Incremental Computation of Synthesis Rules for Free-Choice Petri nets

P.M. Dixit[1], H.M.W. Verbeek[1], and W.M.P. van der Aalst[2]

[1] *Eindhoven University of Technology, Eindhoven, The Netherlands*
[2] *Rheinisch-Westflische Technische Hochschule, (RWTH) Aachen, Germany*
`{p.m.dixit,h.m.w.verbeek}@tue.nl`
`wvdaalst@pads.rwth-aachen.de`

Abstract. In this paper, we propose a novel approach that calculates all the possible applications of synthesis rules, for well-formed free-choice Petri nets [8], in a speedy way to enable an interactive editing system. The proposed approach uses a so-called *incremental synthesis structure*, which can be used to extract all the synthesis rules, corresponding to a given net. Furthermore, this structure is updated incrementally, i.e. after usage of a synthesis rule, to obtain the incremental synthesis structure of the newly synthesized net. We prove that the proposed approach is correct and complete in order to synthesize any well-formed free-choice Petri net, starting with an initial well-formed atomic net and the corresponding incremental synthesis structure. A variant of the proposed approach has been implemented that allows interactive modeling (discovery) of sound business processes (from event logs). Experimental results show that the proposed approach is fast, and outperforms the baseline, and hence is well-suited for enabling interactive synthesis of very large nets.

Keywords: free-choice Petri nets, interactive system, incremental synthesis rules computation

1 Introduction

Petri nets serve as an effective tool for modeling the control flow of asynchronous concurrent systems. In order to be useful, in application areas such as business process management, it is imperative that the Petri nets representing the business processes adhere to certain properties, e.g. soundness [1]. In this paper, we propose an approach that allows interactive editing of free-choice Petri nets from smaller nets by preserving certain properties (such as well-formedness). We show that the proposed approach is fast, and thus has negligible waiting times and thereby well-suited for enabling an interactive editing system. Moreover, using the guarantees from [8], we show that the approach can be used to synthesize any well-formed free-choice Petri net. A sub-class of such nets can also be used to model sound business processes, and is also applicable in the field of process mining.

Petri net synthesis techniques allow synthesizing bigger Petri nets from smaller Petri nets. In many cases, the synthesized net guarantees certain structural and behavioral properties, depending on the net it was synthesized from. The technique of synthesizing Petri nets from smaller nets is not new, and has been well-researched for over two decades [2,12,7]. Synthesis rules are basically reverse applications of *reduction rules*, which are well researched in the literature [5,4,14,13,3] for a variety of different applications [11,16]. The work in [6] provides a way of interactively synthesizing Petri nets using a knitting technique. However, the completeness and correctness of these rules is not guaranteed. In our case, we refer to the synthesis techniques that allow Petri net expansion, one transition and/or place at a time. We use such rules to enable a user interactive well-formed free-choice Petri net editing system. Compared to the approaches from the literature, our main focus is not to suggest new synthesis rules, but to develop an approach that allows speedy computation of all possible applications of synthesis rules to enable an interactive system for editing/modeling of well-formed free-choice Petri nets.

In our approach, we use the synthesis rule kit from [8], which are mainly derived from [5], that allow synthesis of any well-formed free-choice Petri net, starting with an initial well-formed atomic net. The synthesis rule kit consists of three rules which allow (i) addition of a transition, (ii) addition of a place, or (iii) addition of a transition and a place to a Petri net. Thus these rules could be used as the building blocks for enabling interactive free-choice Petri net editing/modeling, which guarantee well-formedness. To enable interactive modeling, it is ideal to pre-populate all the possible applications of synthesis rules for any given net. However, computing all the synthesis operations in a brute-force way for any given net can be computationally expensive and hence very slow, especially for the first two rules, which require solving a system of linear equations to check the applicability of the rule. This is clearly not feasible as the size of the net grows, as the number of applications of synthesis rules would grow too.

In this paper, we address the issue of computing all the applications of synthesis rules for a given well-formed free choice net in a speedy way to allow interactive editing. We propose an approach based on an *incremental synthesis structure*, that can be used to calculate all the possible applications of synthesis rules, for a given well-formed net. This meta-structure is updated after application of any rule, to contain information to extract all the possible applications of synthesis rules corresponding to the newly synthesized net. We show that by starting with a correct and complete incremental synthesis structure for the initial net, we can calculate the incremental synthesis structure of any synthesized net, and thereby calculate all the possible applications of synthesis rules in an incremental way.

The remainder of the paper is structured as follows. We review the preliminaries in Section 2. In Section 3 and Section 4 we discuss the approach for calculating the synthesis rules in an incremental fashion, as well as discuss the correctness and completeness of the proposed approach. In section 5, we briefly discuss the implementation and application of the approach to business processes. We evaluate the proposed approach in Section 6, followed by the conclusions and future research directions in Section 7.

2 Preliminaries

This section introduces some basic definitions and background of the concepts used in this paper.

A bag over some set S is a function from S to the natural numbers that assigns only a finite number of elements from S a positive value. For a bag B over set S and $s \in S$, $B(s)$ denotes the number of occurrences of s in B, often called the cardinality of s in B. Note that a finite *set* of elements of S is also a bag over S, namely the function yielding 1 for every element in the set and 0 otherwise. The set of all bags over set S is denoted $\mathcal{B}(S)$. We use brackets to explicitly enumerate a bag and superscripts to denote cardinalities. For example $[a^2, b^3, c]$ denotes a bag in which the elements a, b and c are contained 2, 3 and 1 times resp. Bag B is a subbag of bag B', denoted $B \leq B'$, iff, for all $s \in S$, $B(s) \leq B'(s)$. The standard operators can be extended to bag, e.g., $[a^2] + [a^2, b^3, c] = [a^4, b^3, c]$. Furthermore, we also allow addition or subtraction of bags with sets, e.g., $[a^2, b^3, c] - \{a\} = [a, b^3, c]$.

A relation $R \subseteq X \times Y$ is a set of pairs, where $\pi_1(R) = \{x \mid (x, y) \in R\}$ denotes the domain of R, $\pi_2(R) = \{y \mid (x, y) \in R\}$ denotes the range of R, and $\omega(R) = \pi_1(R) \cup \pi_2(R)$ denotes the elements of R. For example, $\omega(\{(a, b), (b, c)\}) = \{a, b, c\}$. $f : X \not\rightarrow Y$ denotes a partial function with domain $dom(f) \subseteq X$ and range $rng(f) = \{f(x) \mid x \in X\} \subseteq Y$. $f : X \rightarrow Y$ denotes a total function, i.e., $dom(f) = X$. Given a finite set $A = \{a_1, \ldots, a_k\}$, every mapping $\mathbf{X}$ from A to $\mathbb{Q}$, denoted $\mathbf{X} : A \rightarrow \mathbb{Q}$ can be represented by the vector $(\mathbf{X}(a_1) \ldots X(a_k))$. We do not distinguish between the mapping $\mathbf{X}$ and the vector $(\mathbf{X}(a_1) \ldots \mathbf{X}(a_k))$. $\mathbf{X} \cdot \mathbf{Y}$ denotes the scalar product of two vectors. Similarly, if $\mathbf{C}$ is a matrix, then $\mathbf{X} \cdot \mathbf{C}$ and $\mathbf{C} \cdot \mathbf{X}$ denote the left and right products of $\mathbf{X}$ and $\mathbf{C}$. We do not use different symbols for row and column vectors. For e.g., if we write $\mathbf{X} = (\mathbf{X}(a_1) \ldots \mathbf{X}(a_k))$, it serves as a column vector in $\mathbf{C} \cdot \mathbf{X}$.

Definition 1 (Petri net). *A Petri net N is a tuple (P, T, F) such that:*

- *P is a finite set of places,*
- *T is a finite set of transitions such that $P \cap T = \emptyset$, and*
- *$F \subseteq (P \times T) \cup (T \times P)$ is the set of arcs, also called the flow relation.*

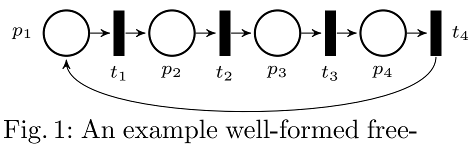


Fig. 1: An example well-formed free-choice Petri net (N_3).

For a Petri net $N = (P, T, F)$, and a node $n \in P \cup T$, the *preset* of n in N, denoted $\overset{N}{\bullet} n$, is the set of all nodes that have an arc to n, that is, $\overset{N}{\bullet} n = \{n' | (n', n) \in F\}$. Likewise, the *postset* of n in N, denoted $n \overset{N}{\bullet}$, is the set of all nodes that have an arc from n, that is, $n \overset{N}{\bullet} = \{n' | (n, n') \in F\}$. In case the net N is clear from the context, we typically omit it from these notations and use $\bullet n$ instead of $\overset{N}{\bullet} n$ etc. A Petri net $N = (P, T, F)$ is called a free-choice (or FC) net if for $r, u \in T$, it holds that $\bullet r \cap \bullet u = \emptyset$ or $\bullet r = \bullet u$. Figure 1 is also a FC net. The nodes of a Petri net can be used to populate a so-called incidence matrix.

Definition 2 (Incidence matrix). *Let $N = (P, T, F)$ be a Petri net. The incidence matrix $\mathbf{N} : (P \times T) \to \{1, 0, -1\}$ of N is defined by*

$$\mathbf{N}(s, r) = \begin{cases} -1, \textit{ if } (s, r) \in F \textit{ and } (r, s) \notin F; \\ \;\;1, \textit{ if } (s, r) \notin F \textit{ and } (r, s) \in F; \\ \;\;0, \textit{ otherwise.} \end{cases}$$

Table 1 shows the incidence matrix corresponding to the Petri net from Figure 1. The column vector of $\mathbf{N}$ is associated to a transition t is denoted by $\mathbf{t}$ and given as $\mathbf{t} : P \to \{-1, 0, 1\}$. Similarly, the row vector associated to a place p is $\mathbf{p} : T \to \{-1, 0, 1\}$. $\mathbf{t}(s)$ denotes the value corresponding to the place s, e.g., $\mathbf{t_2}(p_2) = -1$.

Table 1: Incidence matrix $\mathbf{N_3}$ of Figure 1.

	t_1	t_2	t_3	t_4
p_1	-1	0	0	1
p_2	1	-1	0	0
p_3	0	1	-1	0
p_4	0	0	1	-1

A marking M of a Petri net $N = (P, T, F)$ is a bag over the places of N, that is, $M \in \mathcal{B}(P)$. A marking M is typically represented as a collection $M(s)$ of *tokens* for every place s. Removing a token from s then corresponds to removing one occurrence of s from M, and adding a token to s corresponds to adding one occurrence of s to M.

Let $N = (P, T, F)$ be a Petri net, let M be a marking of N, and let $r \in T$ be a transition of N. Transition r is *enabled* in M, denoted $M \xrightarrow{r}$, if and only if $\bullet r \leq M$. An enabled transition may *fire*, which leads to a new marking M', where $M' = M - \bullet r + r \bullet$, that is, in the new marking, a token is first removed from every place in the preset of r and a token is then added to every place in the postset of r. This firing is denoted as $M \xrightarrow{r} M'$.

Let $\sigma = \langle t_1, t_2 ..., t_n \rangle \in T^*$ be a sequence of transitions. $M \xrightarrow{\sigma} M'$ denotes that there is a set of markings $M_0, M_1, ..., M_n$ such that $M_0 = M$, $M_n = M'$, and $M_i \xrightarrow{t_{i+1}} M_{i+1}$ for $0 \leq i < n$. A marking M' is *reachable* from M if there exists a σ such that $M \xrightarrow{\sigma} M'$.

We use $R(N, M)$ to denote the set of markings reachable from marking M in Petri net N, i.e. $R(N, M) = \{M' | M' \in \mathcal{B}(P) \wedge M \xrightarrow{\sigma} M' \text{ for some } \sigma \in T^* \}$.

A Petri net $N = (P, T, F)$ is called *strongly connected* if and only if $\forall_{n, n' \in P \cup T} (n, n') \in F^*$, where F^* is the reflexive transitive closure of F. Let $N' = (P', T', F')$ such that $P' \subseteq P$ and $T' \subseteq T$ and $F' = F \cap ((P' \times T') \cup (T' \times P'))$. The net N' is called an *S-net* if and only if for all $r \in T'$ it holds that $| \overset{N'}{\bullet} r| = 1 = |r \overset{N'}{\bullet} |$, that is, every transition has a single place in its preset and a single place in its postset. The S-net N' is called an *S-component* of net N if and only if N' is strongly connected and for all $s \in P'$ it holds that $\overset{N}{\bullet} s \cup s \overset{N}{\bullet} \subseteq T'$. The net N is called *S-coverable* if and only if for every place $s \in P$ there exists an S-component that contains s. Similarly, we can define *T-components*.

A transition r belonging to a Petri net N with an initial marking M is called *live*, if and only if for every $M' \in R(N, M)$ there exists an $M'' \in R(N, M')$ that enables the transition r. The net N is called *live* in M if and only if all its transition are live in M. Furthermore, N is *deadlock-free* in M if and only if every reachable marking enables at least one transition.

A place s in a Petri net N with the initial marking M is called *k-bounded* (where k is a natural number), if and only if the place s never holds more than k tokens in any reachable marking, i.e. $\forall_{M' \in R(N,M)} : M'(s) \leq k$. A place p is called bounded in M if and only if it is k-bounded in M for some k. The net N is called *bounded* in M if and only if all its places are bounded in M, it is called *unbounded* in M otherwise. A Petri net net N is called well-formed, if and only if there exists a marking M, such that N is live and bounded in M [8].

2.1 Synthesis Rules

The starting net used in our approach is a strongly connected atomic net containing one place and one transition, as shown in Figure 2. The synthesis rules described in [8] are valid for *all* well-formed FC nets. We first discuss the two linearly dependent synthesis rules. A linearly dependent rule allows for an introduction of a new place *or* a new transition in a net. We begin by introducing linearly dependency, which is then used to define the linear dependency rules.

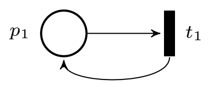


Fig. 2: Initial atomic net N_0.

Definition 3 (Linear dependency). *Let* $\mathbf{M}$ *be an* $m \times n$ *matrix and let* $\mathbb{Q}$ *be the set of all rational numbers. An* n*-dimensional vector* $\mathbf{A}$ *is linearly dependent on the rows of* $\mathbf{M}$ *iff there exists an* m*-dimensional vector* $\lambda : \{1 \cdots m\} \rightarrow \mathbb{Q}$*, s.t.,* $\lambda \cdot \mathbf{M} = \mathbf{A}$*. Similarly, an* m*-dimensional vector* $\mathbf{B}$ *is linearly dependent on the columns of* $\mathbf{M}$ *iff there exists an* n*-dimensional vector* $\mu : \{1 \cdots n\} \rightarrow \mathbb{Q}$*, s.t.* $\mathbf{M} \cdot \mu = \mathbf{B}$.

Following the definition of linear dependency, we now discuss the two linearly dependent rules from [8].

Definition 4 (Linearly Dependent Place Rule ψ_P (derived from [8])). *Let* $N = (P, T, F)$ *and* $N' = (P', T', F')$ *be two FC nets.* N' *is synthesized from* N*, i.e.* $(N, N') \in \psi_P$ *if and only if:*

1. $T' = T$
2. $P' \setminus P = \{p\}$
3. $F' = F \cup \tilde{F}$*, where* $\tilde{F} \subseteq ((\{p\} \times T) \cup (T \times \{p\}))$
4. $\mathbf{p}$ *is linearly dependent on the rows of* $\mathbf{N}$
5. $\overset{N'}{\bullet} p \cup p \overset{N'}{\bullet} \neq \emptyset$

Figure 3a shows the application of ψ_P rule on the net N_3 from Figure 1. It can easily be seen that $\mathbf{p_5} = (1, 0, 0, -1)$, where $\mathbf{p_5}(t_1) = 1$, $\mathbf{p_5}(t_2)$=0 and so on, is linearly dependent on the rows of $\mathbf{N_3}$ (Table 1), such that $\mathbf{p_5} = \mathbf{p_2} + \mathbf{p_3} + \mathbf{p_4}$. Similarly, we can define the linearly dependent transition rule as follows:

Definition 5 (Linearly Dependent Transition Rule ψ_T (derived from [8])). *Let* $N = (P, T, F)$ *and* $N' = (P', T', F')$ *be two FC nets.* N' *is synthesized from* N*, i.e.* $(N, N') \in \psi_T$ *if and only if:*

1. $P' = P$
2. $T' \setminus T = \{t\}$
3. $F' = F \cup \tilde{F}$, *where* $\tilde{F} \subseteq ((P \times \{t\}) \cup (\{t\} \times P))$
4. $\mathbf{t}$ *is linearly dependent on the columns of* $\mathbf{N}$
5. $\overset{N'}{\bullet} t \cup t \overset{N'}{\bullet} \neq \emptyset$

Figure 3b shows the application of ψ_T rule on the net N_4 from Figure 3a. Having defined the linear dependency rules, we now define the final rule, i.e. the abstraction rule, which allows expansion of a net by adding a new place *and* a new transition. The abstraction rule can be formally defined as:

Definition 6 (Abstraction Rule ψ_A [8]). *Let $N = (P, T, F)$ and $N' = (P', T', F')$ be two FC nets. N' is synthesized from N, i.e. $(N, N') \in \psi_A$ if and only if there exists a non-empty set of transitions $R \subseteq T$ and a non-empty set of places $S \subseteq P$ such that:*

1. $P' \setminus P = \{p\}$
2. $T' \setminus T = \{t\}$
3. $R \times S \subseteq F \cap (T \times P)$
4. $F' = (F \setminus (R \times S)) \cup ((R \times \{p\}) \cup (\{p\} \times \{t\}) \cup (\{t\} \times S))$

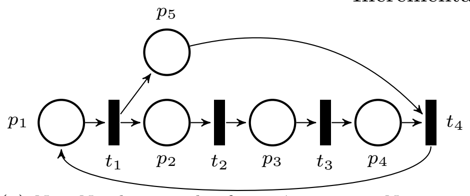
We can get to the net N_3 from Figure 1, by using three applications of the ψ_A rule on the initial net N_0 from Figure 2. An important property of these synthesis rules is that they preserve well-formedness for FC nets [8]. That is, if $(N, N') \in \psi_A \cup \psi_T \cup \psi_P$, then N' is well-formed iff N is well-formed. Furthermore, it has been shown that these rules are complete to synthesize any well-formed FC net, starting with the initial atomic net of Figure 2.

3 Synthesis Space

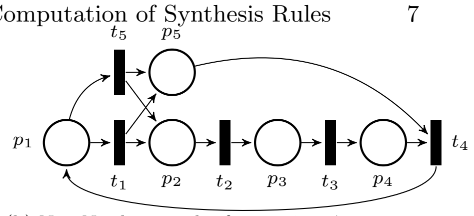
In this section, we use the synthesis rules for FC nets from [8], in order to discuss the concept of synthesis space. The synthesis space contains all the applications of ψ_A, ψ_P and ψ_T rules on a well-formed FC net by adding a new transition and/or a new place. Formally, the synthesis space is defined as:

Definition 7 (Synthesis Space SS). *Let $N = (P, T, F)$ be a well-formed FC net, and let F_N be the universe of well-formed FC nets. The synthesis space $SS(N) = SS_A(N) \cup SS_P(N) \cup SS_T(N)$, where: $SS_A(N) = \{N' = (P', T', F') \in F_N \mid (N, N') \in \psi_A \wedge \{t\} = T' \setminus T \wedge \{p\} = P' \setminus P\}$, $SS_T(N) = \{N' = (P', T', F') \in F_N \mid (N, N') \in \psi_T \wedge \{t\} = T' \setminus T\}$, and $SS_P(N) = \{N' = (P', T', F') \in F_N \mid (N, N') \in \psi_P \wedge \{p\} = P' \setminus P\}$.*

$SS_A(N)$, $SS_P(N)$ and $SS_T(N)$ are all disjoint for a well-formed FC net N, because ψ_A results in addition of a new transition t *and* a new place p, whereas ψ_P or ψ_T results in addition of a new place p *or* a new transition t resp. In essence, the newly added place p or transition t can be renamed as any place s or r resp. Thus, using [8], we argue that the synthesis space contains all the



(a) Net N_4 that results from ψ_P on net N_3, see Figure 1.



(b) Net N_5 that results from using ψ_T on net N_4.

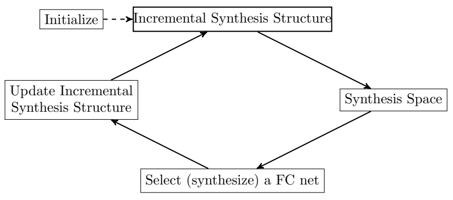
Fig. 3: Applications of linear dependency synthesis rules.

possible applications of synthesis rules, for a given well-formed FC net N. After N' is selected from $SS(N)$, the synthesis space $SS(N')$ corresponding to N' needs to be recomputed. Calculating the synthesis space would thus require exploring all the possible ways in which ψ_A, ψ_P or ψ_T can be applied.

The two linearly dependent rules require solving a system of linear equations in order to check their applicability. Contrary to this, the ψ_A rule can be checked *locally*, i.e. by checking the existence of arcs between pre transitions R and post places S. Hence, it is trivial to calculate all the possible applications of ψ_A rule, however for ψ_P and ψ_T rules this is not the case. In order to overcome this, we introduce an incremental way of calculating the synthesis space for linear dependency rules, discussed in the following section.

4 Incremental Synthesis Structure

As recomputing the synthesis space from scratch is computationally expensive especially while calculating the linear dependency rules, we introduce an intermediary structure, called the *incremental synthesis structure (ISS)*, which is used to extract the synthesis space for non-local rules. After choosing any net from the synthesis space, the incremental synthesis structure is updated to contain information required to extract the synthesis space corresponding to the new net.



Initialize
Incremental Synthesis Structure
Update Incremental Synthesis Structure
Synthesis Space
Select (synthesize) a FC net

Fig. 4: The incremental synthesis structure (ISS) corresponding to the initial net is calculated in a brute force way. ISS contains all the possible linear dependencies that could be added to a net resulting in well-formed FC nets, among others. The linear dependencies that result in well-formed FC nets are extracted to populate the synthesis space. Upon selecting a net from the synthesis space, the ISS is updated corresponding to the net selected.

Figure 4 provides an overview of our approach. As we cater only to the applications of linear dependency rules (ψ_T or ψ_P), the incremental synthesis structure contains only column and row vectors, from which candidates can be derived for linearly dependent transitions or places. A (non-existing) candidate transition can be identified by two sets of (existing) places: the set P_I of input places for the candidate transition and the set P_O of output places for the candidate transition. Likewise, a candidate place can be identified by a set T_I of input transitions and a set T_O of output transitions. Using these nodes as input and output, a new place (transition) can be added to the net. We now define, how we can derive these sets of input and output nodes from a vector.

Definition 8 (Input/output sets from a vector). *Let $N = (P, T, F)$ be a FC Petri net, and let $\mathbf{tt} : P \rightarrow \{-1, 0, 1\}$ be a $|P|-$dimensional vector. Let $P^i(\mathbf{tt}) = \{s \in P \mid \mathbf{tt}(s) = i\}$. Then the set of input/output places for $\mathbf{tt}$, denoted as $\mathfrak{N}(\mathbf{tt})$, is defined as follows:*
$\mathfrak{N}(\mathbf{tt}) = \{(P_I, P_O) \mid P_I = P^{-1}(\mathbf{tt}) \cup X \wedge P_O = P^1(\mathbf{tt}) \cup X \wedge X \subseteq P^0(\mathbf{tt}) \wedge P_I \cup P_O \neq \emptyset\}$.

Similarly, let $\mathbf{pp} : T \rightarrow \{-1, 0, 1\}$ be a $|T|$-dimensional vector. Let $T^i(\mathbf{pp}) = \{r \in T \mid \mathbf{pp}(r) = i\}$. Then the sets of input/output transitions for $\mathbf{pp}$, denoted as $\mathfrak{N}(\mathbf{pp})$, is defined as follows:
$\mathfrak{N}(\mathbf{pp}) = \{(T_I, T_O) \mid T_I = T^1(\mathbf{pp}) \cup X \wedge T_O = T^{-1}(\mathbf{pp}) \cup X \wedge X \subseteq T^0(\mathbf{pp}) \wedge T_I \cup T_O \neq \emptyset\}$.

A single vector can result in multiple pairs of input and output nodes. For example, consider a vector $\mathbf{tt_1} = (1, -1, 0, 0, -1)$ corresponding to the net N_4 from Figure 3a, where $\mathbf{tt_1}(p_1) = 1$, $\mathbf{tt_1}(p_2)=-1$ and so on. Then $\mathfrak{N}(\mathbf{tt_1}) = \{(\{p_2, p_3, p_5\}, \{p_1, p_3\}), (\{p_2, p_4, p_5\}, \{p_1, p_4\}), (\{p_2, p_3, p_4, p_5\}, \{p_1, p_3, p_4\}), (\{p_2, p_5\}, \{p_1\})\}$. Having linked vectors to possible input and output nodes, we now define a so-called *valid ISS vector.*

Definition 9 (Valid ISS vector). *Let $N = (P, T, F)$ be a well-formed FC net. A $|P|$-dimensional vector $\mathbf{tt} : P \rightarrow \mathbb{Z}$ is called a valid ISS column vector in N iff*

1. $\mathbf{tt} : P \rightarrow \{-1, 0, 1\}$ *and* $\mathbf{tt}$ *is linearly dependent on the columns of* $\mathbf{N}$,
2. $\exists (P_I, P_O) \in \mathfrak{N}(\mathbf{tt})$, *such that either*
   (a) $\exists_{r \in T} \bullet r = P_I$, *or*
   (b) $\exists_{R \subseteq T} R \times P_I \subseteq F \wedge R \neq \emptyset$

Similarly, a $|T|$-dimensional vector $\mathbf{pp} : T \rightarrow \mathbb{Z}$ is called a valid ISS row vector in N, iff

1. $\mathbf{pp} : T \rightarrow \{-1, 0, 1\}$ *and* $\mathbf{pp}$ *is linearly dependent on the rows of* $\mathbf{N}$,
2. $\exists (T_I, T_O) \in \mathfrak{N}(\mathbf{pp})$, *such that either*
   (a) $\exists_{s \in P} s \bullet = T_O$, *or*
   (b) $\exists_{S \subseteq P} T_O \times S \subseteq F \wedge S \neq \emptyset$

Again, consider the vector $\mathbf{tt_1} = (1, -1, 0, 0, -1)$ corresponding to the net N_5 from Figure 3b, where $\mathbf{tt_1}(p_1) = 1$, $\mathbf{tt_1}(p_2)= -1$ and so on. This vector is linearly dependent on the columns of the incidence matrix $\mathbf{N_4}$ (see Table 2), as $\mathbf{tt_1} = \mathbf{t_2} + \mathbf{t_3} + \mathbf{t_4}$. Hence $\mathbf{tt_1}$ satisfies condition (1) from Definition 9.

Furthermore we have $\mathfrak{N}(\mathbf{tt_1}) = \{(\{p_2, p_3, p_5\}, \{p_1, p_3\}), (\{p_2, p_4, p_5\}, \{p_1, p_4\}), (\{p_2, p_3, p_4, p_5\}, \{p_1, p_3, p_4\}), (\{p_2, p_5\}, \{p_1\})\}$. The first three pairs do not satisfy either of the conditions, i.e. (2)(a) or (2)(b), from Definition 9. However, the fourth pair satisfies condition (2)(b), which states that $\exists_{R\in T} R \times P_I \subseteq F$, in this particular case, $R = \{t_1\}$, and $P_I = \{p_2, p_5\}$. Hence, as $\mathbf{tt_1}$ satisfies condition (1) and (2)(b) of Definition 9, it is a valid ISS column vector. Having defined *valid ISS vectors*, we now define the incremental synthesis structure, as follows.

Table 2: Incidence matrix $\mathbf{N_4}$ of Figure 3a.

	t_1	t_2	t_3	t_4
p_1	-1	0	0	1
p_2	1	-1	0	0
p_3	0	1	-1	0
p_4	0	0	1	-1
p_5	1	0	0	-1

Definition 10 (Incremental Synthesis Structure (ISS)). *Let $N = (P, T, F)$ be a well-formed FC net. The incremental synthesis structure is a tuple $ISS(N) = (\mathbf{TT}, \mathbf{PP})$, where:*

1. $\mathbf{TT}$ *is a non-empty set of vectors, s.t.* $\mathbf{tt} \in \mathbf{TT}$ *iff* $\mathbf{tt}$ *is a valid ISS column vector in N.*
2. $\mathbf{PP}$ *is a non-empty set of vectors, s.t.* $\mathbf{pp} \in \mathbf{PP}$ *iff* $\mathbf{pp}$ *is a valid ISS row vector in N.*

If a valid ISS vector satisfies conditions (1) and (2)(a) in Definition 9, then it can be used to apply ψ_T (or ψ_P), as shown in Lemma 1.

Lemma 1. *Let $N = (P, T, F)$ be a well-formed FC net, let* $\mathbf{tt}$ *be a valid ISS column vector, let $(P_I, P_O) \in \mathfrak{N}(\mathbf{tt})$, and let $r \in T$ such that $\overset{N}{\bullet} r = P_I$. Then, we can use ψ_T to add a new transition t in N to get $N' = (P', T', F')$, i.e. $(N, N') \in \psi_T$, where $\{t\} = T' \setminus T$ and $\overset{N'}{\bullet} t = P_I \wedge t \overset{N'}{\bullet} = P_O$.*

Proof. The fact that $\exists_{r\in T} \overset{N}{\bullet} r = P_I$ ensures that the net N remains a FC net after adding t. The fact that $\mathbf{tt}$ is a valid ISS column vector ensures that $P_I \cup P_O \neq \emptyset$, and that $\mathbf{t}$ ($= \mathbf{tt}$) is linearly dependent on the columns of $\mathbf{N}$. Hence, it follows from Definition 5 that a vector following conditions (1) and (2)(a) can be used to apply ψ_T. A similar argument can be made for ψ_P.

Consider a vector $\mathbf{tt_2} = (-1, 1, 0, 0, 1)$ corresponding to the net from Figure 3a, such that $\mathbf{tt_2}(p_1) =$ -1, $\mathbf{tt_2}(p_2) = 1$, $\mathbf{tt_2}(p_3) = 0$ and so on. Clearly, $\mathbf{tt_2}$ satisfies condition (1) of Definition 9, as $\mathbf{tt_2} = \mathbf{t_1}$ (see Table 2). We can obtain a pair $(\{p_1\}, \{p_2, p_5\}) \in \mathfrak{N}(\mathbf{tt_2})$. This pair is valid according to condition (2)(a) of Definition 9, as $\bullet t_1 = \{p_1\}$. Hence according to Lemma 1, we can use $\mathbf{tt_2}$ to add a new linearly dependent transition with $\{p_1\}$ as the input and $\{p_2, p_4\}$ as the output. Figure 3b is indeed a demonstration of adding this transition. In contrast, the second condition of valid ISS vector is in place to deduce linear dependencies that may come into effect in future, as will become evident in sections to follow.

Lemma 2. *Synthesis space for linear dependency rules can be extracted using* $\mathbf{TT}$ *and* $\mathbf{PP}$.

Proof. For every $\mathbf{tt} \in \mathbf{TT}$, we can extract a set of pair of places $(P_I, P_O) \in \mathfrak{N}(\mathbf{tt})$. From Lemma 1, we know that if a valid ISS vector is valid according to conditions (1) and (2)(a), then we can use ψ_T. If ISS is complete, i.e. contains all the possible valid ISS vectors, then it is clear that we can extract all the possible pairs that result in valid constructions of well-formed nets by adding transitions using Lemma 1. By definition, $\mathbf{TT}$ (and $\mathbf{PP}$) is complete and contains all the valid ISS vectors. It should be noted that the newly added node can be named as desired. Therefore, using the definition of Definition 5 (and Definition 4) it can be said that $\mathbf{TT}$ (and $\mathbf{PP}$) contains complete information to extract all the pairs which could be used to apply ψ_T (and ψ_P).

The synthesis space for ψ_P and ψ_T rules can be extracted using the incremental synthesis structure. It should be noted that a single valid ISS vector can result in multiple valid candidate nodes in the net, and hence can result in multiple applications of a rule.

We argue that ISS can be incrementally updated which can then be used to obtain the synthesis space. The ISS corresponding to the initial atomic net N_0 from Figure 2 is: $\mathbf{TT}_0 = \{(0)\}$, and $\mathbf{PP}_0 = \{(0)\}$.

It should be noted that $\mathbf{TT}_0$ and $\mathbf{PP}_0$ contain symmetrical elements. Since there is only one place (transition), the mapping between vector and places (transitions) of $\mathbf{TT}_0$ ($\mathbf{PP}_0$)is evident. In the following sections we discuss the changes to ISS after the selection of a net (application of a rule) from the synthesis space. Furthermore, we prove that the changes made are necessary and sufficient to derive the new ISS corresponding to the new net. We know that the synthesis of a net leads to addition of a transition and/or place. Since the incidence matrix is extended after the application of a synthesis rule, we need to extend the corresponding vectors from the ISS with one dimension corresponding to the newly added node. In order to do so, we first define extending a vector as follows.

Definition 11 (Vector extension $\mathbf{v}^{\frown e \mapsto k}$). *Let $\mathbf{v} : A \to \mathbb{Q}$ be an n-dimensional vector, such that $e \notin A \wedge |A| = n$. Then $\mathbf{v}^{\frown e \mapsto k} : A \cup \{e\} \to \mathbb{Q}$ is an $n+1$-dimensional vector such that:*

$$\mathbf{v}^{\frown e \mapsto k}(a) = \begin{cases} \mathbf{v}(a), \text{ if } a \in A; \\ \quad k, \text{ otherwise (i.e. } a = e) \end{cases}$$

We now discuss the updates to the ISS, after application of each type of synthesis rule. We argue that starting with $ISS(N_0)$, we can incrementally update the ISS corresponding to any synthesized net. Furthermore, by maintaining the ISS corresponding to each synthesized net, we can always deduce the synthesis space using Lemma 2.

4.1 ISS updates after application of ψ_P and ψ_T

In this section, we discuss the updates in the incremental synthesis structure after the usage of linear dependency rules.

Theorem 1 (Extracting *ISS* after ψ_P). *Let $N = (P, T, F)$ and $N' = (P', T', F')$ be two well-formed FC nets, where $(N, N') \in \psi_P$ and $\{p\} = P' \setminus P$. Let λ be such that $\lambda \cdot \mathbf{N} = \mathbf{p}$. As $(N, N') \in \psi_P$, such a λ exists. Let $ISS(N) = (\mathbf{TT}, \mathbf{PP})$ be the incremental synthesis structure of N. Then we can extract $\mathbf{PP'}$ and $\mathbf{TT'}$:*

- $\mathbf{PP'} = \mathbf{PP}$
- $\mathbf{TT'} = \bigcup_{\mathbf{tt} \in \mathbf{TT}} f_t(\mathbf{tt})$ *where,*
  $f_t(\mathbf{tt}) = \begin{cases} \{\mathbf{tt}^{\frown p \mapsto \lambda \cdot \mathbf{tt}}\}, \textit{ if } \mathbf{tt}^{\frown p \mapsto \lambda \cdot \mathbf{tt}} \textit{ is a valid ISS vector in } N' \\ \emptyset, \textit{ otherwise} \end{cases}$

s.t. $ISS(N') = (\mathbf{TT'}, \mathbf{PP'})$ is the incremental synthesis structure of N'.

Proof. We show that Theorem 1 is correct and complete separately for $\mathbf{PP'}$ and $\mathbf{TT'}$.

$\mathbf{PP'}$ Since $\mathbf{p}$ is linearly dependent on the rows of $\mathbf{N}$, the ranks of $\mathbf{N}$ and $\mathbf{N'}$ are the same. Since the rank is unchanged, there cannot be any new linear combinations possible. Since, there are no new transitions added, it can be trivially verified that all the elements of $\mathbf{PP}$ are valid as-is in the new net N', i.e. $\mathbf{PP} = \mathbf{PP'}$.

$\mathbf{TT'}$ As $\mathbf{p}$ is linearly dependent on the rows of $\mathbf{N}$, the rank is not changed, and no new linear combinations are possible. However, since there is a newly added row, all the vectors from $\mathbf{TT}$ need to be extended corresponding to the row of the newly added place p. In Theorem 1, for a vector $\mathbf{tt}$ this value is chosen to be $\lambda \cdot \mathbf{tt}$. We now show that this is indeed the correct value. Without loss of generality, lets assume $\mathbf{p}$ is the last row of $\mathbf{N'}$:

$$\mathbf{N'} = \begin{pmatrix} \mathbf{N} \\ \lambda \cdot \mathbf{N} \end{pmatrix} \quad \textit{as } \mathbf{p} = \lambda \cdot \mathbf{N} \qquad (1)$$

Let $\mathbf{tt} \in \mathbf{TT}$. We know that $\mathbf{tt}$ is linearly dependent on the columns of $\mathbf{N}$, i.e. for some μ it holds that $\mathbf{tt} = \mathbf{N} \cdot \mu$ We can obtain a corresponding vector which is linearly dependent on the columns of $\mathbf{N'}$ as: $\mathbf{N'} \cdot \mu$. If we extend $\mathbf{N'}$ with such a vector as the last column, then we get $\begin{pmatrix} \mathbf{N'} \, \mathbf{N'} \cdot \mu \end{pmatrix}$. From Equation 1, we have

$$\begin{pmatrix} \mathbf{N} & \mathbf{N} \cdot \mu \\ \lambda \cdot \mathbf{N} & \lambda \cdot \mathbf{N} \cdot \mu \end{pmatrix} = \begin{pmatrix} \mathbf{N} & \mathbf{tt} \\ \lambda \cdot \mathbf{N} & \lambda \cdot \mathbf{tt} \end{pmatrix}, \textit{ as we know that } \mathbf{N} \cdot \mu = \mathbf{tt}$$

Therefore, for $\mathbf{tt}$ to be linearly dependent in $\mathbf{N'}$ the value of the row corresponding to the newly added place should be $\lambda \cdot \mathbf{tt}$. This is exactly whats done in Theorem 1. This is irrespective of the λ chosen. Consider two vectors $\lambda_\mathbf{1}$ and $\lambda_\mathbf{2}$ such that $\lambda_\mathbf{1} \cdot \mathbf{N} = \mathbf{p} = \lambda_\mathbf{2} \cdot \mathbf{N} \wedge \lambda_\mathbf{1} \neq \lambda_\mathbf{2}$. Then $\lambda_\mathbf{1} \cdot \mathbf{tt} = \lambda_\mathbf{2} \cdot \mathbf{tt} : \lambda_\mathbf{1} \cdot \mathbf{tt} = \lambda_\mathbf{1} \cdot \mathbf{N} \cdot \mu = \mathbf{p} \cdot \mu = \lambda_\mathbf{2} \cdot \mathbf{N} \cdot \mu = \lambda_\mathbf{2} \cdot \mathbf{tt}$.

Let's re-visit the running example from Figure 1, and the usage of ψ_P as shown in Figure 3a. The incidence matrices of the nets before and after the usage of ψ_P are shown in Figure 5, which also contains three valid ISS column vectors and the changes to those vectors using Theorem 1. It is clear that all the three derived vectors satisfy condition (1) of Definition 9 in the derived net.

	t_1	t_2	t_3	t_4
p_1	-1	0	0	1
p_2	1	-1	0	0
p_3	0	1	-1	0
p_4	0	0	1	-1

$\mathbf{tt_1}$	$\mathbf{tt_2}$	$\mathbf{tt_3}$
1	-1	1
-1	1	0
0	0	-1
0	0	0

(a) Incidence matrix $\mathbf{N_3}$ and valid ISS column vectors corresponding to the net N_3 from Figure 1 ($\mathbf{tt_1} = \mathbf{t_2} + \mathbf{t_3} + \mathbf{t_4}$, $\mathbf{tt_2} = \mathbf{t_1}$ and $\mathbf{tt_3} = \mathbf{t_3} + \mathbf{t_4}$).

	t_1	t_2	t_3	t_4
p_1	-1	0	0	1
p_2	1	-1	0	0
p_3	0	1	-1	0
p_4	0	0	1	-1
p_5	1	0	0	-1

$\mathbf{tt_1}$	$\mathbf{tt_2}$	$\mathbf{tt_3}$
1	-1	1
-1	1	0
0	0	-1
0	0	0
-1	1	-1

(b) Incidence matrix $\mathbf{N_4}$ and the updated vectors corresponding to the net N_4 from Figure 3a (after using ψ_P).

Fig. 5: Example showing the usage of Theorem 1.

$(\{p_2, p_5\}, \{p_1\}) \in \mathfrak{N}(\mathbf{tt_1})$, and hence $(\{p_2, p_5\}, \{p_1\})$ satisfies condition (2)(b) of Definition 9 in the derived net. Hence $\mathbf{tt_1}$ is a valid ISS column vector in the new net too. $(\{p_1\}, \{p_2, p_5\}) \in \mathfrak{N}(\mathbf{tt_2})$, which satisfies condition (2)(a) according to Definition 9. Hence $\mathbf{tt_2}$ is a valid ISS column vector in the new net too. However, there is no pair in $\mathfrak{N}(\mathbf{tt_3})$ which satisfies either condition (2)(a) or (2)(b) of Definition 9, hence $\mathbf{tt_3}$ is not a valid ISS vector in the new net, and hence its removed according to Theorem 1.

Similarly, the incremental synthesis structure can be updated after the usage of ψ_T, as follows.

Theorem 2 (Extracting *ISS* after ψ_T). *Let $N = (P, T, F)$ and $N' = (P', T', F')$ be two well-formed FC nets, where $(N, N') \in \psi_T$, where $\{t\} = T' \setminus T$. Let μ be such that $\mathbf{N} \cdot \mu = \mathbf{t}$. As $(N, N') \in \psi_T$, such a μ exists. Let $ISS(W) = (\mathbf{TT}, \mathbf{PP})$ be the incremental synthesis structure corresponding to the net N. Then we can extract $\mathbf{PP'}$ and $\mathbf{TT'}$:*

- $\mathbf{TT'} = \mathbf{TT}$
- $\mathbf{PP'} = \bigcup_{\mathbf{pp} \in \mathbf{PP}} f_p(\mathbf{pp})$ *where,*
$$f_p(\mathbf{pp}) = \begin{cases} \mathbf{pp}^{\frown t \mapsto \mathbf{pp}\cdot\mu}, \textit{ if } \mathbf{pp}^{\frown t \mapsto \mathbf{pp}\cdot\mu} \textit{ is a valid ISS vector in } N' \\ \emptyset, \textit{ otherwise} \end{cases}$$

s.t. $ISS(N') = (\mathbf{TT'}, \mathbf{PP'})$ is the incremental synthesis structure of N'.

The proof of Theorem 2 is symmetrical to the proof of Theorem 1.

4.2 ISS updates for ψ_A

Following the updates to the incremental synthesis structure after the usage of ψ_P and ψ_T, we now discuss the approach used for extracting the incremental synthesis structure after the usage of ψ_A. Before that, we discuss a couple of lemmata, which are used to support the theorem for extracting incremental synthesis structure after the usage of ψ_A.

Lemma 3. *Let $N = (P, T, F)$ and $N' = (P', T', F')$ be two well-formed FC nets, such that $(N, N') \in \psi_A, p = P' \setminus P$ and $t = T' \setminus T$. Let a vector $\mathbf{tt'}$ be linearly dependent on the columns of $\mathbf{N'}$, such that $\mathbf{tt'}(p) = 0$. Consider a vector $\mathbf{tt}$, such that $\mathbf{tt'} = \mathbf{tt}^{\frown p \mapsto 0}$. Then $\mathbf{tt}$ is linearly dependent on the columns of $\mathbf{N}$ iff $\mathbf{tt'}$ is linearly dependent on the columns of $\mathbf{N'}$.*

Proof. Without loss of generality, if we assume that the last row of $\mathbf{N}'$ corresponds to p and the last column of $\mathbf{N}'$ corresponds to t, then from [8] (pg. 139), $\mathbf{N}'$ can be decomposed such that: $\mathbf{N}' = \tilde{\mathbf{N}} \cdot A^{-1}$ where

$$\tilde{\mathbf{N}} = \begin{pmatrix} \mathbf{N} & B \\ 0 \dots 0 & -1 \end{pmatrix} \textit{ where } \begin{pmatrix} B \\ -1 \end{pmatrix} \textit{ is the last column of } \mathbf{N}'$$

and A^{-1} is a $T \times T$ matrix, s.t.: $A^{-1}[u,v] = \begin{cases} 1 & \text{if } u = v \\ -1 & \text{if } u = t \wedge v \in \overset{N'}{\bullet} p \\ 0 & \text{otherwise} \end{cases}$

=> If $\mathbf{tt}'$ is linearly dependent, we have $\mathbf{tt}' = \mathbf{N}' \cdot \mu$. Therefore, $\mathbf{tt}' = \tilde{\mathbf{N}} \cdot A^{-1} \cdot \mu = \tilde{\mathbf{N}} \cdot \gamma$, where $\gamma = A^{-1} \cdot \mu$. We can re-write this as:

$$\begin{pmatrix} \mathbf{tt}'' \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{N} & B \\ 0 \dots 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y} \\ \gamma(p) \end{pmatrix} \textit{ where } \gamma = \begin{pmatrix} \mathbf{Y} \\ \gamma(p) \end{pmatrix} \wedge \mathbf{tt}' = \begin{pmatrix} \mathbf{tt}'' \\ 0 \end{pmatrix}$$

From above, we have $\gamma(p) = 0$. Since $\gamma(p) = 0$ and $\mathbf{tt}'(p) = 0$ we have:

$$\mathbf{tt}'' = \mathbf{N} \cdot \mathbf{Y}$$

Hence, the vector $\mathbf{tt}'' = \mathbf{tt}$ is linearly dependent on the columns of $\mathbf{N}$.

<= Similarly, by following the steps above in reverse, we can show that if $\mathbf{tt}$ is linearly dependent on the columns of $\mathbf{N}$, then $\mathbf{tt}^{\frown p \mapsto 0}$ is linearly dependent on the columns of $\mathbf{N}'$.

Lemma 4. *Let N be a well-formed FC net, let $\mathbf{tt}$ be a valid ISS vector in N satisfying condition (1), let $(P_I, P_O) \in \mathfrak{N}(\mathbf{tt})$, and let $R \subseteq T$ be such that $R \times P_I \subseteq F \wedge R \neq \emptyset$. Let $(N, N') \in \psi_A$, s.t. $t = T' \setminus T \wedge p = P' \setminus P \wedge t \overset{N'}{\bullet} = P_I$. Then $\mathbf{tt}^{\frown p \mapsto 0} + \mathbf{t}$ is a valid ISS vector in N', and there exists a pair ($\{p\}$, $P_O) \in \mathfrak{N}(\mathbf{tt}^{\frown \mathbf{p} \mapsto \mathbf{0}} + \mathbf{t})$ that satisfies condition (2)(a) of Definition 9.*

Proof. By construction of ψ_A, we know the values of the column vector corresponding to t in $\mathbf{N}'$ are: $\mathbf{t}(s) = \begin{cases} -1, & \text{if } s = p \\ 1, & \text{if } s \in P_I \\ 0, & \text{otherwise} \end{cases}$

As $\mathbf{tt}$ is linearly dependent on the columns of $\mathbf{N}$, from Lemma 3, we know that a vector $\mathbf{tt}^{\frown p \mapsto 0}$ is linearly dependent on the columns of $\mathbf{N}'$, and hence a vector $\mathbf{tt}' = \mathbf{tt}^{\frown p \mapsto 0} + \mathbf{t}$ is linearly dependent on columns of $\mathbf{N}'$ too. The values of such a vector $\mathbf{tt}'$ are as follows:

$$\mathbf{tt}'(s) = \begin{cases} 0, \text{ when } s \in P_I \setminus P_O & \text{as } \mathbf{tt}^{\frown \mathbf{p} \mapsto \mathbf{0}}(s) = -1 \wedge \mathbf{t}(s) = 1 \\ 1, \text{ when } s \in P_O \setminus P_I & \text{as } \mathbf{tt}^{\frown \mathbf{p} \mapsto \mathbf{0}}(s) = 1 \quad \wedge \mathbf{t}(s) = 0 \\ 1, \text{ when } s \in P_I \cap P_O & \text{as } \mathbf{tt}^{\frown \mathbf{p} \mapsto \mathbf{0}}(s) = 0 \quad \wedge \mathbf{t}(s) = 1 \\ -1, \text{ when } s = p & \text{as } \mathbf{tt}^{\frown \mathbf{p} \mapsto \mathbf{0}}(s) = 0 \quad \wedge \mathbf{t}(s) = -1 \\ 0, \text{ otherwise} & \text{as } \mathbf{tt}^{\frown \mathbf{p} \mapsto \mathbf{0}}(s) = 0 \quad \wedge \mathbf{t}(s) = 0 \end{cases}$$

As $\mathbf{tt}' : P' \to \{-1, 0, 1\}$ and $\mathbf{tt}'$ is linearly dependent on the columns of $\mathbf{N}'$, it satisfies condition (1) from Definition 9. We have a pair $(\tilde{P_I}, \tilde{P_O}) \in \mathfrak{N}(\mathbf{tt}')$, such that $\tilde{P_I} = \{p\} \wedge \tilde{P_O} = P_O$. Since $\overset{N'}{\bullet} t = \tilde{P_I}$, $\mathbf{tt'}$ is valid according to condition (2)(a) of Definition 9 for the pair $(\{p\}, P_O)$.

A similar argument can be made about a row vector which is valid according to condition (2)(b) of Definition 9.

Lemma 5. *Let $N = (P, T, F)$ be a well-formed FC net. Let $(N, N') \in \psi_A$ s.t. $\{p\} = P' \setminus P \wedge \{t\} = T' \setminus T$, and let $\overset{N'}{\bullet} p = R \wedge t \overset{N'}{\bullet} = S$. Let $\mathbf{tt'}$ be a valid ISS column vector in the net N'. Let $(P_I, P_O) \in \mathfrak{N}(\mathbf{tt'})$ which satisfies either condition (2)(a) or (2)(b) of Definition 9. Then, $p \in P_I \implies S \cap P_I = \emptyset$, and $p \in P_O \implies S \cap P_O = \emptyset$.*

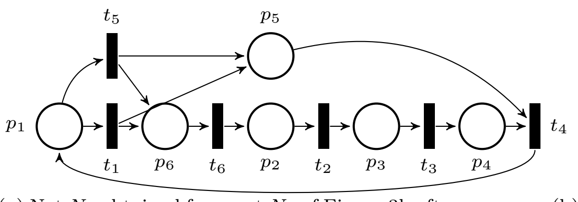
Proof. The crux of this proof lies in the fact that well-formedness implies S-coverability. As a result, all synthesis rules preserve S-coverability. For both condition (2)(a) and condition (2)(b) (with Lemma 4) it can be shown that any S-component covering a place $s \in S$ also has to cover the place p. Adding a transition which would have s and p both as inputs (outputs), would necessarily break all S-components for place s, leaving s uncovered. As a result, if p is an input (output) of a transition to add, then s can not be an input (output) of this transition as well (see Figure 6). Using T-coverability, we can prove a similar theorem on R and t.

Having discussed lemmata and theorem related to the usage of ψ_A , we now discuss the theorem for updating the prior elements and creating new elements in the incremental synthesis structure after the usage of ψ_A.

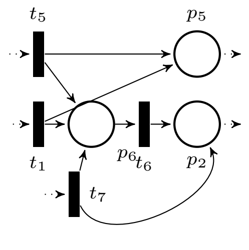
Theorem 3 (Extracting *ISS* after usage of ψ_A). *Let $N = (P, T, F)$ be a well-formed FC net and let $ISS(N) = (\mathbf{TT}, \mathbf{PP})$ be its incremental synthesis structure. Let $N' = (P', T', F')$, such that $(N, N') \in \psi_A$. Note that, $\{p\} = P' \setminus P \wedge \{t\} = T' \setminus T$. Let $\overset{N'}{\bullet} p = R \wedge t \overset{N'}{\bullet} = S$. Then we can extract $\mathbf{PP'}$ and $\mathbf{TT'}$:*

– $\mathbf{TT'} = \bigcup_{\mathbf{tt} \in \mathbf{TT}} f_a(\mathbf{tt})$ *where,*

$$f_a(\mathbf{tt}) = \begin{array}{l} \{\mathbf{tt}^{\frown p \mapsto 0} \mid \mathbf{tt}^{\frown p \mapsto 0} \text{ is a valid ISS vector in } N' \} \cup \\ \{\mathbf{tt}^{\frown p \mapsto 0} + \mathbf{t} \mid \mathbf{tt}^{\frown p \mapsto 0} + \mathbf{t} \text{ is a valid ISS vector in } N' \} \cup \\ \{\mathbf{tt}^{\frown p \mapsto 0} - \mathbf{t} \mid \mathbf{tt}^{\frown p \mapsto 0} - \mathbf{t} \text{ is a valid ISS vector in } N' \} \end{array}$$



(a) Net N_6 obtained from net N_5 of Figure 3b after using ψ_A rule with $R = \{t_1, t_5\} \wedge S = \{p_2\}$.



(b) Fragment of N_6 after using ψ_T rule to add t_7 with some input P_I and output $\{p_2, p_6\}$. No S-component covers p_2.

Fig. 6: Example demonstrating Lemma 5.

Table 3: Values corresponding to an arbitrary place s.

$s = p$	$s \in \tilde{P_I}$	$s \in \tilde{P_O}$	$s \in S$	**t**	$\mathbf{tt'_1}$	$\mathbf{tt'_2}$	$\mathbf{tt'_3}$	$\mathbf{tt'_4}$
✓	✗	✗	✗	-1	0	-1	1	0
✗	✗	✗	✓	1	0	0	0	0
✗	✗	✓	✗	0	1	1	1	1
✗	✓	✗	✗	0	-1	-1	-1	-1
✗	✗	✓	✓	0	-	1	-	1
✗	✓	✗	✓	0	-	-	-1	-1
✗	✓	✓	✗	0	0	0	0	0
✗	✓	✓	✓	0	-	-	-	0
✗	✗	✗	✗	0	0	0	0	0

– $\mathbf{PP'} = \bigcup_{\mathbf{pp} \in \mathbf{PP}} f_a(\mathbf{pp})$ *where,*

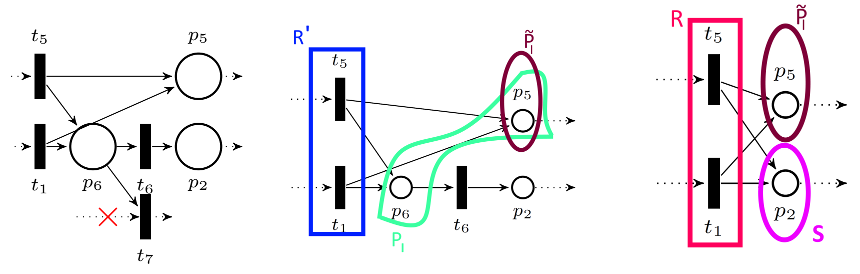
$$f_a(\mathbf{pp}) = \begin{array}{l} \{\mathbf{pp}^{\frown t \mapsto 0} \mid \mathbf{pp}^{\frown t \mapsto 0} \text{ is a valid ISS vector in } N' \} \cup \\ \{\mathbf{pp}^{\frown t \mapsto 0} + \mathbf{p} \mid \mathbf{pp}^{\frown t \mapsto 0} + \mathbf{p} \text{ is a valid ISS vector in } N' \} \cup \\ \{\mathbf{pp}^{\frown t \mapsto 0} - \mathbf{p} \mid \mathbf{pp}^{\frown t \mapsto 0} - \mathbf{p} \text{ is a valid ISS vector in } N' \} \end{array}$$

s.t. $ISS(N')$= $(\mathbf{PP'}, \mathbf{TT'})$ *is the incremental synthesis structure of* N'*.*

Proof. We assume that **TT** and **PP** are correct and complete, and by induction, show that $\mathbf{TT'}$ and $\mathbf{PP'}$ are then correct and complete. Theorem 3 is correct by construction as only the valid ISS vectors are added to $\mathbf{TT'}$ and $\mathbf{PP'}$. Therefore, we only need to show that Theorem 3 is complete according to Definition 10.

We take an arbitrary valid ISS vector $\mathbf{tt'} \in \mathbf{TT'}$, and show that we can obtain it from some vector $\mathbf{tt} \in \mathbf{TT}$. Since $\mathbf{tt'}$ is a valid ISS vector in N', we know that there exists a pair $(P_I, P_O) \in \mathfrak{N}(\mathbf{tt'})$, which satisfies either condition (2)(a) or condition (2)(b) of Definition 9. We prove the completeness based on the presence (or absence) of newly added place p in P_I and P_O. We can have four cases: *(i)* $p \in P_I \wedge p \in P_O$, in this case we refer to the vector as $\mathbf{tt'_1}$, *(ii)* $p \in P_I \wedge p \notin P_O$, in this case we refer to the vector as $\mathbf{tt'_2}$, *(iii)* $p \notin P_I \wedge p \in P_O$, in this case refer to the vector as $\mathbf{tt'_3}$, and *(iv)* $p \notin P_I \wedge p \notin P_O$, in this case refer to the vector as $\mathbf{tt'_4}$.

Let $\tilde{P_I} = P_I \setminus \{p\}$ and $\tilde{P_O} = P_O \setminus \{p\}$. The value corresponding to an arbitrary place s in the vectors $\mathbf{tt'_1}$,$\mathbf{tt'_2}$,$\mathbf{tt'_3}$,$\mathbf{tt'_4}$ and **t** is shown in Table 3. For example,



(a) Fragment of net N_6 from Figure 6a. Transition t_7 is extracted from a pair which satisfies condition (2)(a) of Definition 9. If $p_6 \in P_I$, then $P_I = \{p_6\}$.

(b) Fragment of N_6. A pair (P_I, P_O) that satisfies condition (2)(b) or Definition 9, such that $R' = \{t_1, t_5\} \wedge P_I = \{p_5, p_6\}$.

(c) Fragment from Figure 3b, corresponding to net N_5, such that $R = R' \wedge S = \{p_2\} \wedge \tilde{P_I} = \{p_5\}$.

Fig. 7: Demonstration of Theorem 3 using well-formed FC Petri net fragments.

for a place s, such that $s \neq p \wedge s \in \tilde{P_I}, S \wedge s \notin \tilde{P_O}$ the value of $\mathbf{tt'_3}(s)$ is -1, according to Table 3. The values for some of the elements are left blank. These are the impossible cases which would otherwise violate Lemma 5. For example, for $\mathbf{tt'_2}$, the value corresponding to $s \neq p \wedge s \in \tilde{P_I}, S \wedge s \notin \tilde{P_O}$ is empty. This is because in the case (ii) corresponding to $\mathbf{tt'_2}$, $p \in P_I$. Hence $S \cap \tilde{P_I} = \emptyset$. We now show the proof of completeness for case (i) above, i.e. when *(i)* $p \in P_I \wedge p \in P_O$ ($\mathbf{tt'_1}$). The proof for completeness of the other cases is similar.

(i) $p \in P_I \wedge p \in P_O$ **($\mathbf{tt'_1}$)** Consider a vector $\mathbf{tt}$, such that $\mathbf{tt'_1} = \mathbf{tt}^{\frown p \mapsto 0}$. Using Lemma 3, we can say that $\mathbf{tt}$ satisfies the condition (1) of Definition 9 in N. For all places except p, $\mathbf{tt}$ has the same values as $\mathbf{tt'_1}$. Hence from Table 3, we know that $(S \cup \tilde{P_I}, S \cup \tilde{P_O}) \in \mathfrak{N}(\mathbf{tt})$ in N. We show that this pair satisfies the condition (2)(b) of Definition 9 in N, and thus using $\mathbf{tt}$ we can get $\mathbf{tt'_1}$. Since $\mathbf{tt'_1}$ is a valid ISS vector, we have two cases:

$\mathbf{tt'_1}$ satisfies condition (2)(a) Assume $r \in T'$ such that $\overset{N'}{\bullet} r = P_I$. From the construction of ψ_A, we know that $\overset{N'}{\bullet} t = \{p\}$. As $p \in P_I$ and the net is free-choice, we conclude that $P_I = \{p\}$. Hence, we have $\tilde{P_I} = \emptyset$ (see Figure 7a). Hence the set of input places is only S. However, by construction of ψ_A, $R \times S \subseteq F$ in N. Hence $(S \cup \tilde{P_I}, S \cup \tilde{P_O})$ satisfies condition (2)(b) of Definition 9 in the net N.

$\mathbf{tt'_1}$ satisfies condition (2)(b) Assume $R' \subseteq T'$ such that $R' \times P_I \subseteq F'$, i.e. $R' \times (\{p\} \cup \tilde{P_I}) \subseteq F'$. As $R' \times \{p\} \subseteq F'$, we know that $R' \subseteq R$. Hence, by construction of ψ_A, we know that in the net N, $R' \times (\tilde{P_I} \cup S) \subseteq F$ (see Figure 7b and Figure 7c). Hence $(S \cup \tilde{P_I}, S \cup \tilde{P_O})$ satisfies condition (2)(b) of Definition 9 in the net N.

The proof of correctness and completeness for **PP** is symmetrical.

5 Implementation and Application

A variant of the proposed approach has also been implemented in the "InteractiveProcessMining" package of the process mining toolkit ProM [15], that serves as the basis for enabling interactive process discovery/modeling of sound business processes. The extraction from well-formed free-choice nets to business processes is outside the scope of this paper, but is straightforward and is well established in the literature [1]. For example, Figure 8 shows the snapshot of a business process built using the implementation from ProM. The business process from Figure 8 is indeed the net from Figure 1 obtained by removing the arc from t_4 to p_1 and adding a new place and an arc from t_4 to the newly added place. Further, the transitions t_2 and t_3 are labeled as workflow activities *Patient enters* and *Get medicines* respectively. Such an editor can be used as a stand-alone business process editor or can be combined with the information from the event logs

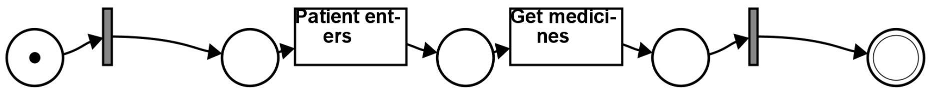


Fig. 8: Snapshot of the implementation from ProM for modeling/discovering business processes using the incremental synthesis structure as the basis.

recorded in the information systems to enable interactive process discovery. The editor allows only those edit operations that are allowed by the synthesis rules, thus guaranteeing the soundness of modeled/discovered process models. [9,10] show the applications of such an editor to discover process models and repair event logs, in the context of process mining. Moreover, calculating the synthesis rules in an incremental way prevents long waiting times, as shown in Section 6.

6 Evaluation

In this section, the time taken to compute the synthesis space by using the incremental synthesis structure (ISS) approach is compared with the Brute force (BF) approach, under the same conditions and by using the same solver. In order to do so, starting with the initial net N_0, a random net was synthesized by using a random synthesis rule. Another random synthesis rule was applied on the synthesized net, to obtain a new random net. This was repeated until the application of 250 random synthesis rules. At any point, each of the three synthesis rules (ψ_A, ψ_P, ψ_T) have equal chance of being chosen. This essentially relates to choosing a random net from the synthesis space of Subsection 2.1. Figure 9 shows the experimental set-up. After applying a synthesis rule, the time taken for computing the synthesis space using the BF approach and the ISS approach were recorded. In an interactive setting, for e.g. for editing business process models, the waiting times for user should be as short as possible. Hence the synthesis space calculation was aborted if the time taken to compute the synthesis space exceeded 5000 milliseconds. This experiment was repeated 30 times in total. That is, starting with the initial net, random synthesis rules were applied 250 times, for a total of 30 times.

The average time taken to compute the synthesis space using the BF approach and the ISS approach at each synthesis iteration is plotted in Figure 10a. It should be noted that the time scale is logarithmic in nature. In order to compare the BF approach and the ISS approach effectively, only time averages below 1000 ms are plotted. For the BF approach, the time taken to compute synthesis space rises quickly and exponentially. Just after 10 synthesis iterations, the average time taken to compute synthesis space using BF approach is more than 5000

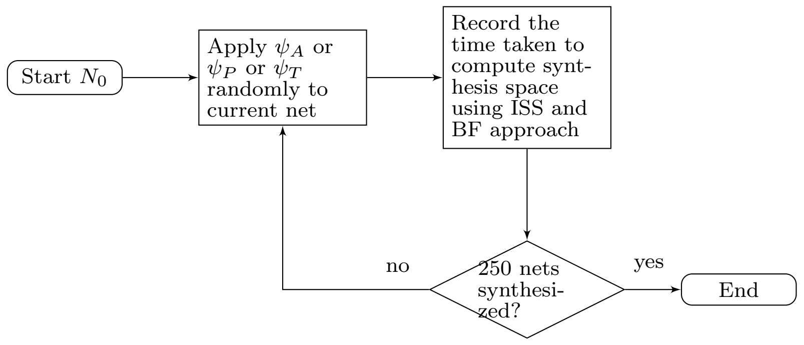

Start N_0
Apply ψ_A or ψ_P or ψ_T randomly to current net
Record the time taken to compute synthesis space using ISS and BF approach
250 nets synthesized?
no
yes
End

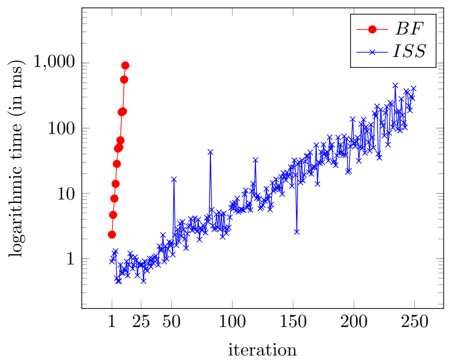
Fig. 9: The experimental setup to test the the performance of Incremental Synthesis Structure (ISS) vs Brute Force (BF) approach (repeated 30 times in total).

milliseconds. The high computation times for BF approach can be attributed to the following factors:

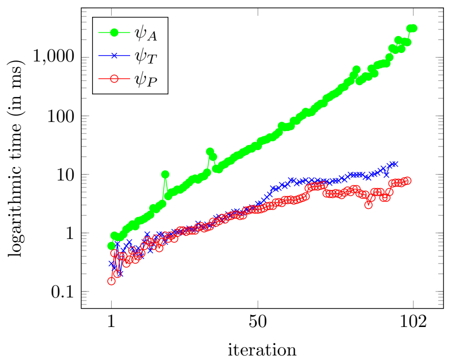
1. As the number of nodes grows with each synthesis iteration, the number of possible permutations grows exponentially. In the BF approach, calculation of all the possible applications of ψ_T and ψ_P rules requires exploration of all the possible vector permutations, to generate the possible candidates.
2. For each of the generated candidates, it is verified if the net would remain a free-choice net. That is, it is verified if there exists at least one pair in Definition 8, using which the net would remain a free-choice net. All the candidates that violate this condition are removed.
3. For all the remaining candidates, it is verified if it is linearly dependent on the incidence matrix of the net.

Clearly, as the number of nodes grows, the number of permutations grows too. Moreover, validating the linear dependence of multiple vectors becomes inefficient rather quickly. Compared to this, in the ISS approach proposed in this paper, the growth seems rather gradual. This can be attributed to the fact that, unlike the BF approach, we do not have to compute all the possible permutations for any given net. More importantly, the third step of verification of linear dependence is not required for our approach. This is due to the fact that, ISS only results in linearly dependent vectors after usage of ψ_A, ψ_P and ψ_T rules. Therefore, our approach mainly deals only with the non-expensive step 2, when compared to the BF approach. Extraction of all the valid ψ_T and ψ_P rules from ISS is rather trivial. Practically, for any non-zero ISS column vector in **TT** to result in ψ_T; it can be quickly verified if the places that have a value of -1 corresponding to them can result in a free-choice node (along with some possible places which have a value of 0 corresponding to them). A similar argument can be made for the row vectors from **PP**.

A lot of variation is observed, in the case of ISS approach, when computing the synthesis space. This is due to the fact that depending on the type of rule



(a) Average time taken for brute force (BF) vs incremental synthesis structure (ISS) approach after 250 random synthesis iterations (repeated 30 times). BF computation was stopped after 10 synthesis iterations as it took longer than 5000 ms.



(b) Average computation times for incremental synthesis structure after usage of ψ_P, ψ_T and ψ_A rules resp.

Fig. 10

used, the computation times of ISS vary, as shown in Figure 10b. For e.g., if a ψ_A rule is used, then additional candidates are generated, each of which needs to be validated according to Definition 9. It should be noted that, the linear dependence condition of these candidate vectors is valid by construction. Compared to ψ_A, after the usage of ψ_P and ψ_T rules, no new candidates are generated. On the contrary, prior candidates are updated and invalid candidates may be removed. Hence, the computation of synthesis space after usage of ψ_P and ψ_T rules is much faster compared to the computation of synthesis rules after the usage of ψ_A rule. It should be noted that in Figure 10b we plot the averages corresponding to each type of synthesis rule. Contrary to this, in Figure 10a we plot the averages across all the rules.

As evident, the time taken for computing the synthesis space grows exponentially with the BF approach. The time taken to compute the synthesis space using ISS also grows exponentially, however this growth is gradual and acceptable in practical circumstances. For e.g., while synthesizing larger nets, the ISS approach took, on an average, less than 1 second even after 250 iterations, i.e. after applying 250 synthesis rules starting with the initial net N_0. It is clear that the proposed approach is much faster and outperforms the BF approach, and hence is suited for synthesizing very large free-choice Petri nets in an interactive way (i.e. by having short waiting times).

7 Conclusion and Future Work

In order to enable interactive editing of well-formed free-choice Petri nets, we presented a robust approach to incrementally calculate all the possible applications of synthesis rules to deduce well-formed free-choice Petri nets. After fixing the incremental synthesis structure of the initial atomic net, we have shown that the incremental synthesis structure can be correctly and completely calculated after the usage of each synthesis rule. As shown in the evaluation, the proposed approach outperforms the brute force approach in terms of speed, without losing on accuracy of the results. Moreover, by using the guarantees of [8], we can conclude that we can use this incremental approach to calculate any well-formed free-choice Petri net, starting with a well-formed initial atomic net. The proposed approach also served as the basis of the winning entry for discovering process models in the process discovery challenge of at the BPM 2017 conference, which also demonstrates very well the applicability of this approach in the field of business process mining. In the future, we would like to explore the possibilities of extending such an incremental synthesis approach in the context of non-free-choice constructs.

References

1. van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.: Soundness of workflow nets: classification, decidability, and analysis. Formal Aspects of Computing 23(3), 333–363 (May 2011)

2. Badouel, E., Bernardinello, L., Darondeau, P.: Process Discovery, pp. 283–300. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
3. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Synthesis of Petri nets from scenarios with viptool. In: International Conference on Applications and Theory of Petri Nets. pp. 388–398. Springer (2008)
4. Berthelot, G.: Transformations and decompositions of nets. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) Petri Nets: Central Models and Their Properties. pp. 359–376. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)
5. Berthelot, G., Lri-Iie: Checking properties of nets using transformations. In: Rozenberg, G. (ed.) Advances in Petri Nets 1985. pp. 19–40. Springer Berlin Heidelberg, Berlin, Heidelberg (1986)
6. Chao, D.Y., Wang, D.T.: Petri net synthesis and synchronization using knitting technique. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics. vol. 1, pp. 652–657 vol.1 (Oct 1994)
7. Datta, A., Ghosh, S.: Synthesis of a class of deadlock-free Petri nets. Journal of the ACM (JACM) 31(3), 486–506 (1984)
8. Desel, J., Esparza, J.: Free choice Petri nets, vol. 40. Cambridge university press (2005)
9. Dixit, P.M., Buijs, J.C.A.M., van der Aalst, W.M.P.: Prodigy : Human-in-the-loop process discovery. In: 2018 12th International Conference on Research Challenges in Information Science (RCIS). pp. 1–12 (May 2018)
10. Dixit, P.M., Suriadi, S., Andrews, R., Wynn, M.T., ter Hofstede, A.H.M., Buijs, J.C.A.M., van der Aalst, W.M.P.: Detection and interactive repair of event ordering imperfection in process logs. In: Krogstie, J., Reijers, H.A. (eds.) Advanced Information Systems Engineering. pp. 274–290. Springer International Publishing, Cham (2018)
11. van Dongen, B.F., van der Aalst, W.M.P., Verbeek, H.M.W.: Verification of epcs: Using reduction rules and petri nets. In: Pastor, O., Falcão e Cunha, J. (eds.) Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005. Proceedings. pp. 372–386. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
12. Esparza, J.: Synthesis rules for Petri nets, and how they lead to new results. In: International Conference on Concurrency Theory. pp. 182–198. Springer (1990)
13. Esparza, J., Hoffmann, P.: Reduction rules for colored workflow nets. In: Proceedings of the 19th International Conference on Fundamental Approaches to Software Engineering - Volume 9633. pp. 342–358. Springer-Verlag New York, Inc., New York, NY, USA (2016)
14. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (Apr 1989)
15. 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: International Conference on Application and Theory of Petri Nets. vol. 3536, pp. 444–454 (2005)
16. Verbeek, H.M.W., van der Aalst, W.M.P.: Woflan 2.0 a petri-net-based workflow diagnosis tool. In: Nielsen, M., Simpson, D. (eds.) International Conference on Applications and Theory of Petri Nets. pp. 475–484. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)