullet t + \bullet t_1 \leq s'$, this is not possible due to the free-choiceness and soundness as this allows t_1 to fire twice, and hence (by free-choiceness) it is possible to mark the output place with two tokens.
- The proof for $t \bullet \cap \bullet t_2$ goes analogous.
- We only have the case where $\bullet t_1 \cap \bullet t_2 \neq \emptyset$, thus $\bullet t_1 = \bullet t_2$. Again by soundness and free-choiceness it follows that t_1 can fire twice from state s, hence it is not sound.

□

Theorem 2. *Let $N = (P, T, F)$ be a sound, free-choice WF-net, then N is acyclic if and only if:* $\forall t_1, \ldots t_n \in T : \bullet t_1 \cap t_n \bullet = \emptyset \vee (\exists 1 \leq k < n : t_k \bullet \cap \bullet t_{k+1} = \emptyset)$.

Proof. We need to prove: $(\forall n \in \mathbb{N}, s_0, \ldots, s_n \in S, t_1, \ldots, t_n \in T : n < 1 \vee s_0 \neq s_n \vee \exists 1 \leq k \leq n : (s_{k-1}, t_k, s_k) \notin \rightarrow) \Leftrightarrow (\forall t_1, \ldots t_n \in T : \bullet t_1 \cap t_n \bullet = \emptyset \vee (\exists 1 \leq k < n : t_k \bullet \cap \bullet t_{k+1} = \emptyset))$.

$\Rightarrow$ This follows from the free-choice property, i.e., as soon as we can mark a place in the preset of a transition, then this token can only be removed after this transition has been enabled. As a result, if we mark a place in the preset of a previously enabled transition, then we can do this an infinite amount of times as this transition has to be able to fire. Since the state-space is finite (bounded net due to soundness) this means we have a path to a previously visited state.

$\Leftarrow$ When we do not have a sequence of transition to mark a place again, this means as soon as this place has been marked it will never be marked again. Hence, we cannot revisit a state.

□

From Thm. 1 and Thm. 2, we can conclude that, for free-choice WF-nets the characteristics can be efficiently computed, i.e., in polynomial time. As a result, we can use these characteristics in our framework.

4.4 Abstractions

This section builds on the previous sections by presenting concrete instantiations of the framework, consisting of selected abstractions, that can be used to efficiently decide language inclusion for nets with certain characteristics. Therefore, in the remainder, we display different approaches, each tailored towards nets with certain characteristics. We start with the approach which does not require an abstraction of the Petri net.

No abstraction needed In case the Petri net is sequential and free-choice, the transition systems will not suffer from the state-space-explosion problem. For this reason, we do not need an abstraction, as we can efficiently decide the language inclusion problem on the transition systems.

Negative abstraction using T-mapping The following negative abstraction is valid for all sound WF-nets.

Definition 8 (T-Comparator). *Let $N = (P, T, F)$ and $N' = (P', T', F')$ be sound WF-nets, let λ_T be the T-mapping, then $\lambda_T(N) \leq_T \lambda_T(N')$ if and only if: $\lambda_T(N) \subseteq \lambda_T(N')$*

Theorem 3. *$(\lambda_T, \leq_T, \emptyset)$ is a negative abstraction.*

Proof. As both nets are sound WF-nets, all transitions of N are included in $\mathcal{L}(N)$. That is, for every $t \in T$ there exists a trace $\sigma \in \mathcal{L}(N)$ such that $t \in \sigma$. Clearly, if there exists a transition $t \in T$ such that $t \notin T'$, then the traces in N that contain t are not traces of N'. Hence, language inclusion cannot hold. □

Negative abstraction using α-mapping The following negative abstraction is also valid for all sound WF-nets. However, to our knowledge, only for free-choice WF-nets there is a polynomial-time algorithm to compute the α-mapping. Hence we can use the negative abstraction to efficiently decide language inclusion either if we have a free-choice net or if the required α-mappings are given.

Definition 9 (α-Comparator). *Let $N = (P, T, F)$, $N' = (P', T', F')$, be sound WF-nets, let λ_α be the α-mapping, then $\lambda_\alpha(N) \leq_\alpha \lambda_\alpha(N')$ if and only if: $\lambda_\alpha(N) \subseteq \lambda_\alpha(N')$*

Theorem 4. *$(\lambda_\alpha, \leq_\alpha, \emptyset)$ is a negative abstraction.*

Proof. We need to prove that $\neg(\lambda_\alpha(N) \leq_\alpha \lambda_\alpha(N')) \Rightarrow \neg(\mathcal{L}(N) \subseteq \mathcal{L}(N'))$ which can be rewritten to $\mathcal{L}(N) \subseteq \mathcal{L}(N') \Rightarrow \lambda_\alpha(N) \leq_\alpha \lambda_\alpha(N')$ which follows straightforward from the definition of λ_α. □

Theorem 4 yields the following result: Given two sound WF-nets and their corresponding α-mappings, Def. 9 can be computed in polynomial time. Hence, Thm. 4 yields in polynomial time whether the language is *not* included, i.e., if

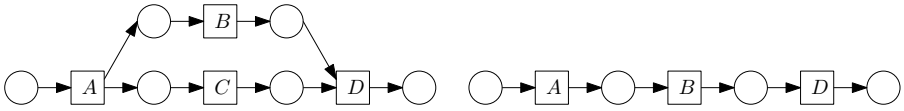

Fig. 4: Two process models where we have a false positive w.r.t. language inclusion if we simply use inclusion of the α-relation.

$\lambda_\alpha(N) \subseteq \lambda_\alpha(N')$ yields *false* then the inclusion does not hold. In case no α-mapping is provided, we can, for free-choice WF-nets, compute the α-mapping efficiently and determine if language inclusion does not hold.

Using the α-mapping, we did not obtain a positive abstraction, as we cannot do an inclusion of the relations between the different process models. Consider the models in Fig. 4. Here, it is easy to see that the relations of the right-hand model are a subset of left-hand model. That is, in the left model we have the relations $\lambda_\alpha(N) = \{(A, B), (A, C), (B, C), (C, B), (B, D), (C, D)\}$, and in the right model we have the relations: $\lambda_\alpha(N') = \{(A, B), (B, D)\}$. However, the word ABD is included in the right model, but not in the left model.

The problem is that we are omitting related activities, e.g., the A is followed, in Fig. 4, by a C, while with omitting relations, we also omit these dependencies.

Consider Fig. 5, at the bottom, for both models, we have transformed the parallel execution into a sequential execution such that λ_α(right model) $\subseteq \lambda_\alpha$(left model). In the top two models, we have done the same, but this yields a model allowing words not possible in the other model. So, we need to find an inclusion operator, which can either differentiate between both models or considers both models not to be included in the other. In the first case, we have false negatives (see bottom model), in the second case, we have false positive (see top model). The problem is: The α-mapping considers two activities exclusive if they cannot follow each other directly. Hence, we have to look at the transitivity of these relations.

Positive abstraction using ∞-mapping In case the WF-net is acyclic, we can use the ∞-mapping instead of the α-mapping. Using this mapping, we can compute the transitive closure of the α-relations to deduce the relations between all activities. However, similar to the α-mapping, a simple inclusion of the sets is not strong enough to deduce language inclusion. Therefore, we need an approach which takes into account the alternatives between different activities. Alternative activities are activities which do not occur together in a trace. So, if an activity does not occur in a word, then an alternative activity occurs in that word.

Theorem 5. *Let $N = (P, T, F)$ be a sound acyclic free-choice net, then:* $\forall a \in T, \sigma \in \mathcal{L}(N) : a \notin \sigma \Rightarrow \exists b \in T : b \in \sigma \wedge (a, b) \notin \lambda_\infty(N) \wedge (b, a) \notin \lambda_\infty(N)$.

Proof. Assume there is an $a \in T$ and $a \notin \sigma$ (by soundness we know a can be executed). This means there is a choice to not execute a. Since we have free-choice nets, this means somewhere there are two transitions t_1, t_2 ($\bullet t_1 = \bullet t_2$),

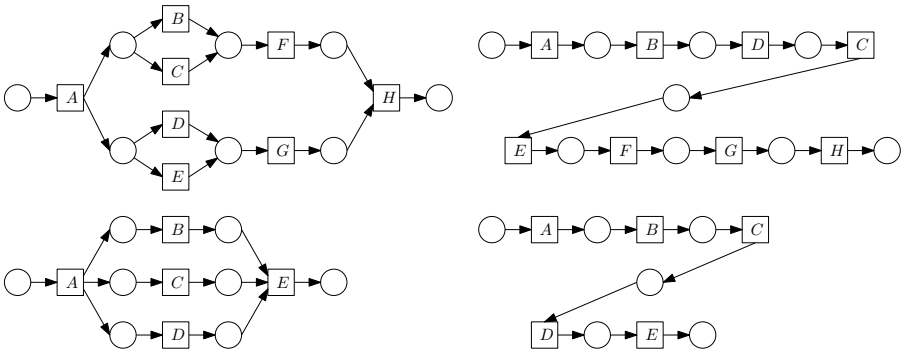

Fig. 5: Using the α-relation, we cannot differentiate between valid transformations from parallel to sequential (bottom) and invalid transformations from parallel to sequential (top).

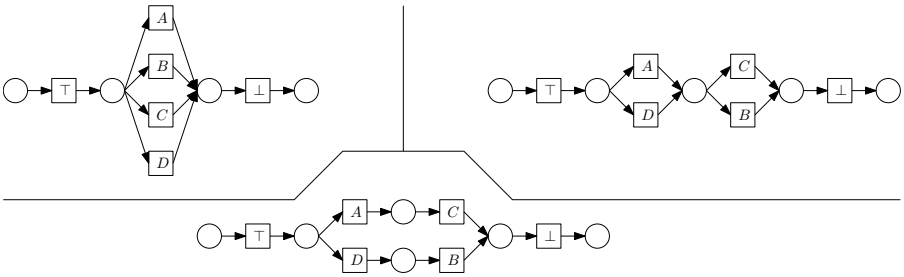

Fig. 6: The left and middle model are language included in the right model without clause 3 in Def. 10.

such that after t_1 a can still follow, but after executing t_2 a cannot follow. Due to the fact that the net is acyclic and free-choice, a cannot precede t_2, i.e., if a can, via firing some transitions, mark the preset of t_2 then also the preset of t_1 is marked and hence a can execute again. Since a is not in σ, we have chosen t_2, and thus $t_2 \in \sigma$. Also because a cannot follow t_2, and t_2 cannot follow a, $(a, t_2) \notin \lambda_\infty(N) \wedge (t_2, a) \notin \lambda_\infty(N)$. □

Now the comparator denotes that the activities have to be the same, the relations have to be a subset (similar to Def. 9), and there is always at least one alternative activity remaining. We have to require clause 3, as this allows us to only remove alternative activities. Consider Fig. 6, the lower model is language included in the right model, and this is valid since an alternative activity for B (namely C) follows A. However, the left model is not language included in the lower model, as there is no alternative activity for C following A. Without clause 3, the comparator would have denoted that the left model is included in the lower model.

Definition 10 (∞-Comparator). *Let $N = (P, T, F)$, $N' = (P', T', F')$ be sound FC-nets adhering to the {Acyclic} characterisation, let λ_∞ be the ∞-mapping, then $\lambda_\infty(N) \leq_\infty \lambda_\infty(N')$ if and only if:*

1. $T = T'$
2. $\lambda_\infty(N) \subseteq \lambda_\infty(N')$
3. $(\forall (a,b) \in \lambda_\infty(N') : (a,b) \in \lambda_\infty(N) \vee (b,a) \in \lambda_\infty(N) \vee (\exists c \in T : ((a,c) \in \lambda_\infty(N) \vee (c,a) \in \lambda_\infty(N)) \wedge (b,c) \notin \lambda_\infty(N') \wedge (c,b) \notin \lambda_\infty(N')))$

Theorem 6. $(\lambda_\infty, \leq_\infty, \{Acyclic\})$ *is a positive abstraction.*

Proof. We need to prove: $\lambda_\infty(N) \leq_\infty \lambda_\infty(N') \Rightarrow \mathcal{L}(N) \subseteq \mathcal{L}(N')$.

We prove this by contradiction, thus $\mathcal{L}(N) \not\subseteq \mathcal{L}(N')$ *but* $\lambda_\infty(N) \leq_\infty \lambda_\infty(N')$. By definition, this means there is a trace $\sigma \in \mathcal{L}(N)$ such that $\sigma \notin \mathcal{L}(N')$.

Now it remains to prove that this σ does not exist. We prove this by induction on the prefix of the trace. Let $\sigma = \langle t_1, \ldots, t_{k-1}, t_k, \ldots, t_m \rangle$ be a trace, the prefix of σ (denoted by σ_{prefix}) are transitions before position k (i.e., $\sigma_{\text{prefix}} = \langle t_1, \ldots, t_{k-1} \rangle$). It remains to show that if the prefix of σ corresponds to a prefix of a trace $\sigma_{N'} \in \mathcal{L}(N')$ then t_k has to be directly executable in N' after having executed this prefix.

Base case $k = 1$. In this case, the prefix is empty and by definition we always start with an unique start activity, hence both models can execute $\top$.

Inductions hypothesis: assume $\sigma_{\text{prefix}} = \langle t_1, \ldots t_{k-1} \rangle$ corresponds to a prefix of a trace in N', then there are two options: (1) t_k cannot follow σ_{prefix} in N', or (2) t_k cannot follow σ_{prefix} directly in N', i.e., there is at least on transition in between σ_{prefix} and t_k.

(1) If t_k cannot follow σ_{prefix} in N', then either $t_k \notin T'$ (not possible due to clause 1) or there is an alternative activity $t_i \in \sigma$ (Thm. 5). Note that this alternative activity has to be in σ_{prefix}, because else from σ_{prefix} it cannot follow that t_k cannot follow σ_{prefix} in N'. This latter option cannot happen due to clause 2 (else $(t_i, t_k) \in \lambda_\infty(N)$ *but* $(t_i, t_k) \notin \lambda_\infty(N')$). Hence t_k must be able to follow σ_{prefix} in N'.

(2) When t_k cannot directly follow σ_{prefix} this means there is an activity between σ_{prefix} and t_k in N'. We denote this activity with $b \in T'$. We know that $(b, t_k) \in \lambda_\infty(N')$ but also that $(t_k, b) \notin \lambda_\infty(N')$ (else b does not need to be between σ_{prefix} and t_k, by free-choiceness and soundness). Furthermore, by the absence of loops and soundness, we know that $b \notin \sigma_{\text{prefix}}$ (else b can be followed by b in N'). Then there are two options; (a) either b can follow σ_{prefix} in N, or (b) b cannot follow σ_{prefix} in N.

(a) Since $(t_k, b) \notin \lambda_\infty(N')$ it follows from clause 2, that $(t_k, b) \notin \lambda_\infty(N)$. This means that if b can follow σ_{prefix} in N, it has to be executed before t_k is executed. In σ this is not the case, i.e., there is no activity executed between σ_{prefix} and t_k, and Thm. 5 yields that there has to be an alternative activity for b in σ (note that it does not need to be an alternative activity for b in N') and by free-choiceness it has to be before t_k, thus in σ_{prefix}. From this it follows that b cannot follow σ_{prefix} in N. We only need to prove that part (b) yields a contradiction.

(b) Thm. 5 yields that somewhere we have made a choice in σ_{prefix} to not execute b in N, we call the activity after making this choice a. Since σ_{prefix} can be followed by b in N', we obtain that $(a, b) \in \lambda_\infty(N')$, but also $(a, b) \notin \lambda_\infty(N)$ (note that $(b, a) \notin \lambda_\infty(N)$, since this violates the acyclic property). From clause 3, we obtain there has to be an activity c such that $(a, c) \in \lambda_\infty(N) \vee (c, a) \in \lambda_\infty(N)$ and $(b, c) \notin \lambda_\infty(N') \wedge (c, b) \notin \lambda_\infty(N')$. Finally, from Thm. 5, we obtain that there has to be an alternative activity for b in σ, which c also is. Now it remains to show that c is in σ_{prefix}. By free-choiceness, we obtain that c has to be executed before t_k. If c can be executed after t_k, then dependent on the activities chosen before t_k, c can either be executed or not, which violates the free-choice property. Since $(c, t_k) \in \lambda_\infty(N)$, it follows that c has to be in σ_{prefix}, and thus no b can be between σ_{prefix} and t_k in N'.

From this it follows that both models can execute t_k directly after prefix σ_{prefix}, and hence both models are able to perform σ (note that termination is taken care of by including $\perp$ as unique end activity) and thus it follows that σ is also in $\mathcal{L}(N')$. Thus by contradiction, we have shown that $\lambda_\infty(N) \leq_\infty \lambda_\infty(N') \Rightarrow \mathcal{L}(N) \subseteq \mathcal{L}(N')$. □

In case $T = T'$, we can decide language inclusion based on the ∞-mapping. However, in general it is the case that $T \neq T'$. Therefore, we extend our approach to the case in which $T \subset T'$. Note that we do not need to consider the case that $T' \subset T$ as language inclusion cannot hold.

In order to achieve $T = T'$, we have the following reduction on N':

- Remove all transitions $t \in (T' \setminus T)$ from N'
- Remove all places not connected to any transition

If the reduced net is sound then we can use Def. 10 on the transformed net with $T = T'$. It is easy to see that language inclusion is preserved as we only remove transitions not present in T, i.e., we do not reduce the language w.r.t. the overlap with T. If the resulting model is not sound, then a part of the language of N cannot be part of N'.

Apart from showing the soundness of the abstraction using the ∞-mapping, we also want to show the completeness. This means that if the language of N is included in N', then our comparator always yields true.

Theorem 7. *$(\lambda_\infty, \leq_\infty, \{Acyclic\})$ is complete, i.e., for two acyclic free-choice WF-nets N and N', we have $\lambda_\infty(N) \leq_\infty \lambda_\infty(N') \Leftarrow \mathcal{L}(N) \subseteq \mathcal{L}(N')$.*

Proof. We prove this by contradiction. Assume $\mathcal{L}(N) \subseteq \mathcal{L}(N')$, but $\lambda_\infty(N) \leq_\infty \lambda_\infty(N')$ does not yield true.

Then, at least one of the clauses has to be false. Clause 1 and 2 follow in a straightforward way. Assume clause 1 is false, then there is an activity $x \in T$ which is not in T'. Since T is sound, this means x occurs in at least one trace and this trace cannot occur in $\mathcal{L}(N')$. Assume clause 2 is false, this means there is a relation between 2 activities $x, y \in T$ such that $(x, y) \in \lambda_\infty(N)$, which is not in $\lambda_\infty(N')$. By definition, this means there is a word such that

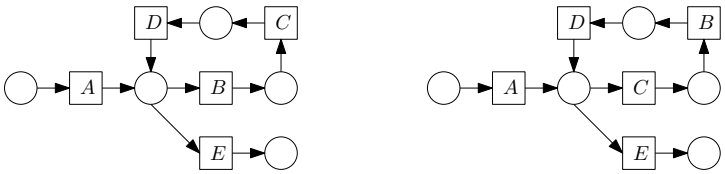

Fig. 7: Two process models where using the ∞-mapping gives a false positive.

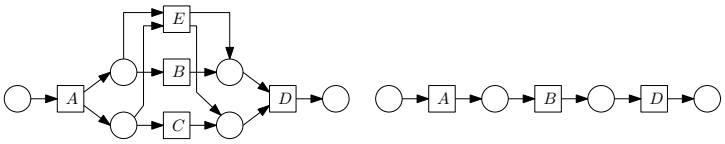

Fig. 8: Naive inclusion does not work for the k-mapping.

$\sigma = \langle \ldots, x, \ldots, y, \ldots \rangle$, but this trace cannot be in $\mathcal{L}(N')$, else this relation was also included in $\lambda_\infty(N')$.

Assume clause 3 is not valid, this means that: $\exists (a, b) \in \lambda_\infty(N') : (\forall c \in T : \neg((a, c) \in \lambda_\infty(N) \vee (c, a) \in \lambda_\infty(N)) \vee (c, b) \in \lambda_\infty(N') \vee (b, c) \in \lambda_\infty(N'))$ has to hold (negation of the clause 3). Recall that due to $\top$ and $\bot$, $\lambda_\infty(N')$ cannot be empty.

We choose an a and b, the universal quantifier denotes that either the activity c is exclusive to a, or can occur together with b in a trace. Thus, there is no alternative for b to occur together with a in the trace. If we chose $b = c$ then we have that either b does not occur with a in a trace in N, or $(b, b) \in \lambda_\infty(N')$ which is not possible due to the acyclicness. Thus, in N' a always occurs with b, while in N they are alternatives. Hence, language inclusion cannot hold.

We can conclude that Thm. 7 holds, and Def. 10 is complete. □

No abstraction using the *k*-mapping In the most general case, when the characterisation does not pose any constraints, we have the same problems as mentioned earlier, e.g., that a simple inclusion does not work. Furthermore, we now have difficulties introduced by the combination of cycles and non-sequential activities, e.g., short loops in the α-mapping. Furthermore, the ∞-mapping has problems with cycles. For instance, see Fig. 7 where both models have the same ∞-mapping. However, the trace $ACBDE$ is included in the right model, but not in the left model. Obviously, with equivalent ∞-mapping, we should have language equivalence, i.e., any comparator denotes that the sets are included in each other.

Therefore, we consider the k-mapping. Again, using a naive approach, using the inclusion of relations does not work. Consider, for instance, the models in Fig. 8, where the relations in the right model are a subset of the relations in the left model.

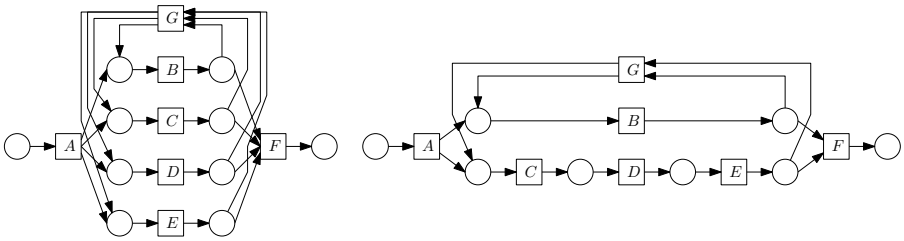

Fig. 9: For the k-mapping, we cannot find a value for k such that it yields that the right model is language included in the left model, but does not yield that the left model is language included in the right model.

Table 1: The framework for sound free-choice WF-nets

Abstraction	Characterisation	Pos/Neg	Complete
Trans. Syst.	S	Pos	✓
T	$\emptyset$	Neg	
α	$\emptyset$	Neg	
∞	A	Pos	✓

The problem with looking at k elements in advance is that this k is bounded by the shortest cycle. In other words, if k is larger than the shortest cycle then activities are considered in parallel when they are in sequence. Consider Fig. 9, from which it is clear that the right model is language-included in the left model. However, if we compute the relation between the activities, we need to have a relation between C and E (because present in right model, thus $k \geq 2$), but also $k < 2$ because $\langle ...EGC... \rangle$ is a proper part of a trace due to the cycle, while this would yield that C and E are in parallel in the right model, and thus that the left model is language included in the right model.

4.5 The complete framework

In Table 1, the total framework is listed. *Trans. Syst.* denotes that language inclusion can be efficiently computed on the transitions system. T is the negative abstraction only considering transitions. The negative abstraction using the α-mapping is listed under α. Finally, ∞ denotes the sound and complete positive abstraction using the ∞-mapping. Under characteristics: A denotes acyclic, and S denotes sequentiality. Pos/Neg denotes if it is a positive or negative approach, and complete denotes if the approach is complete.

5 Conclusion

We have presented a framework that supports the decision of language inclusion for sound unlabelled WF-nets in an efficient way. Apart from the framework,

we have provided a number of mappings, characterisations, and comparators to be used in the framework. If there is sequentiality present in both WF-nets, the language inclusion problem can be computed in polynomial time in terms of the underlying transition systems. If one of the WF-nets is non-sequential, then our framework may still efficiently decide language inclusion using a number of abstractions. The first abstraction uses the sets of transitions: If the set of transitions of the first net is not a subset of the set of transitions of the second, then the language of the first net cannot be included in the language of the second net. The second abstraction uses the directly-follows relation (also called the α-relation) between transitions: If the α-relation of the first net is not a subset of the α-relation of the second net, then the language of the first net cannot be included in the language of the second net. The third abstraction uses the eventually follows relation (also called the ∞-relation) between transitions: If the ∞-relation of the first free-choice net is a subset of the ∞-relation of the second free-choice net, and if some additional structural requirements hold, then the language of the first net is included in the language of the second net. For free-choice acyclic nets this third abstraction is complete. In other words, language inclusion can only hold if the ∞-relation of the first net is a subset of the ∞-relation of the second net, and if the additional structural requirements hold.

In case of free-choice unlabelled WF-nets, all relations can be computed polynomial in the size of the WF-net. On these relations, we mainly perform containment between sets. In the third abstraction, we search for an alternative activity which, naively, entails trying every activity and adds a polynomial factor. Since each of the steps is polynomial in the size of the WF-net, we can conclude that our computations can be done polynomial in the size of the WF-net.

The framework cannot efficiently decide on language inclusion in all cases. Especially for nets that are (1) non-sequential, (2) are cyclic or are not free-choice, and (3) for which language inclusion is known to hold, our current set of abstractions will not be able to provide an answer in polynomial time. Due to the fact that the comparators of the first two abstraction will yield true and the preconditions of the third abstraction are not met. However, we do not claim that our framework is complete. Yet, our framework can be easily extended based on the general pattern provided with new abstractions that may provide efficient answers for cases uncovered by the current abstractions: Given an abstraction, a characterisation, and a comparator, a positive (like the third abstraction) or negative (like the first two abstractions) abstraction can be formulated.

In some cases, our abstractions can easily be applied to general Petri nets. However, there does not exist, to our knowledge, a polynomial time algorithm to compute the abstractions on general nets. One can argue that computing the abstractions means analysing the behaviour in part or in full.

In the future, we plan to extend our framework with abstractions that also can deal with labelled WF-nets and with nets that are cyclic, for example by considering abstractions based on block-structured nets. For block-structured

nets (with a single entry and a single exit), the relations (like α and ∞) between transitions can often be derived efficiently from the block-structure.

Furthermore, we want to quantify the difference between two languages based on our framework, i.e., if language inclusion does not hold, what portion of the language in not included, and which part of the net is the main reason the language is not included. This can then be used as a quantification measure of inclusion between nets, and as a diagnostic result that can be used to align nets in such a way that language inclusion will hold.

References

1. Weidlich, M., Werf, J.M.E.M.v.d.: On Profiles and Footprints - Relational Semantics for Petri Nets. [22] 148–167
2. Glabbeek, R.J.v., Weijland, W.P.: Branching Time and Abstraction in Bisimulation Semantics. Journal of the ACM (JACM) (1996)
3. Kunze, M., Weidlich, M., Weske, M.: Behavioral Similarity - A Proper Metric. In Rinderle-Ma, S., Toumani, F., Wolf, K., eds.: BPM. Volume 6896 of Lecture Notes in Computer Science., Springer (2011) 166–181
4. Aalst, W.M.P.v.d., Adriansyah, A., Dongen, B.F.v.: Causal Nets: A Modeling Language Tailored towards Process Discovery. In Katoen, J.P., König, B., eds.: CONCUR. Volume 6901 of Lecture Notes in Computer Science., Springer (2011) 28–42
5. Schunselaar, D.M.M., Verbeek, H.M.W., Aalst, W.M.P.v.d., Reijers, H.A.: Creating Sound and Reversible Configurable Process Models Using CoSeNets. In Abramowicz, W., Kriksciuniene, D., Sakalauskas, V., eds.: BIS. Volume 117 of Lecture Notes in Business Information Processing., Springer (2012) 24–35
6. Clarke Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press (1999)
7. Abdulla, P.A., Chen, Y.F., Holík, L., Mayr, R., Vojnar, T.: When simulation meets antichains. In Esparza, J., Majumdar, R., eds.: TACAS. Volume 6015 of Lecture Notes in Computer Science., Springer (2010) 158–174
8. Dill, D.L., Hu, A.J., Wong-Toi, H.: Checking for language inclusion using simulation preorders. In Larsen, K.G., Skou, A., eds.: CAV. Volume 575 of Lecture Notes in Computer Science., Springer (1991) 255–265
9. Aalst, W.M.P.v.d., Weijters, A.J.M.M., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering **16**(9) (2004) 1128–1142
10. Badouel, E.: On the α-reconstructibility of workflow nets. [22] 128–147
11. Wen, L., van der Aalst, W.M.P., Wang, J., Sun, J.: Mining process models with non-free-choice constructs. Data Mining and Knowledge Discovery **15**(2) (2007) 145–180
12. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Causal Behavioural Profiles - Efficient Computation, Applications, and Evaluation. Fundamenta Informaticae **113**(3-4) (2011) 399–435
13. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., Weske, M.: Process compliance analysis based on behavioural profiles. Information Systems **36**(7) (2011) 1009–1025

14. Smirnov, S., Weidlich, M., Mendling, J.: Business process model abstraction based on behavioral profiles. In Maglio, P.P., Weske, M., Yang, J., Fantinato, M., eds.: ICSOC. Volume 6470 of Lecture Notes in Computer Science. (2010) 1–16
15. Dongen, B.F.v., Mendling, J., Aalst, W.M.P.v.d.: Structural Patterns for Soundness of Business Process Models. In: EDOC, IEEE Computer Society (2006) 116–128
16. Mendling, J., van Dongen, B.F., van der Aalst, W.M.P.: On the degree of behavioral similarity between business process models. In Nüttgens, M., Rump, F.J., Gadatsch, A., eds.: EPK. Volume 303 of CEUR Workshop Proceedings., CEUR-WS.org (2007) 39–58
17. Solé, M., Carmona, J.: A High-Level Strategy for C-net Discovery. In Brandt, J., Heljanko, K., eds.: ACSD, IEEE Computer Society (2012) 102–111
18. Desel, J.: Validation of process models by construction of process nets. In Aalst, W.M.P.v.d., Desel, J., Oberweis, A., eds.: BPM. Volume 1806 of Lecture Notes in Computer Science., Springer (2000) 110–128
19. Lautenbach, K.: Liveness in Petri Nets. Internal Report of the Gesellschaft für Mathematik und Datenverarbeitung. Bonn, Germany, ISF/75-02-1 (1975)
20. Desel, J., Esparza, J.: Free Choice Petri Nets (Cambridge Tracts in Theoretical Computer Science). Cambridge University Press (1995)
21. Aalst, W.M.P.v.d.: Verification of Workflow Nets. In Azéma, P., Balbo, G., eds.: ICATPN. Volume 1248 of Lecture Notes in Computer Science., Springer (1997) 407–426
22. Haddad, S., Pomello, L., eds.: Application and Theory of Petri Nets - 33rd International Conference, PETRI NETS 2012, Hamburg, Germany, June 25-29, 2012. Proceedings. In Haddad, S., Pomello, L., eds.: Petri Nets. Volume 7347 of Lecture Notes in Computer Science., Springer (2012)