Appeared in Fundamenta Informaticae 183(1-2) : 1–31 (2021).
Available at IOS Press through:
https://doi.org/10.3233/FI-2021-2080

Articulations and Products of Transition Systems and their Applications to Petri Net Synthesis

Raymond Devillers[*]
Département d'Informatique
Université Libre de Bruxelles
Boulevard du Triomphe, C.P. 212, B-1050 Bruxelles, Belgium
rdevil@ulb.ac.be

Abstract. In order to speed up the synthesis of Petri nets from labelled transition systems, a divide and conquer strategy consists in defining decompositions of labelled transition systems, such that each component is synthesisable iff so is the original system. Then corresponding Petri Net composition operators are searched to combine the solutions of the various components into a solution of the original system. The paper presents two such techniques, which may be combined: products and articulations. They may also be used to structure transition systems, and to analyse the performance of synthesis techniques when applied to such structures.

Keywords: labelled transition systems; composition; decomposition; Petri net synthesis.

1. Introduction

Instead of analysing a given system to check if it satisfies a set of desired properties, the synthesis approach tries to build a system "correct by construction" directly from those properties. In particular, more or less efficient algorithms have been developed to build a bounded Petri net (possibly of some subclass, called a PN-solution) with a reachability graph isomorphic to (or close to) a given finite labelled transition system [1, 2, 3, 4, 5].

[*]Address for correspondence: Département d'Informatique, Université Libre de Bruxelles, Boulevard du Triomphe, C.P. 212, B-1050 Bruxelles, Belgium

The synthesis problem is usually polynomial in terms of the size of the LTS, with a degree between 2 and 7 depending on the subclass of Petri nets one searches for [1, 6, 4, 2], but can also be NP-complete [7]. Hence the interest to apply a "divide and conquer" synthesis strategy when possible. The general idea is to decompose the given LTS into components, to synthesise each component separately and then to recombine the results in such a way to obtain a solution to the global problem. We thus have to find a pair of operators, one acting on transition systems and a corresponding one acting on Petri nets, such that a composed LTS has a PN-solution if and only if so are its components, and a possible solution is given by the application of the Petri net operator applied to solutions of the components. It is also necessary to be able to rapidly decompose a given LTS, or to state it is not possible. This is summarised in Figure 1.

$$TS = TS_1 \ \mathbf{op_{TS}} \ TS_2$$

$$TS \text{ PN-solvable} \iff TS_1 \text{ and } TS_2 \text{ PN-solvable}$$

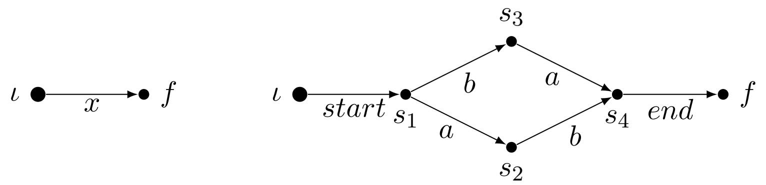
$$sol(TS) = sol(TS_1) \ \mathbf{op_{PN}} \ sol(TS_2)$$

$$TS \Rightarrow \text{discovering of } TS_1 \text{ and } TS_2$$

Figure 1. Divide and conquer strategy for synthesis

We shall here present two such strategies, which have been introduced recently and may be combined efficiently. The first one is given by the disjoint products of LTS, which correspond to disjoint sums of Petri nets [8, 9], and the second one is given by articulations on a state of a transition system and on a non-dominated reachable marking of a Petri net [10], which may occur in various forms (choice, sequence, loop).

An example of their mixed usage is illustrated in Figure 2, where articulations are instantiated in their sequence form, and one of the components has a product form which was not apparent initially. Not only this allows to simplify the Petri net synthesis, if needed, but also this allows to exhibit an interesting internal structure for complex systems which could otherwise be considered as "spaghetti-like".



$TS(x)$ $\qquad TS = TS(start); (TS(a) \otimes TS(b)); TS(end)$

Figure 2. Combination of sequence operators with a product.

The structure of the paper is as follows. First, we shall recall the bases of labelled transition systems and Petri nets. Then, products of transition systems and sums of Petri nets are examined, followed by articulations around states and markings, as well as the combination of both techniques.

Performance issues are detailed in section 6 and, as usual, the last section concludes. With respect to previous papers on the subject, sections 5 and 6 are new, as well as some results and proofs (for instance Propositions 3.10, 4.9, 4.12 and Theorem 3.12); some small improvements are also scattered all over the rest of the presentation.

2. Labelled transition systems and Petri nets

A classic way for representing the possible (sequential) evolutions of a dynamic system is through a (labelled) transition system [11].

Definition 2.1. LABELLED TRANSITION SYSTEMS
A *labelled transition system* (LTS for short) with initial state is a tuple $TS = (S, \rightarrow, T, \iota)$ with node (or state) set S, edge label set T, edges $\rightarrow \subseteq (S \times T \times S)$, and an initial state $\iota \in S$. We shall denote $s[t\rangle$ for $t \in T$ if there is an arc labelled t from s, $[t\rangle s$ if there is an arc labelled t to s, and $s[\alpha\rangle s'$ if there is a path labelled $\alpha \in T^*$ from s to s'. Such a path will also be called an evolution of the LTS (from s to s'); s' is then said reachable from s and the set of states reachable from s is denoted $[s\rangle$.

In some proofs, we shall need the following extension of the reachability notion: for each label $t \in T$ we shall denote by $-t$ the corresponding *reverse label*, i.e., $s[-t\rangle s'$ if $s'[t\rangle s$ (this may also be denoted $s\langle t]s'$). We shall assume that $-T = \{-t \mid t \in T\}$ is disjoint from T, and that $--t = t$; then $\pm T = T \cup -T$ is the set of all forward and reverse labels. The general paths $s[\alpha\rangle s'$ for $\alpha \in (\pm T)^*$ are then defined like the forward ones, and we shall denote by $\langle s\rangle$ the set of states reachable from s through a general path. If $T' \subseteq T$, we shall denote by $\langle \iota\rangle^{T'}$ the set of states reachable from ι with general paths only using labels from T': $\langle \iota\rangle^{T'} = \{s' \in S \mid \iota[\sigma\rangle s' \text{ for some } \sigma \in (\pm T')^*\}$, as well as the restriction of TS to this set (the context will indicate which one is used); similarly, $[\iota\rangle^{T'}$ will be the same but with directed paths only.

If $\sigma \in (\pm T)^*$, we shall denote by $\Psi(\sigma)$ the T-indexed vector in $\mathbb{Z}^T$ such that $\forall t \in T : \Psi(\sigma)(t) =$ the number of occurrences of t in σ minus the number of occurrences of $-t$ in σ, i.e., the generalised Parikh vector of σ. We shall also denote by $s[-\sigma\rangle s'$ the path $s'[\sigma\rangle s$ ran backwards.

In the following we shall only consider finite transition systems, i.e., such that S and T (hence also $\rightarrow$) are finite.

Two transition systems $TS_1 = (S_1, \rightarrow_1, T, \iota_1)$ and $TS_2 = (S_2, \rightarrow_2, T, \iota_2)$ with the same label set T are (state-)isomorphic, denoted $TS_1 \equiv_T TS_2$ (or simply $TS_1 \equiv TS_2$ if T is clear from the context), if there is a bijection $\zeta\colon S_1 \rightarrow S_2$ with $\zeta(\iota_1) = \iota_2$ and $(s, t, s') \in \rightarrow_1 \Leftrightarrow (\zeta(s), t, \zeta(s')) \in \rightarrow_2$, for all $s, s' \in S_1$ and $t \in T$. We shall usually consider LTSs up to isomorphism.

A transition system TS is said totally reachable if each state is reachable from the initial one: $[\iota\rangle = S$.

It is reversible if $\forall s \in [\iota\rangle : \iota \in [s\rangle$, i.e., it is always possible to return to the initial state.
It is deterministic if $\forall s, s', s'' \in S\ \forall t \in T : s[t\rangle s' \wedge s[t\rangle s'' \Rightarrow s' = s''$ and $s'[t\rangle s \wedge s''[t\rangle s \Rightarrow s' = s''$.
It is weakly periodic if, for every $\alpha \in T^*$ and infinite path $s_1[\alpha\rangle s_2[\alpha\rangle s_3 \cdots$, either for every $i, j \in \mathbb{N}$: $s_i = s_j$ or for every $i, j \in \mathbb{N}$: $i \neq j \Rightarrow s_i \neq s_j$ (the second case is of course excluded for finite transition systems). □

A non-trivial LTS is illustrated on of Figure 3.

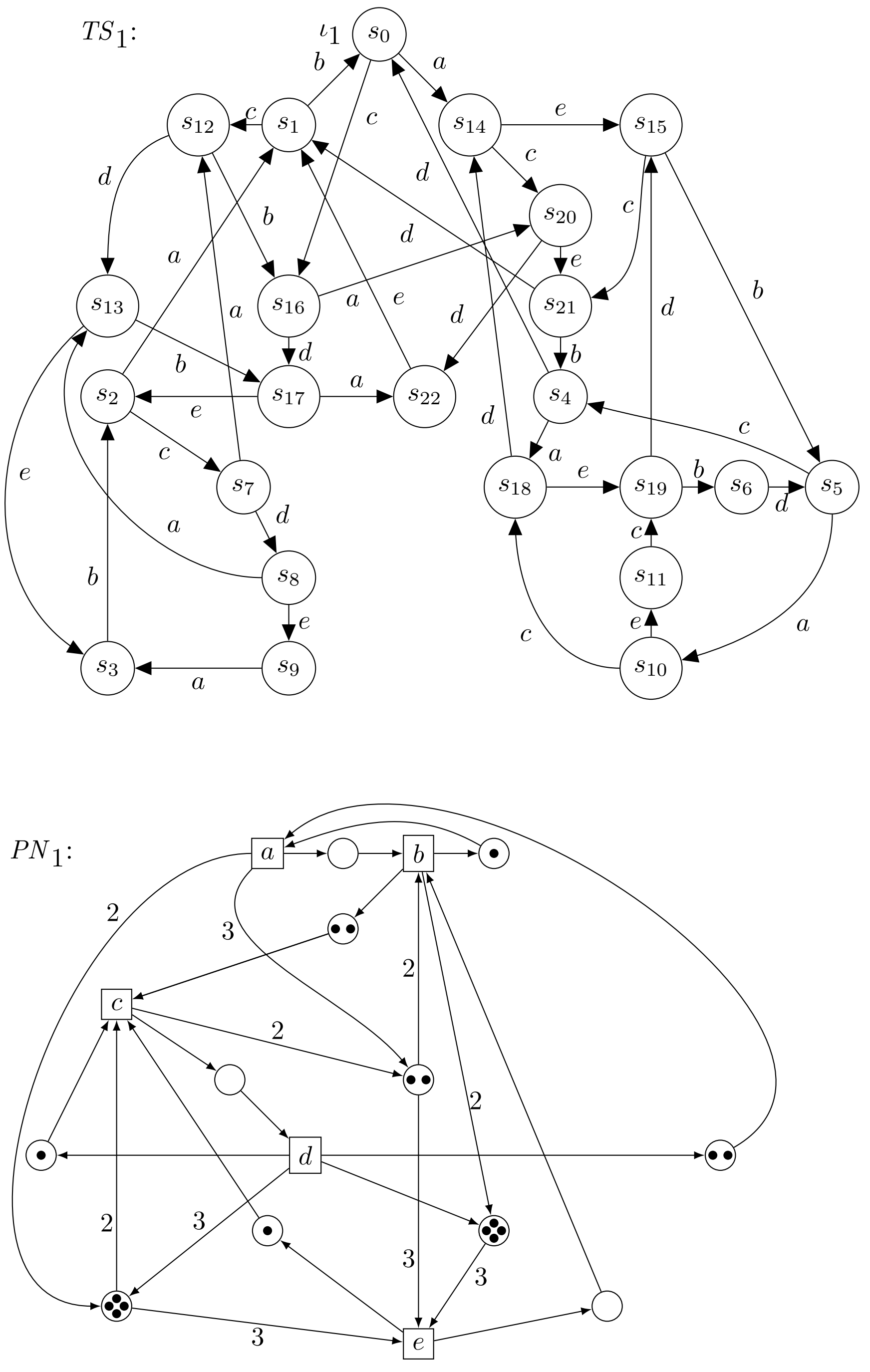


Figure 3. A reversible *LTS*, with a possible Petri net solution.

Definition 2.2. Weighted P/T nets
A (finite, place-transition, arc-weighted) Petri net is a triple $PN = (P, T, F)$ such that P is a finite set of places, T is a finite set of transitions, with $P \cap T = \emptyset$, F is a flow function $F\colon ((P \times T) \cup (T \times P)) \to \mathbb{N}$. The incidence matrix C of PN is the member of $\mathbb{Z}^{P \times T}$ such that $\forall p \in P, t \in T : C(p,t) = F(t,p) - F(p,t)$. The predecessors of a node x form the set ${}^\bullet x = \{y | F(y,x) > 0\}$. Symmetrically its successor set is $x^\bullet = \{y | F(x,y) > 0\}$.

A marking is a mapping $M\colon P \to \mathbb{N}$, indicating the number of (black) tokens in each place. Let M_1 and M_2 be two markings, we shall say that M_1 is dominated by M_2 if $M_1 \lneqq M_2$, i.e., M_1 is distinct from M_2 and componentwise not greater. A Petri net system is a net provided with an initial marking (P, T, F, M_0). A transition $t \in T$ is enabled by a marking M, denoted by $M \xrightarrow{t}$ or $M[t\rangle$, if for all places $p \in P$, $M(p) \geq F(p,t)$. If t is enabled at M, then t can occur (or fire) in M, leading to the marking M' defined by $M'(p) = M(p) - F(p,t) + F(t,p)$ and denoted by $M \xrightarrow{t} M'$ or $M[t\rangle M'$; as usual, $[M\rangle$ denotes the set of markings reachable from M. A Petri net system is bounded if, for some integer k, $\forall M \in [M_0\rangle \forall p \in P : M[p] \leq k$. It is safe if $\forall M \in [M_0\rangle \forall p \in P : M[p] \leq 1$ (i.e., no place will ever receive more than 1 token). It is k-safe, for some known bound k, if $\forall M \in [M_0\rangle \forall p \in P : M[p] \leq k$.

Two Petri net systems $N_1 = (P_1, T, F_1, M_0^1)$ and $N_2 = (P_2, T, F_2, M_0^2)$ with the same transition set T are isomorphic, denoted $N_1 \equiv_T N_2$ (or simply $N_1 \equiv N_2$ if T is clear from the context), if there is a bijection $\zeta\colon P_1 \to P_2$ such that, $\forall p_1 \in P_1, t \in T$: $M_0^1(p_1) = M_0^2(\zeta(p_1))$, $F_1(p_1, t) = F_2(\zeta(p_1), t)$ and $F_1(t, p_1) = F_2(t, \zeta(p_1))$.

The reachability graph of a Petri net system is the labelled transition system whose initial state is M_0, whose vertices are the reachable markings, and whose edges are $\{(M, t, M') \mid M \xrightarrow{t} M'\}$. It is finite iff the system is bounded. Two isomorphic Petri net systems have isomorphic reachability graphs, hence we shall usually consider Petri nets up to isomorphism. In examples, when it is clear that we have an initial marking, we shall often use the shorter terminology "Petri net" instead of the longer "Petri net system". A labelled transition system is PN-solvable if it is isomorphic to the reachablility graph of a Petri net system (called a possible solution). □

A classical property of reachability graphs of Petri net systems is the following:

Proposition 2.3. (State equation)
In the reachability graph of a Petri net system (P, T, F, M_0), if $\sigma \in (\pm T)^*$ and $M[\sigma\rangle M'$, then $M' = M + C \cdot \Psi(\sigma)$.

An immediate observation is that the reachability graph of any Petri net system is totally reachable and deterministic; it is also weakly periodic (see the state equation), and finite when bounded. Hence, if a transition system is not totally reachable or not deterministic, it may not be PN-solvable. The bottom of Figure 3 illustrates a Petri net system, which is a possible PN-solution of the transition system given on top of the same figure. It may happen that a transition system has no PN-solution, but if it has one, it has many ones, sometimes with very different structures.

When linking transition systems and Petri nets, it is useful to introduce regions.

Definition 2.4. REGIONS
A region $(\rho, \mathbb{B}, \mathbb{F})$ of a transition system $TS = (S, \rightarrow, T, \iota)$ is a triple of functions ρ (from states to $\mathbb{N}$), and $\mathbb{B}, \mathbb{F}$ (both from labels to $\mathbb{N}$), satisfying the property that for any states s, s' and label a:

$$(s, a, s') \text{ is an edge of } TS \quad \Rightarrow \quad \rho(s) \geq \mathbb{B}(a) \wedge \rho(s') - \rho(s) = \mathbb{F}(a) - \mathbb{B}(a)$$

since this is the typical behaviour of a place p with token count ρ during the firing of a with backward and forward connections $\mathbb{B}, \mathbb{F}$ to p (anywhere in TS). To solve an SSP(s_1, s_2) (for State Separation Problem, where $s_1 \neq s_2$), we need to find an appropriate region $(\rho, \mathbb{B}, \mathbb{F})$ satisfying $\rho(s_1) \neq \rho(s_2)$, i.e., separating states s_1 and s_2. For an ESSP(s, a) (for Event-State Separation Problem, where a is not enabled at s), we need to find a region $(\rho, \mathbb{B}, \mathbb{F})$ with $\rho(s) < \mathbb{B}(a)$. This can be done by solving suitable systems of linear inequalities which arise from these two requirements, and from the requirement that regions are not too restrictive. □

A classical result [12] is that a transition system is PN-solvable if and only if all its SSP and ESSP problems may be solved, and the corresponding places yield a possible solution.

3. Products and sums

A product of two disjoint *LTS* is again an *LTS*. Its states are pairs of states of the two *LTS* and an edge exists if one of the underlying states can do the transition. An example is shown in Figure 4.

Definition 3.1. PRODUCT OF TWO DISJOINT *LTS*
Let $TS_1 = (S_1, \rightarrow_1, T_1, \iota_1)$ and $TS_2 = (S_2, \rightarrow_2, T_2, \iota_2)$ be two *LTS* with disjoint label sets ($T_1 \cap T_2 = \emptyset$). The (disjoint) product $TS_1 \otimes TS_2$ is the *LTS* $\big(S_1 \times S_2, \rightarrow, T_1 \uplus T_2, (\iota_1, \iota_2)\big)$, where $\rightarrow = \{\big((s_1, s_2), t_1, (s_1', s_2)\big) \mid (s_1, t_1, s_1') \in \rightarrow_1\} \cup \{\big((s_1, s_2), t_2, (s_1, s_2')\big) \mid (s_2, t_2, s_2') \in \rightarrow_2\}$. □

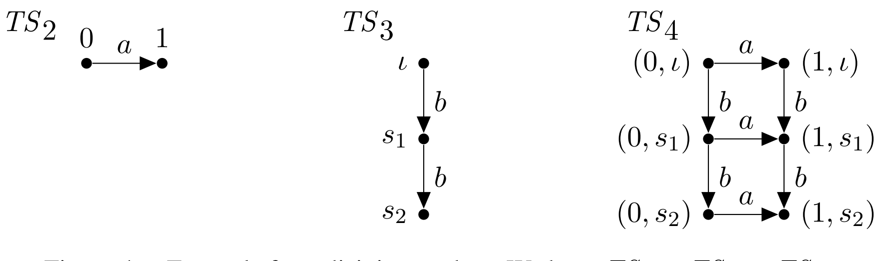


Figure 4. Example for a disjoint product. We have $TS_2 \otimes TS_3 = TS_4$.

When a product is given and the individual label sets T_1 and T_2 are known, the factors can be computed by only following edges with labels in T_1, resp. T_2, from the initial state:

Proposition 3.2. (Factors of a product [8])
When $TS = (S, \rightarrow, T_1 \uplus T_2, \iota) \equiv_T TS_1 \otimes TS_2$, TS is totally reachable (resp. deterministic) iff so are the factors TS_1 and TS_2.

Moreover, $(s_1, s_2)[\sigma\rangle(s_1', s_2')$ in TS iff $s_1[\sigma_1\rangle s_2$ in TS_1 and $s_2[\sigma_2\rangle s_2'$ in TS_2, where σ_1 is the projection of σ on T_1^* and similarly for σ_2.

Then, $TS_1 \equiv_T (S^1, \rightarrow^1, T_1, \iota)$ with $S^1 = \{s \in S \mid \exists \alpha_1 \in T_1^* : \iota[\alpha\rangle s\}$ and $\rightarrow^1 = \{(s_1, t_1, s_2) \in \rightarrow$ such that $s_1, s_2 \in S^1, t_1 \in T_1\}$, and similarly for TS_2.

Up to isomorphism[1], the disjoint product of *LTS* is commutative, associative and has a neutral (the LTS with a single state and no label). Each *LTS* has itself and the neutral system as (trivial) factors.

An *LTS* is prime if it has exactly two factors (up to isomorphism). If $TS = (S, \rightarrow, T, \iota)$ is connected and finite, there is a finite set I of indices and a unique set of connected prime *LTS*'s $\{TS_i | i \in I\}$ such that $TS \equiv_T \bigotimes_{i \in I} TS_i$.

There is an interesting relation between *LTS* products and Petri nets: if two nets are disjoint, putting them side by side yields a new net whose reachability graph is (up to isomorphism) the disjoint product of the reachability graphs of the two original nets. This may be generalised up to Petri net isomorphism:

Definition 3.3. (DISJOINT) SUM OF PETRI NETS
Let $N_1 = (P_1, T_1, F_1, M_0^1)$ and $N_2 = (P_2, T_2, F_2, M_0^2)$ be two Petri net systems with disjoint transition sets ($T_1 \cap T_2 = \emptyset$). The disjoint sum $N_1 \oplus N_2$ is defined (up to isomorphism) as the system $N = (P, T_1 \cup T_2, F, M_0)$ where $P = \zeta_1(P_1) \cup \zeta_2(P_2)$; $F(\zeta_1(p_1), t_1) = F_1(p_1, t_1)$, $F(t_1, \zeta_1(p_1)) = F_1(t_1, p_1)$, $F(\zeta_2(p_2), t_2) = F_2(p_2, t_2)$, $F(t_2, \zeta_2(p_2)) = F_2(t_2, p_2)$, $M_0(\zeta_1(p_1)) = M_0^1(p_1)$ and $M_0(\zeta_2(p_2)) = M_0^2(p_2)$, for $t_1 \in T_1, t_2 \in T_2, p_1 \in P_1, p_2 \in P_2$. In these formulas, ζ_1 is a bijection between P_1 and $\zeta_1(P_1)$ and ζ_2 is a bijection between P_2 and $\zeta_2(P_2)$ such that $\zeta_1(P_1) \cap (\zeta_2(P_2) \cup T_1 \cup T_2) = \emptyset$ and $\zeta_2(P_2) \cap (\zeta_1(P_1) \cup T_1 \cup T_2) = \emptyset$. □

It may be observed that the resulting system is not uniquely defined since it depends on the choice of the two bijections ζ_1 and ζ_2 used to separate the place sets from the rest, but this is irrelevant since we want to work up to isomorphism. Again, up to isomorphism, the disjoint sum of Petri net systems is commutative, associative and has a neutral (the empty net).

An additional remark that may be precious for applications is that many subclasses of Petri nets found in the literature (plain, free-choice, choice-free, join-free, fork-attribution, homogeneous, ..., not defined here) are compatible with the presented (de)composition, in the sense that a disjoint sum of nets belongs to such a subclass if and only if each component belongs to the same subclass. In particular, we have

Corollary 3.4. (Bound preservation)
$N_1 \oplus N_2$ is safe iff so are N_1 and N_2. If $N_1 \oplus N_2$ is k-safe, so are N_1 and N_2.

Finally, if N_1 is k_1-safe and N_2 is k_2-safe, then $N_1 \oplus N_2$ is $\max(k_1, k_2)$-safe.

Proposition 3.5. (Reachability graph of a sum of nets)
The reachability graph of a disjoint sum of net systems is isomorphic to the disjoint product of the reachability graphs of the composing nets.

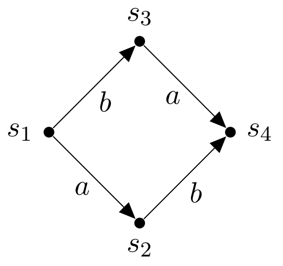
There is a kind of reverse of this property [13, 8].

[1]Those properties could be expressed in terms of categories, but we shall refrain from doing this here.

Proposition 3.6. (Petri net solution of a disjoint product of *LTS*)
A disjoint product of *LTS* has a Petri net solution iff each composing *LTS* has a Petri net solution, and a possible solution is the disjoint sum of the latter.

Let us now examine when and how an *LTS* may be decomposed into (non-trivial) disjoint factors. From Proposition 3.2, it is enough to discover an adequate decomposition of the label set $T = T_1 \uplus T_2$. A general characterisation of such adequate decompositions is presented in [13], but it may be simplified in the context of Petri net synthesis.

Definition 3.7. GENERAL DIAMOND PROPERTY
An *LTS* $TS = (S, \rightarrow, T, \iota)$ presents the *general diamond property* for two distinct labels $a, b \in T$ if, whenever there are two adjacent edges in a diamond like in Fig. 5, the other two are also present (in other words, $\forall s, s_1, s_2 \in S$, $\forall u \in \{a, -a\}, \forall v \in \{b, -b\} : s[u\rangle s_1 \wedge s[v\rangle s_2 \Rightarrow (\exists s' \in S : s_1[v\rangle s' \wedge s_2[u\rangle s'))$. If $T_1, T_2 \subseteq T$ with $T_1 \cap T_2 = \emptyset$, TS will be said $\{T_1, T_2\}$-*gdiam* if it presents the general diamond property for each pair of labels $a \in T_1, b \in T_2$ (note that any *LTS* $TS = (S, \rightarrow, T, \iota)$ is $\{\emptyset, T\}$-gdiam). □



$s_1[a\rangle s_2 \ \wedge \ s_1[b\rangle s_3 \Rightarrow \exists s_4 : s_3[a\rangle s_4 \ \wedge \ s_2[b\rangle s_4$
(forward persistence)
$s_3[a\rangle s_4 \ \wedge \ s_2[b\rangle s_4 \Rightarrow \exists s_1 : s_1[a\rangle s_2 \ \wedge \ s_1[b\rangle s_3$
(backward persistence)
$s_1[a\rangle s_2 \ \wedge \ s_2[b\rangle s_4 \Rightarrow \exists s_3 : s_1[b\rangle s_3 \ \wedge \ s_3[a\rangle s_4$
(permutation)
$s_1[b\rangle s_3 \ \wedge \ s_3[a\rangle s_4 \Rightarrow \exists s_2 : s_1[a\rangle s_2 \ \wedge \ s_2[b\rangle s_4$
(permutation)

Figure 5. General diamond property.

Note that the four configurations in Figure 5 are condensed in a single formula in Definition 3.7, using both direct and reverse arcs. Among the many properties implied by general diamonds, we may cite:

Proposition 3.8. (Projections and permutation [9])
Let $TS = (S, \rightarrow, T, \iota)$ be a $\{T_1, T_2\}$-gdiam *LTS* with $T_1 \cap T_2 = \emptyset$. If $s[\alpha\rangle s'$ for some $s, s' \in S$ and general path $\alpha \in (\pm T_1 \cup \pm T_2)^*$, let α_1 be the projection of α on $\pm T_1$ (i.e., α where all the elements in $\pm T_2$ are dropped) and α_2 be the projection of α on $\pm T_2$ (thus dropping the elements in $\pm T_1$). Then there are $s_1, s_2 \in S$ such that $s[\alpha_1\rangle s_1[\alpha_2\rangle s'$ and $s[\alpha_2\rangle s_2[\alpha_1\rangle s'$.

This also implies a variant of the well-known Keller's theorem [14], meaning that the general diamonds property is local, but implies a global variant.

Proposition 3.9. (General diamonds imply big general diamonds [9])
Let $TS = (S, \rightarrow, T, \iota)$ be a $\{T_1, T_2\}$-gdiam *LTS* with $T_1 \cap T_2 = \emptyset$. If $s[\alpha_1\rangle s_1$ and $s[\alpha_2\rangle s_2$ for some $s, s_1, s_2 \in S$, $\alpha_1 \in (\pm T_1)^*$ and $\alpha_2 \in (\pm T_2)^*$, then for some $s' \in S$, $s_1[\alpha_2\rangle s'$ and $s_2[\alpha_1\rangle s'$.

This may be interpreted as the fact that big general diamonds are filled with small ones.

Proposition 3.10. (Sequences generated by big general diamonds)
Let $TS = (S, \rightarrow, T_1 \uplus T_2, \iota)$ be a $\{T_1, T_2\}$-gdiam *LTS*. If $s_0[\alpha_1\rangle s_1$ and $s_0[\alpha_2\rangle s_1$ for some $s_0, s_1 \in S$, $\alpha_1 \in (\pm T_1)^*$ and $\alpha_2 \in (\pm T_2)^*$,

1. then for any $i \in \mathbb{Z}$, there is some $s_i \in S$ such that $s_i[\alpha_1\rangle s_{i+1}$ and $s_i[\alpha_2\rangle s_{i+1}$;
2. if S is finite, for some $h \neq k \in \mathbb{Z}$, $s_h = s_k$;
3. if the various s_i's belong to $[\iota\rangle^{T_1}$ which is Petri net solvable, then $s_h = s_k$ for any $h, k \in \mathbb{Z}$.

Proof:
The first point results from successive applications of Proposition 3.9 to $s_i[\alpha_1\rangle s_{i+1}$ and $s_i[\alpha_2\rangle s_{i+1}$, as well as to $s_{i+1}[-\alpha_1\rangle s_i$ and $s_{i+1}[-\alpha_2\rangle s_i$.

The case where S is finite is then immediate.

The last point results from the fact that, in any Petri net solution of $[\iota\rangle^{T_1}$, if M_s is the marking corresponding to state s (if reachable), then from Proposition 2.3, $\forall i \in \mathbb{Z} : M_{s_{i+1}} - M_{s_i} = C \cdot \Psi(\alpha_1)$, hence is constant. As a consequence, $\forall i, j \in \mathbb{Z} : M_{s_j} - M_{s_i} = (j - i) \cdot C \cdot \Psi(\alpha_1)$. When S is finite, from the previous point, $C \cdot \Psi(\alpha_1) = 0$, the markings corresponding to all s_i's are the same, hence the s_i's are also the same. But this remains true if TS is infinite, because all the markings are nonnegative (if for some $p \in P$ we have $C \cdot \Psi(\alpha_1) \neq 0$, then either $M_{s_i}(p)$ becomes negative for i large enough, or for i small enough in $\mathbb{Z}$). $\square$

Now, the interest of the notion of general diamonds in our context arises from the following observation:

Proposition 3.11. (Product implies general diamonds [9])
Let $TS_1 = (S_1, \rightarrow_1, T_1, \iota_1)$ and $TS_2 = (S_2, \rightarrow_2, T_2, \iota_2)$ be two *LTS* with disjoint label sets, then $TS_1 \otimes TS_2$ presents the general diamond property for each $a \in T_1$ and $b \in T_2$.

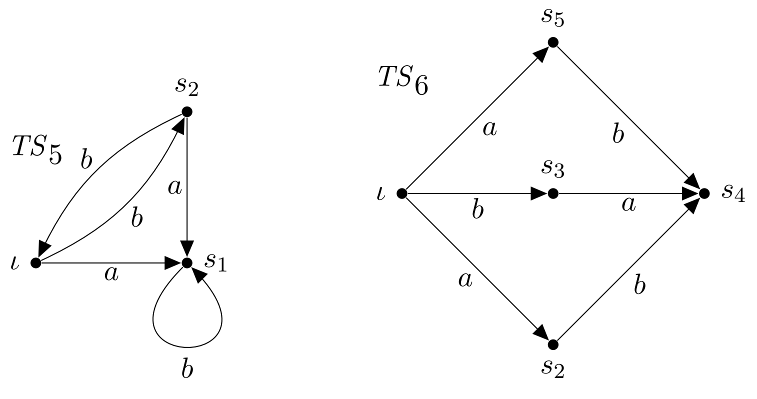


Figure 6. General diamond property does not imply product.

Unfortunately, the reverse property does not hold in all generality, as shown by the counterexamples in Fig. 6. Indeed, in both cases, a and b form general diamonds. However, from Proposition 3.2, with $T_1 = \{a\}$ and $T_2 = \{b\}$, TS_5 should be isomorphic to the product of $\iota \xrightarrow{a} s_1$ and $\iota \xleftrightarrow{b} s_2$, since $\{\iota, s_1\}$ are the only states directly reachable with a and $\{\iota, s_2\}$ are the only states directly reachable with b; however, this is not the case since we would then have four states in the product. Similarly, TS_6 should be isomorphic to the product of $\{\iota \xrightarrow{a} s_2, \iota \xrightarrow{a} s_5\}$ and of a single arrow $\iota \xrightarrow{b} s_3$, which is not the case since then we would have six states in the product.

But note that both counterexample are non-deterministic, hence cannot have Petri net solutions. And indeed, an essential result is that the local general diamond property suffices in the context of synthesis:

Theorem 3.12. [General diamonds and Petri net synthesis imply product]
If a (finite) totally reachable *LTS* $TS = (S, \to, T_1 \uplus T_2, \iota)$ is deterministic and satisfies the general diamond property for each pair of labels $a \in T_1$ and $b \in T_2$, then it is Petri net synthesisable iff so are $[\iota\rangle^{T_1}$ and $[\iota\rangle^{T_2}$; moreover, we then have $TS \equiv_T [\iota\rangle^{T_1} \otimes [\iota\rangle^{T_2}$ and therefore a possible solution of the synthesis problem for TS is the disjoint sum of a solution of $[\iota\rangle^{T_1}$ and a solution of $[\iota\rangle^{T_2}$.

Proof:
First, we may observe that if TS is Petri net synthesisable, (it is deterministic, totally reachable and) by dropping the transitions in T_2 in such a solution we shall get a solution to $[\iota\rangle^{T_1}$, and by dropping the transitions in T_1 we shall get a solution to $[\iota\rangle^{T_2}$.

Let us now assume that TS is totally reachable, deterministic and satisfies the general diamond property for each pair of labels $a \in T_1$ and $b \in T_2$; let us also assume that $[\iota\rangle^{T_1}$ and $[\iota\rangle^{T_2}$ are Petri net synthesisable.

For each state $s \in S$, from the total reachability we have $\iota[\sigma\rangle s$ and, from Proposition 3.8, $\iota[\alpha\rangle s_1[\beta\rangle s$ and $\iota[\beta\rangle s_2[\alpha\rangle s$ for some $s_1, s_2 \in S$, $\alpha \in T_1^*$ and $\beta \in T_2^*$ (α_1 being the projection of σ on T_1^*, and β being the projection of σ on T_2^*), so that $s_1 \in [\iota\rangle^{T_1}$ and $s_2 \in [\iota\rangle^{T_2}$.

Conversely, if $s_1 \in [\iota\rangle^{T_1}$ and $s_2 \in [\iota\rangle^{T_2}$, i.e., there is $\iota[\alpha\rangle s_1$ in $[\iota\rangle^{T_1}$ and $\iota[\beta\rangle s_2$ in $[\iota\rangle^{T_2}$, then from Proposition 3.9 we also have $\iota[\alpha\rangle s_1[\beta\rangle s$ and $\iota[\beta\rangle s_2[\alpha\rangle s$ for some $s \in S$.

It remains to show that the correspondence between s and the pair (s_1, s_2) is unique to derive that $TS \equiv_T [\iota\rangle^{T_1} \otimes [\iota\rangle^{T_2}$ (the compatibility of the transition rules again results from the projections and antiprojections in Propositions 3.8 and 3.9), so that TS is solved by the sum of a solution of $[\iota\rangle^{T_1}$ and a solution of $[\iota\rangle^{T_2}$.

For the direction $(s_1, s_2) \to s$, if $\iota[\alpha_1'\rangle s_1$ for some $\alpha_1' \in T_1^*$, from Proposition 3.9 we have $\iota[\alpha_1'\rangle s_1[\alpha_2\rangle s'$ and $\iota[\alpha_2\rangle s_2[\alpha_1'\rangle s'$ for some $s' \in S$. But from the determinism, $s' = s$. And similarly if $\iota[\alpha_2'\rangle s_2$ with $\alpha_2' \in T_2^*$.

For the other direction, let us assume that $\iota[\sigma\rangle s$ for $\sigma \in (T_1 \cup T_2)^*$ and $s \in S$, and that $\iota[\alpha_1\rangle s_1[\alpha_2\rangle s$ as well as $\iota[\alpha_1'\rangle s_1'[\alpha_2'\rangle s$ with $\alpha_1, \alpha_1' \in T_1^*$, $\alpha_2, \alpha_2' \in T_2^*$ and $s \in S$. We need to show that $s_1 = s_1'$ (the case based on T_2 will be obtained symmetrically).

Since we have both $s_1'[(-\alpha_1')\alpha_1\rangle s_1$ and $s_1'[\alpha_2'(-\alpha_2)\rangle s_1$, with $(-\alpha_1')\alpha_1 \in (\pm T_1)^*$ and $\alpha_2'(-\alpha_2) \in (\pm T_2)^*$, from Proposition 3.10(1) we may construct a series of configurations, forward and backward,

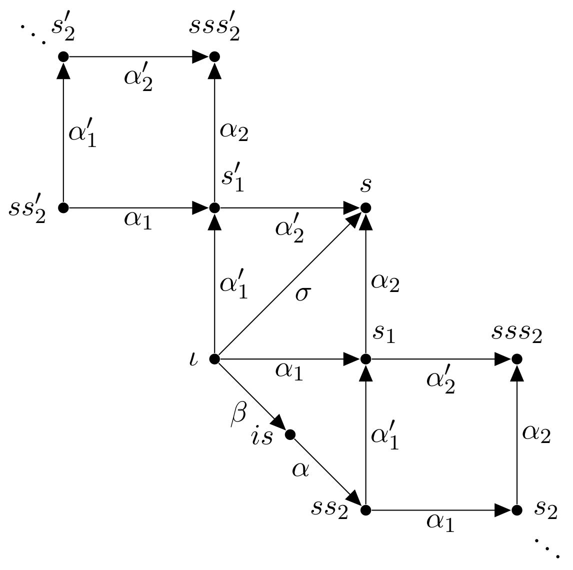


Figure 7. Proof of Theorem 3.12.

as also illustrated in Figure 7, with the same paths $(-\alpha'_1)\alpha_1$ and $\alpha'_2(-\alpha_2)$, leading to states $s_2, s_3, \ldots$ and $s'_2, s'_3, \ldots$

Since TS is totally reachable and satisfies the general diamond property for T_1 and T_2, from Proposition 3.8 we also have a path $\iota[\beta\rangle is[\alpha\rangle ss_2$ with $\beta \in T_2^*$ and $\alpha \in T_1^*$. Since $\iota[\beta\rangle is$ and $\iota[\alpha_1(-\alpha'_1)(-\alpha)\rangle is$ with $\alpha_1(-\alpha'_1)(-\alpha) \in (\pm T)^*$, we may again apply Proposition 3.10(1) (we shall only need it forwardly here), constructing paths of increasing length β^n. Since TS is finite, $[\iota\rangle^{T_2}$ is Petri net solvable and those paths $\iota[\beta^n\rangle$ belong to $[\iota\rangle^{T_2}$, by weak periodicity (see also the proof of Proposition 3.10(3)) we have that $is = \iota$. Hence ss_2 belongs to $[\iota\rangle^{T_1}$, and we may perform the same reasoning for $ss_3, ss_4, \ldots$ (with increasing paths in $(\pm T_1)^*$), as well as for $ss'_2, ss'_3, \ldots$

As a consequence, all the states $s_1, ss_2, s_3, ss_3, \ldots, ss'_2, s'_2, ss'_3, s'_3, \ldots$ belong to $[\iota\rangle^{T_1}$. Since $[\iota\rangle^{T_1}$ is by hypothesis Petri net solvable, we may then apply Proposition 3.10(3) and we get $s_1 = s'_1$, as expected. □

It remains to find adequate subsets of labels T_1 and T_2, partitioning T and satisfying the general diamond property. To do that, one may rely on the following, which is again a local property.

Definition 3.13. CONNECTED LABELS
Let $TS = (S, \to, T, \iota)$ be an *LTS* and $a, b \in T$ be two distinct labels.

We shall denote by $a \not\leftrightarrow b$ the fact that they do not form general diamonds, i.e., there are states which do not satisfy one of the constraints in Figure 5.

We shall also note $\not\Diamond = \not\leftrightarrow^*$, i.e., the reflexive and transitive closure of $\not\leftrightarrow$, meaning in some sense that the labels are 'non-diamondisable'. □

These relations mean that in any decomposition, if $a \not\leftrightarrow b$ they must belong to the same component, i.e., $a \in T_1 \Rightarrow [a] \subseteq T_1$, where $[a] = \{b \in T \mid a \not\Diamond b\}$. But from Theorem 3.12, we know this is enough: for each equivalence class, either the synthesis works and we have a global solution by taking the disjoint sum of all the solutions, or one (or more) of the subproblems fails, and we know there is no global solution for the whole system (and we may spot the culprits).

Our proposed factorisation algorithm now works as follows: First, iterate over all states of the given lts, and for each state check if the adjacent edges form general diamonds. If not, their labels must be in the same equivalence class. Then, for each equivalence class $[a]$ try Petri net synthesis on $[\iota\rangle^{[a]}$. If it works, the result is the disjoint sum of the computed Petri nets. This constructs the equivalence relation by repeatedly joining classes, but it also allows to stop the iteration early when only one equivalence class remains (in which case no non-trivial factorisation is possible).

4. Articulations

Let us consider two disjoint transition systems TS_1 and TS_2 and a state s in the first of them ($s \in S_1$); the general idea of their articulation around s is to 'plug' the second one on the chosen state, as schematised in Figure 8.

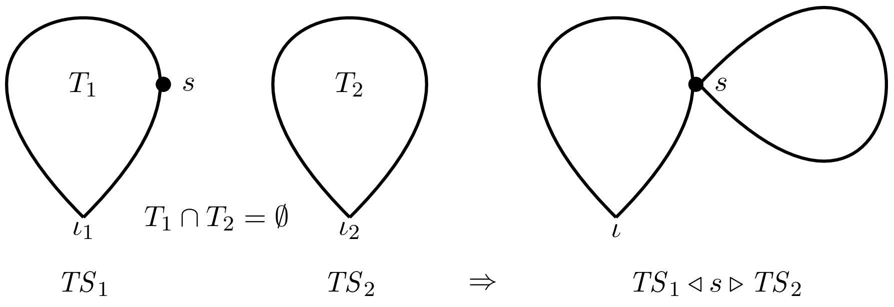


Figure 8. General idea of articulations.

Definition 4.1. ARTICULATION OF TWO DISJOINT *LTS*
Let $TS_1 = (S_1, \rightarrow_1, T_1, \iota_1)$ and $TS_2 = (S_2, \rightarrow_2, T_2, \iota_2)$ be two (totally reachable and deterministic) LTSs with $T_1 \cap T_2 = \emptyset$ and $s \in S_1$. Thanks to isomorphisms we may assume that $S_1 \cap S_2 = \{s\}$ and $\iota_2 = s$. We shall then denote by $TS_1 \lhd s \rhd TS_2 = (S_1 \cup S_2, T_1 \cup T_2, \rightarrow_1 \cup \rightarrow_2, \iota_1)$ the *articulation* of TS_1 and TS_2 around s.

Conversely, let $TS = (S, \rightarrow, T, \iota)$ be a (totally reachable and deterministic) LTS. We shall say that a label t is useful if $\exists s, s' \in S : s[t\rangle s'$. Let $\emptyset \subseteq T_1 \subseteq T$; we shall then denote by $adj(T_1) = \{s \in S | \exists t \in T_1 : s[t\rangle \text{ or } [t\rangle s\}$ if there are useful labels in T_1, $\{\iota\}$ otherwise. This is the *adjacency set* of T_1, i.e., the set of states connected to T_1 (with the convention that, if T_1 is empty or only contains useless labels, the result is the singleton initial state). Let $T_2 = T \setminus T_1$ and $s \in S$. We shall say that TS is *articulated*[2] by T_1 and T_2 around s if $adj(T_1) \cap adj(T_2) = \{s\}$, $\forall s_1 \in adj(T_1) \exists \alpha_1 \in T_1^* : \iota[\alpha_1\rangle s_1$ and $\forall s_2 \in adj(T_2) \exists \alpha_2 \in T_2^* : s[\alpha_2\rangle s_2$. □

[2]This notion has some similarity with the cut vertices (or articulation points) introduced for connected unlabelled undirected graphs, whose removal disconnects the graph. They have been used for instance to decompose such graphs into biconnected components [15, 16]. Note however that here we have labelled graphs.

This operator is only defined up to isomorphism since we may need to rename the state sets (usually the right one, but we may also rename the left one, or both). The only constraint is that, after the relabellings, s is the unique common state of TS_1 and TS_2, and is the state where the two systems are to be articulated. Figure 9 illustrates this operator. It also shows that the articulation highly relates on the state around which the articulation takes part. It may also be observed that, if $TS_0 = (\{\iota\}, \emptyset, \emptyset, \iota)$ is the trivial empty LTS, we have that, for any state s of TS, $TS \lhd s \rhd TS_0 \equiv TS$, i.e., we have a kind of right neutral trivial articulation. Similarly, $TS_0 \lhd \iota \rhd TS \equiv TS$, i.e., we have a kind of left neutral trivial articulation. However, these neutrals will play no role in the following of this paper, so that we shall exclude them from our considerations (and assume the edge label sets to be non-empty, and only composed of useful labels).

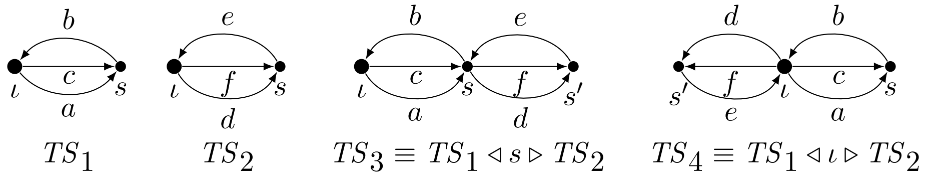


Figure 9. Some articulations.

Several easy but interesting properties may be derived for this articulation operator [10].

Proposition 4.2. (Both forms of articulation are equivalent)
If $TS = (S, \rightarrow, T, \iota)$ is articulated by T_1 and T_2 around s, then with $\rightarrow_1 = \rightarrow \cap adj(T_1) \times T_1 \times adj(T_1)$ (i.e., the restriction of $\rightarrow$ to T_1) and similarly for $\rightarrow_2$, the structures $TS_1 = (adj(T_1), \rightarrow_1, T_1, \iota)$ and $TS_2 = (adj(T_2), \rightarrow_2, T_2, s)$ are totally reachable LTSs and $TS \equiv_{T_1 \uplus T_2} TS_1 \lhd s \rhd TS_2$ (in that case we do not need to apply a relabelling to TS_1 and TS_2).

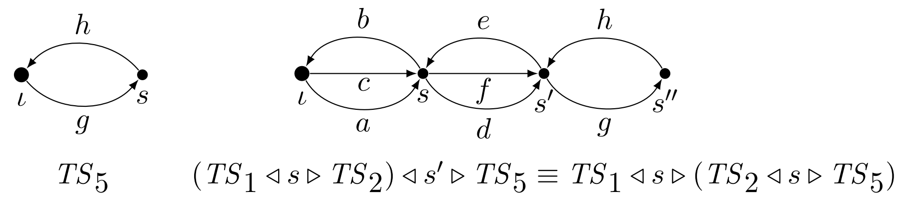
Conversely, $TS_1 \lhd s \rhd TS_2$ is articulated by the label sets of TS_1 and TS_2 around s.

Proposition 4.3. (Evolutions of an articulation)
If $TS \equiv TS_1 \lhd s \rhd TS_2$, $\iota[\alpha\rangle s'$ is an evolution of TS iff it is an alternation of evolutions of TS_1 and TS_2 separated by occurrences of s, i.e., either $\alpha \in T_1^*$ or $\alpha = \alpha_1\alpha_2 \ldots \alpha_n$ such that $\alpha_i \in T_1^*$ if i is odd, $\alpha_i \in T_2^*$ if i is even, $\iota[\alpha_1\rangle s$ and $\forall i \in \{1, 2, \ldots, n-1\} : [\alpha_i\rangle s[\alpha_{i+1}\rangle$.

For instance, for TS_3 in Figure 9, a possible evolution is $\iota[abc\rangle s[fede\rangle s[b\rangle\iota$, but also equivalently $\iota[a\rangle s[\varepsilon\rangle s[bc\rangle s[fe\rangle s[\varepsilon\rangle s[de\rangle s[b\rangle\iota$ (where ε is the empty sequence).

Proposition 4.4. (Associativity of articulations)
Let us assume that TS_1, TS_2 and TS_3 are three LTSs with label sets T_1, T_2 and T_3 respectively, pairwise disjoint. Let s_1 be a state of TS_1 and s_2 be a state of TS_2. Then, $TS_1 \lhd s_1 \rhd (TS_2 \lhd s_2 \rhd TS_3) \equiv_{T_1 \cup T_2 \cup T_3} (TS_1 \lhd s_1 \rhd TS_2) \lhd s_2' \rhd TS_3$, where s_2' corresponds in $TS_1 \lhd s_1 \rhd TS_2$ to s_2 in TS_2 (let us recall that the articulation operator may rename the states of the second operand).

This is illustrated by Figure 10.



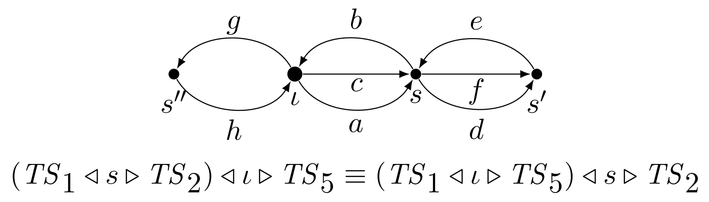
h
ι
s
g
TS_5
b
e
h
c
f
ι
s
s′
s″
a
d
g
$(TS_1 \lhd s \rhd TS_2) \lhd s' \rhd TS_5 \equiv TS_1 \lhd s \rhd (TS_2 \lhd s \rhd TS_5)$

Figure 10. Associativity of articulations.

Proposition 4.5. (Commutative articulations)
If $TS_1 = (S_1, \rightarrow_1, T, \iota_1)$ and $TS_2 = (S_2, \rightarrow_2, T, \iota_2)$ with disjoint label sets (i.e., $T_1 \cap T_2 = \emptyset$), then $TS_1 \lhd \iota_1 \rhd TS_2 \equiv_{T_1 \cup T_2} TS_2 \lhd \iota_2 \rhd TS_1$.

Note that, in the left member of the equivalence, we must rename ι_2 into ι_1, and in the right member we must rename ι_1 into ι_2, in order to apply Definition 4.1 (defined up to isomorphisms). For instance, in Figure 9, $TS_4 \equiv TS_1 \lhd \iota \rhd TS_2 \equiv TS_2 \lhd \iota \rhd TS_1$.

Proposition 4.6. (Commutative associativity of articulations)
Let us assume that TS_1, TS_2 and TS_3 are three LTSs with label sets T_1, T_2 and T_3 respectively, pairwise disjoint. Let s_2 and s_3 be two states of TS_1 ($s_2 = s_3$ is allowed). Then, $(TS_1 \lhd s_2 \rhd TS_2) \lhd s_3 \rhd TS_3 \equiv_{T_1 \cup T_2 \cup T_3} (TS_1 \lhd s_3 \rhd TS_3) \lhd s_2 \rhd TS_2$.



g
b
e
c
f
s″
ι
s
s′
h
a
d
$(TS_1 \lhd s \rhd TS_2) \lhd \iota \rhd TS_5 \equiv (TS_1 \lhd \iota \rhd TS_5) \lhd s \rhd TS_2$

Figure 11. Commutative associativity of articulations.

Proposition 4.7. (Sequence articulations)
If $TS_1 = (S_1, \rightarrow_1, T, \iota_1)$ and $TS_2 = (S_2, \rightarrow_2, T, \iota_2)$ with disjoint label sets (i.e., $T_1 \cap T_2 = \emptyset$), if $\forall s_1 \in S_1 \exists \alpha_1 \in T_1^* : s_1[\alpha_1\rangle s$ (s is a *home state in* TS_1) and $\nexists t_1 \in T_1 : s[t_1\rangle$ (s is a *dead end in* TS_1), then $TS_1 \lhd s \rhd TS_2$ behaves like a sequence, i.e., once TS_2 has started it is no longer possible to execute T_1.

The same occurs when ι_2 does not occur in a non-trivial cycle, i.e., $\iota_2[\alpha_2\rangle\iota_2 \wedge \alpha_2 \in T_2^* \Rightarrow \alpha_2 = \varepsilon$: once TS_2 has started it is no longer possible to execute T_1.

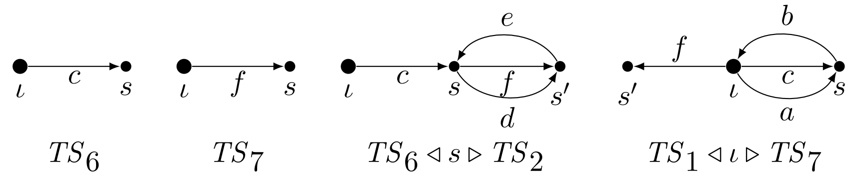


Figure 12. Sequential articulations.

This is illustrated in Figure 12. It may be observed that sequences in [17] (and in Figure 2) correspond to the intersection of both cases.

Let us now examine the connection between articulations and Petri net synthesis.

Proposition 4.8. (Synthesis of components of an articulation)
If $TS = (S, \rightarrow, T_1 \uplus T_2, \iota)$ is articulated by T_1 and T_2 around s, so that $TS \equiv TS_1 \triangleleft s \triangleright TS_2$ with $TS_1 = (adj(T_1), \rightarrow_1, T_1, \iota)$ and $TS_2 = (adj(T_2), \rightarrow_2, s)$ (see Proposition 4.2), and is PN-solvable, component TS_1 and TS_2 are also PN-solvable. Moreover, in the corresponding solution for TS_1, if the decomposition is not trivial, the marking corresponding to s is not dominated by any other reachable marking.

Proof:
Let $N = (P, T, F, M_0)$ be a solution for TS. It is immediate that $N_1 = (P, T_1, F_1, M_0)$, where F_1 is the restriction of F to T_1, is a solution for TS_1 (but there may be many other ones).

Similarly, if M is the marking of N (and N_1) corresponding to s, it may be seen that $N_2 = (P, T_2, F_2, M)$, where F_2 is the restriction of F to T_2, is a solution for TS_2 (but there may be many other ones).

Moreover, if the decomposition is not trivial, $T_2 \neq \emptyset$. Let us thus assume that $s[t_2\rangle$ for some label $t_2 \in T_2$ and M' is a marking of N_1 corresponding to some state s' in TS_1 with $M' \gneqq M$, then $s \neq s'$, $s'[t_2\rangle$ and s is not the unique articulation between T_1 and T_2. □

Note that there may also be solutions to TS_1 (other than N_1) such that the marking M corresponding to s is dominated. This is illustrated in Figure 13.

The other way round, let us now assume that $TS = TS_1 \triangleleft s \triangleright TS_2$ is an articulated LTS and that it is possible to solve TS_1 and TS_2. Is it possible from that to build a solution of TS?

To do that, we shall add the constraint already observed in Proposition 4.8 that, in the solution of TS_1, the marking corresponding to s is not dominated by another reachable marking. If this is satisfied we shall say that the solution is *adequate* with respect to s. Hence, in the treatment of the system in Figure 13, we want to avoid considering the solution N'_1 of TS_1; on the contrary, N_1 or N''_1 will be acceptable.

If TS_1 is reversible and PN-solvable, any solution is adequate. Indeed, for any pair of distinct states $s, s' \in S_1$ we then have a path $s[\alpha\rangle s'$ with $\alpha \in T_1^*$. In the solution PN_1 of TS_1, if M is the marking corresponding to s and M' is the one corresponding to s', we have $M \neq M'$ and $M[\alpha\rangle M'$ and if $M \lneqq M'$ we also have an infinite path $M'[\alpha^\infty\rangle$. Since PN_1 is a solution of TS_1, we also have $s[\alpha^n\rangle s_i$ for an infinite series of different states s_i for $n \in \mathbb{N}$, and TS_2 as well as TS may not be finite as we assumed in this paper. Note that, from a similar argument, since TS_2 is finite, no marking reachable in PN_2 dominates the initial one, corresponding to $\iota_2 = s$.

However, if TS_1 is solvable, it is always possible to get a solution adequate at s, as for any state in fact.

Proposition 4.9. (Adequate solutions)
Let TS_1 be a (finite) solvable *LTS* and s any of its states. Then there is a solution of TS_1 adequate at s.

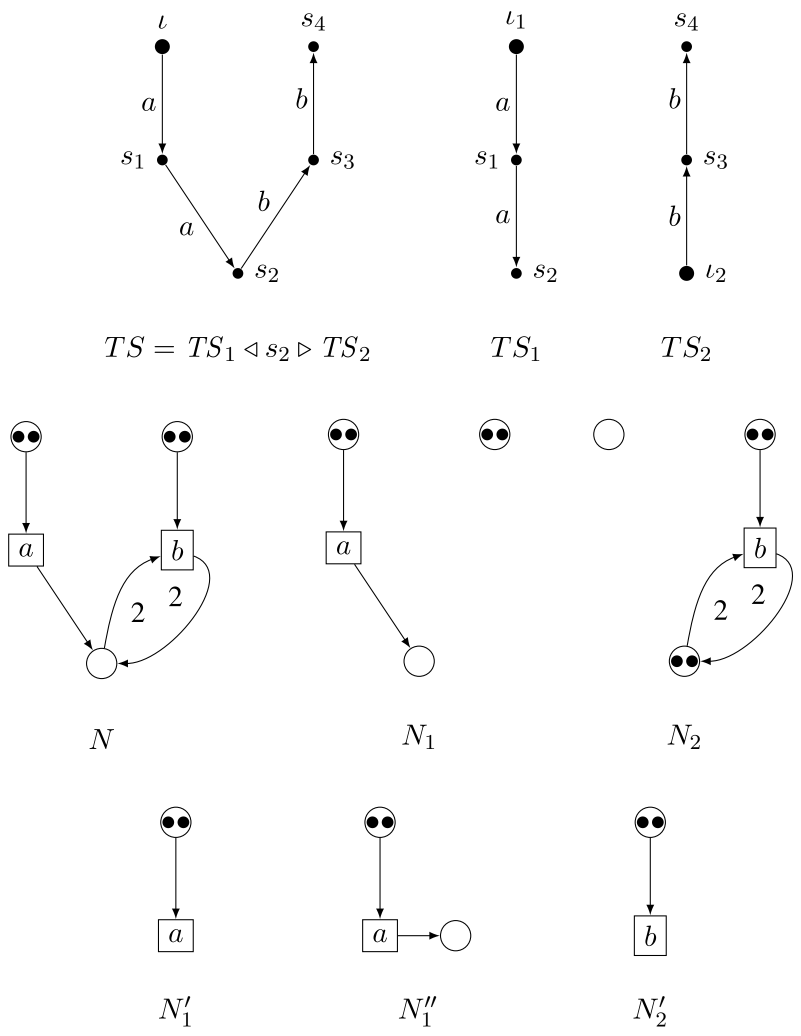


Figure 13. The lts TS is articulated around s_2, with $T_1 = \{a\}$ and $T_2 = \{b\}$, hence leading to TS_1 and TS_2. It is solved by N, and the corresponding solutions for TS_1 and TS_2 are N_1 and N_2, respectively. TS_1 also has the solution N'_1 but the marking corresponding to s_2 is then empty, hence it is dominated by the initial marking (as well as by the intermediate one). This is not the case for the other solution N''_1 (obtained from N_1 by erasing the useless isolated place: we never claimed that N_1 is a minimal solution). TS_2 also has the solution N'_2.

Proof:
Let PN_1 be any solution. For any place $p \in P_1$, we may construct a complement or mirror place $\widetilde{p}$ such that $\forall t \in T_1 : F(t, \widetilde{p}) = F(p, t) \wedge F(\widetilde{p}, t) = F(t, p)$ so that, for any reachable marking $\widetilde{M}$ of the new net, $\widetilde{M}(p) + \widetilde{M}(\widetilde{p})$ is constant. It remains to chose the initial marking of this place in order not to exclude some evolutions available in PN_1. Since TS_1 is finite, the marking of p is bounded (as for any other place). Let k be that bound and let us chose $\widetilde{M_0}(p) = k - M_0(p) + \max_{t \in T_1} F(t, p)$. That way, for any reachable marking, $\widetilde{M}(\widetilde{p}) \geq \max_{t \in T_1} F(t, p) = \max_{t \in T_1} F(\widetilde{p}, t)$, so that place $\widetilde{p}$ does

not block any transition that would be enabled by place p. We may thus conclude that the introduction of $\widetilde{p}$ does not change the reachability graph and the new net is still a PN-solution of TS_1. As a consequence, for each reachable marking M of PN_1, if $M(p) > M_s(p)$ (where M_s is the marking corresponding to s), $\widetilde{M}(\widetilde{p}) < \widetilde{M_s}(\widetilde{p})$. Hence, if we introduce a complement place for each place of PN_1, in the new net no reachable marking may dominate another one and the constraint mentioned above is always satisfied. □

However there is a simpler way to get an adequate solution (since we know there is one), in the following way:

Proposition 4.10. (Forcing an adequate solution for TS_1)
Let us add to TS_1 an arc $s[u\rangle s$ where u is a new fresh label. Let TS'_1 be the LTS so obtained. If TS'_1 is not solvable, there is no (adequate) solution. Otherwise, solve TS'_1 and erase u from the solution. Let N_1 be the net obtained with the procedure just described: it is a solution of TS_1 with the adequate property that the marking corresponding to s is not dominated by another one.

Proof:
If there is an adequate solution N_1 of TS_1 (and from the reasoning above there is one iff there is a solution), with a marking M corresponding to s, let us add a new transition u to it with, for each place p of N_1, $W(p,u) = M(p) = W(u,p)$: the reachability graph of this new net is (isomorphic to) TS'_1 since u is enabled by marking M (or any larger one, but there is none) and does not modify the marking. Hence, if there is no (adequate) solution of TS_1, there is no solution of TS'_1.

Let us now assume there is a solution N'_1 of TS'_1. The marking M corresponding to s is not dominated otherwise there would be a loop $M'[s\rangle M'$ elsewhere in the reachability graph of N'_1, hence also in TS'_1. Hence, dropping u in N'_1 will lead to an adequate solution of TS_1. □

For instance, when applied to TS_1 in Figure 13, this will lead to N''_1, and not N'_1 (N_1 could also be produced, but it is likely that a 'normal' synthesis tool will not construct the additional isolated place).

Now, to understand how one may generate a solution for TS from the ones obtained for TS_1 and TS_2, we may first carefully examine the example illustrated in Figure 14: some side conditions (i.e., pairs of place-transition with arcs going both ways, with identical weights) occur in the global solution. This leads to the following construction.

Definition 4.11. ARTICULATION OF PETRI NETS
Let $PN_1 = (P_1, T_1, F_1, M_0^1)$ and $PN_2 = (P_2, T_2, F_2, M_0^2)$ be two disjoint bounded Petri net systems and M a reachable marking of PN_1 not dominated by another one. $PN_1 \lhd M \rhd PN_2$ is the Petri net built first by putting side by side PN_1 and PN_2.

Then, for each transition t_1 enabled at M in PN_1, and each place $p_2 \in P_2$ such that $M_0^2(p_2) > 0$, create a side condition $F(t_1, p_2) = F(p_2, t_1) = M(p_2)$. For each transition t_2 initially enabled in PN_2, and each place $p_1 \in P_1$ such that $M(p_1) > 0$, create a side condition $F(t_2, p_1) = F(p_1, t_2) = M(p_1)$.

No other modification is afforded. □

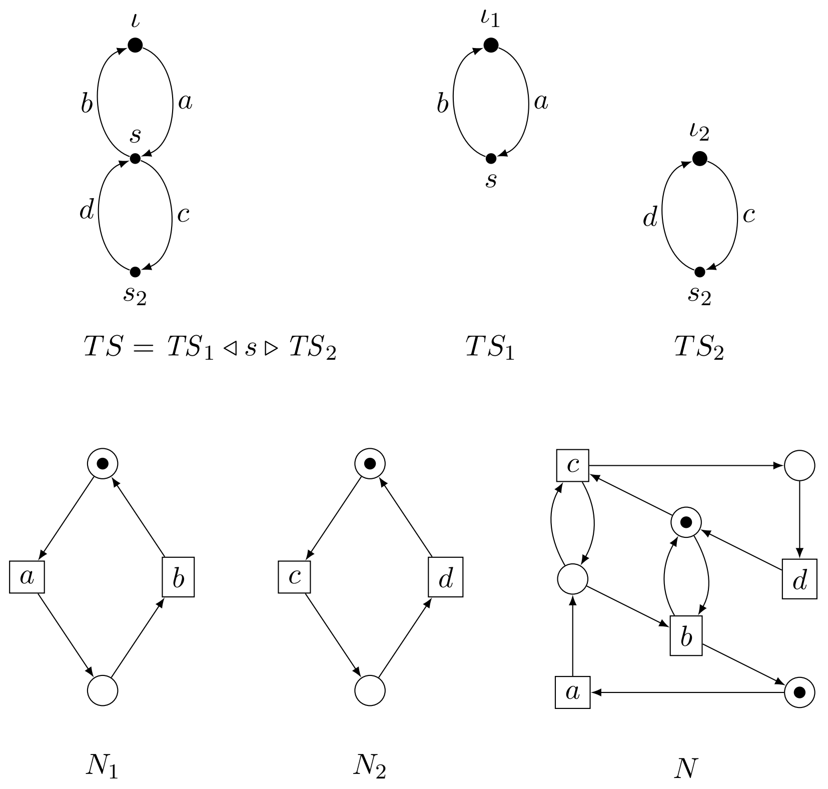


Figure 14. The lts TS is articulated around s, with $T_1 = \{a, b\}$ and $T_2 = \{c, d\}$, hence leading to TS_1 and TS_2. It is solved by N, and the corresponding solutions for TS_1 and TS_2 are N_1 and N_2, respectively. In N, we may recognise N_1 and N_2, connected by two kinds of side conditions: the first one connects the label b out of s in TS_1 to the initial marking of N_2, the other one connects the label c out of ι_2 in TS_2 to the marking of N_1 corresponding to s.

Proposition 4.12. (Synthesis of articulation)
Let $TS = TS_1 \lhd s \rhd TS_2$. If TS_1 or TS_2 are not solvable, so is TS.
Otherwise, let PN_1 be a solution of TS_1 adequate at s, i.e., such that the marking M_s corresponding to s is not dominated by another reachable marking, and let PN_2 be a disjoint solution of TS_2. Then the net $PN_1 \lhd M_s \rhd PN_2$ is a solution of TS.

Proof:
The property arises from Proposition 4.8 and the observation that PN_1 with the additional side conditions behaves like the original PN_1 if PN_2 is in M_0^2 and PN_2 with the additional side conditions behaves like the original PN_2 if PN_1 is in M_s. When PN_1 is not in M_s, (modified) PN_2 may not leave M_0^2 since M_s is not dominated in PN_1. When PN_1 reaches M_s, (modified) PN_2 may start its job but then (modified) PN_1 may not leave M_s until PN_2 returns to M_0^2 since the latter is not dominated in PN_2. When PN_2 is in M_0^2, (modified) PN_1 may leave M_s but then (modified) PN_2 may not leave M_0^2 since M_s is not dominated in PN_1. The evolutions of the constructed net thus correspond exactly to what is described by TS. $\square$

Note that we do not claim this is the only solution, but the goal is to find a solution when there is one.

It remains to show when and how an LTS may be decomposed by a non-trivial articulation (or several ones). Let us thus consider some LTS $TS = (S, \rightarrow, T, \iota)$. We may assume it is finite, totally reachable, deterministic and weakly live (there is no useless label).

First, we may observe that, for any two distinct labels $t, t' \in T$, if $|adj(\{t\}) \cap adj(\{t'\})| > 1$, t and t' must belong to the same subset for defining an articulation. Let us extend the function adj to non-empty subsets of labels by stating $adj(T') = \cup_{t \in T'} adj(t)$ when $\emptyset \subset T' \subset T$. We then have that, if $\emptyset \subset T_1, T_2 \subset T$ and we know that all the labels in T_1 must belong to the same subset for defining an articulation, and similarly for T_2, $|adj(T_1) \cap adj(T_2)| > 1$ implies that $T_1 \cup T_2$ must belong to the same subset of labels defining an articulation. If we get the full set T, that means that there is no possible articulation (but the trivial one).

Hence, starting from any partition $\mathcal{T}$ of T (initially, if $T = \{t_1, t_2, \ldots, t_n\}$, we shall start from the finest partition $\mathcal{T} = \{\{t_1\}, \{t_2\}, \ldots, \{t_n\}\}$), we shall construct the finest partition compatible with the previous rule:

while there is $T_1, T_2 \in \mathcal{T}$ **such that** $T_1 \neq T_2$ **and** $|adj(T_1) \cap adj(T_2)| > 1$**, replace** T_1 **and** T_2 **in** $\mathcal{T}$ **by** $T_1 \cup T_2$.

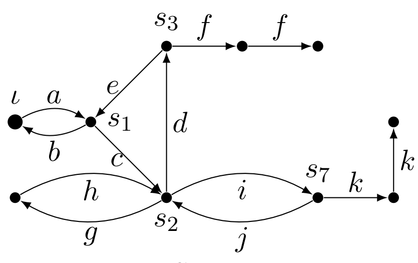
At the end, if $\mathcal{T} = \{T\}$, we may stop with the result: *there is no non-trivial articulation.*

Otherwise, we may define a finite bipartite undirected graph whose nodes are the members of the partition $\mathcal{T}$ and some states of S, such that if $T_i, T_j \in \mathcal{T}, T_i \neq T_j$ and $adj(T_i) \cap adj(T_j) = \{s\}$, there is a node s in the graph, connected to T_i and T_j (and this is the only reason to have a state as a node of the graph). Since TS is weakly live and totally reachable, this graph is connected, and each state occurring in it has at least two neighbours (on the contrary, a subset of labels may be connected to a single state). Indeed, since TS is weakly live, $\cup_{T' \in \mathcal{T}} adj(T') = S$. Each state s occurring as a node in the graph is connected to at least two members of the $\mathcal{T}$, by the definition of the introduction of s in the graph. Let T_1 be the member of $\mathcal{T}$ such that $\iota \in adj(T_1)$, let T_i be any other member of $\mathcal{T}$, and let us consider a path $\iota[\alpha\rangle$ ending with some $t \in T_i$ (we may restrict our attention to a short such path, but this is not necessary): each time there is a sequence $t't''$ in α such that t' and t'' belong to two different members T' and T'' of $\mathcal{T}$, we have $[t'\rangle s[t''\rangle$, where s is the only state-node connected to T' and T'', hence in the graph we have $T' \rightarrow s \rightarrow T''$. This will yield a path in the constructed graph going from T_1 to T_i, hence the connectivity.

If there is a cycle in this graph, that means that there is no way to group the members of $\mathcal{T}$ in this cycle in two subsets such that the corresponding adjacency sets only have a single common state. Hence we need to fuse all these members, for each such cycle, leading to a new partition, and we also need to go back to the refinement of the partition in order to be compatible with the intersection rule, and to the construction of the graph.

Finally, we shall get an acyclic graph G, with at least three nodes (otherwise we stopped the articulation algorithm with the information that there is no non-trivial decomposition).

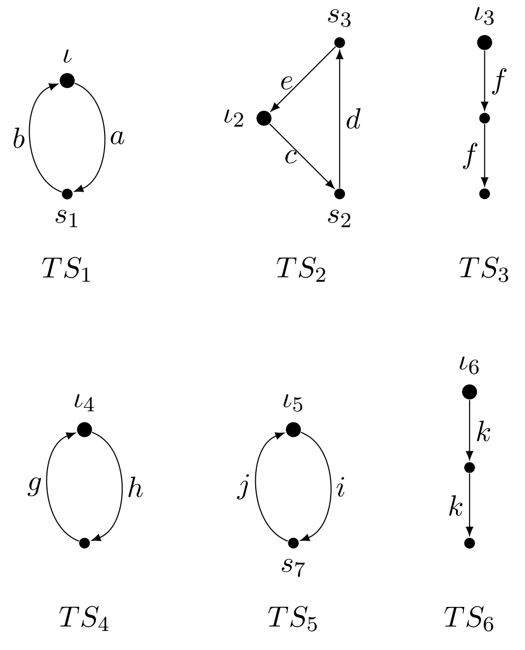
We shall now define a procedure $articul(SG)$ that builds an LTS expression based on articulations from a subgraph SG of G with a chosen state-node root. We shall then apply it recursively to G, leading finally to an articulation-based (possibly complex) expression equivalent to the original LTS TS.



TS

$$\begin{tikzcd}
 & & s_3 \arrow[r, dash] \arrow[d, dash] & \{f\} & & \\
\{a,b\} \arrow[r, dash] & s_1 \arrow[r, dash] & \{c,d,e\} \arrow[r, dash] & s_2 \arrow[r, dash] \arrow[d, dash] & \{h,g\} & \\
 & & & \{i,j\} \arrow[r, dash] & s_7 \arrow[r, dash] & \{k\}
\end{tikzcd}$$

G



$$TS \equiv TS_1 \lhd s_1 \rhd (((TS_2 \lhd s_3 \rhd TS_3) \lhd s_2 \rhd TS_4) \lhd s_2 \rhd (TS_5 \lhd s_7 \rhd TS_6))$$

Figure 15. The lts TS leads to the graph G. The corresponding components are TS_1 to TS_6, which may easily be synthesised; note that, from the total reachability of TS, they are all totally reachable themselves. This leads to the articulated expression below.

The basic case will be that, if SG is a graph composed of a state s connected to a subset node T_i, $articul(SG)$ will be the LTS $TS_i = (adj(T_i), T_i, \to_i, s)$ (as usual $\to_i$ is the projection of $\to$ on T_i; by construction, it will always be the case that $s \in adj(T_i)$).

First, if ι is a state-node of the graph, G then has the form of a star with root ι and a set of satellite subgraphs $G_1, G_2, ..., G_n$ (n is at least 2). Let us denote by SG_i the subgraph with root ι connected to G_i: the result will then be the (commutative, see Proposition 4.5) articulation around ι of all the LTSs $articul(SG_i)$.

Otherwise, let T_1 be the (unique) label subset in the graph such that $\iota \in adj(T_1)$. G may then be considered as a star with T_1 at the center, surrounded by subgraphs SG_1, SG_2, ..., SG_n (here n may be 1), each one with a root s_i connected to T_1 (we have here that $s_i \in adj(T_1)$, and we allow $s_i = s_j$): the result is then $((\ldots((adj(T_1), T_1, \to_1, \iota) \lhd s_1 \rhd articul(SG_1)) \lhd s_2 \rhd articul(SG_2)) \ldots) \lhd s_n \rhd articul(SG_n))$. Note that, if $n > 1$, the order in which we consider the subgraphs is irrelevant from Proposition 4.6.

Finally, if a subgraph starts from a state s', followed by a subset T', itself followed by subgraphs SG_1, SG_2, ..., SG_n ($n \geq 1$; if it is 0 we have the base case), each one with a root s_i connected to T' (we have here that $s' \in adj(T')$, and we allow $s_i = s_j$): the result is then $((\ldots((adj(T'), T', \to', s') \lhd s_1 \rhd articul(SG_1)) \lhd s_2 \rhd articul(SG_2)) \ldots) \lhd s_n \rhd articul(SG_n))$. Again, if $n > 1$, the order in which we consider the subgraphs is irrelevant from Proposition 4.6.

This procedure is illustrated in Figure 15.

Contrary to what happened for the product of nets, articulations do not preserve many Petri net subclasses. For instance, if PN_1 and PN_2 are plain (no arc weight greater than 1), $PN_1 \lhd M \rhd PN_2$ is not plain (unless M and M_0^2 do not have more than one token in any place); however, since we may have many solutions to a synthesis problem, it may happen that $PN_1 \lhd M \rhd PN_2$ is not safe but that another solution of the same problem is safe. On the contrary, an immediate consequence of Definition 4.11 is that

Corollary 4.13. (Bound preservation)
$PN_1 \lhd M \rhd PN_2$ is safe iff so are PN_1 and PN_2. If $PN_1 \lhd M \rhd PN_2$ is k-safe, so are PN_1 and PN_2; finally, if PN_1 is k_1-safe and PN_2 is k_2-safe, then $PN_1 \lhd M \rhd PN_2$ is $\max(k_1, k_2)$-safe.

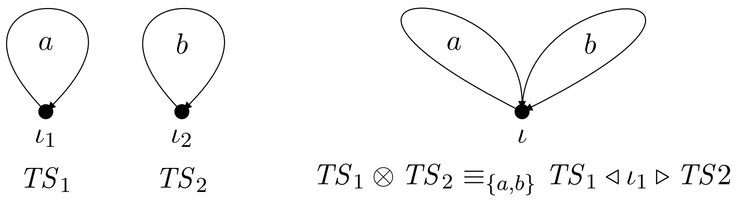
5. Mixed decomposition

In the previous sections we have introduced two pairs of (families of) operators acting on transition systems and Petri net systems: $TS_1 \otimes TS_2$ - $PN_1 \oplus PN_2$ and $TS_1 \lhd s \rhd TS_2$ - $PN_1 \lhd M \rhd PN_2$.

They may be intermixed, as exemplified in Figure 2, where $TS = TS(start); (TS(a) \otimes TS(b)); TS(end)$ may be rewritten $TS = TS(start) \lhd s_1 \rhd ((TS(a) \otimes TS(b)) \lhd s_4 \rhd TS(end))$.

We may wonder however if there are cases where a transition system may be decomposed both as a (non-trivial) product and as an (non-trivial) articulation. In the following, we shall assume there is no useless label (i.e., each label occurs at least once in a transition), and that the transition systems are totally reachable (otherwise there is no Petri net solution). If T is a set of labels, we shall also denote by ι_T the transition system with a single state ι and, for each $t \in T$, a loop $\iota[t\rangle\iota$ (up to isomorphism, it is the only transition system with only one state and label set T, without useless label).

First, we may have $TS \equiv TS_1 \otimes TS_2 \equiv TS_1 \triangleleft s_1 \triangleright TS_2$, but only if TS, TS_1, TS_2 have a single state, as illustrated by Figure 16.



a
b
ι_1
ι_2
TS_1
TS_2
a
b
ι
$TS_1 \otimes TS_2 \equiv_{\{a,b\}} TS_1 \triangleleft \iota_1 \triangleright TS2$

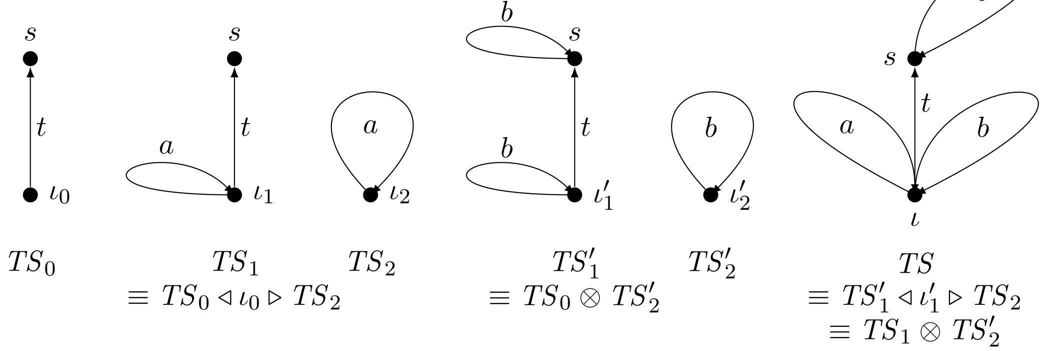
Figure 16. Singleton case.

Proposition 5.1. (Equivalent decompositions (first case))
$TS \equiv TS_1 \otimes TS_2 \equiv TS_1 \triangleleft s_1 \triangleright TS_2$, with $|T_1| > 0 < |T_2|$, iff $|S_1| = 1 = |S_2|$ (hence also $|S| = 1$ and $s_1 = \iota_1$).

Proof:
If $\{T_1, T_2\}$ is a partition of T, we have that $\iota_T \equiv \iota_{T_1} \otimes \iota_{T_2} \equiv \iota_{T_1} \triangleleft \iota \triangleright \iota_{T_2}$.

Let us now assume that $TS \equiv TS_1 \otimes TS_2 \equiv TS_1 \triangleleft s_1 \triangleright TS_2$. We must have $|S| = |S_1| \cdot |S_2| = |S_1| + |S_2| - 1$, so that $(|S_1| - 1) \cdot (|S_2| - 1) = 0$ and $|S_1| = 1$ or $|S_2| = 1$. Let us assume that $|S_2| > 1$ (the case $|S_1| > 1$ is symmetrical): there must be $s \neq s' \in S_2$ and $t \in T_2$ such that $(s, t, s') \in \to_2$. Since $|S_1| = 1$ and it is assumed there is no useless label, $TS_1 \equiv \iota_{T_1}$ for some partition $\{T_1, T_2\}$ of T. In $TS_1 \otimes TS_2$, we have $(\iota, s)[t\rangle(\iota, s')$, $(\iota, s)[a\rangle(\iota, s)$ and $(\iota, s')[a\rangle(\iota, s')$ for any $a \in T_1$. Hence $(\iota, s) \neq (\iota, s') \in adj(t) \cap adj(a)$ and $t, a \in T_2$ in $TS_1 \triangleleft \iota \triangleright TS_2$, contradicting the fact that $\{T_1, T_2\}$ is a partition of T. Hence, we must have $|S_1| = |S_2| = 1$. $\square$

In terms of synthesis, the gain of decomposing TS is not very high in this case, since there is an easy solution (composed of isolated transitions).



s
t
ι_0
TS_0
s
a
t
ι_1
a
ι_2
TS_1
TS_2
$\equiv TS_0 \triangleleft \iota_0 \triangleright TS_2$
b
s
b
t
ι'_1
b
ι'_2
TS'_1
TS'_2
$\equiv TS_0 \otimes TS'_2$
b
s
t
a
b
ι
TS
$\equiv TS'_1 \triangleleft \iota'_1 \triangleright TS_2$
$\equiv TS_1 \otimes TS'_2$

Figure 17. Ambiguous case.

There are special cases, however, where we have a choice between a product and an articulation decomposition with non-singleton state spaces, but with different partitions of the label set, as illustrated by Figure 17. This only occurs however when one of the components is a singleton system.

Proposition 5.2. (Equivalent decompositions (second case))
If $TS \equiv TS_1 \otimes TS_2 \equiv TS'_1 \lhd s' \rhd TS'_2$, then $|S_1| = 1$ or $|S_2| = 1$ as well as $|S'_1| = 1$ or $|S'_2| = 1$.

Proof:
Let us first assume that $|S_1| > 1 < |S_2|$. In TS_1, there are $s_1 \neq s'_1 \in S_1$ and $a_1 \in T_1$ with $(s_1, a_1, s'_1) \in \rightarrow_1$. For any $a_2 \in T_2$, since there is no useless label, we have $(s_2, a_2, s'_2) \in \rightarrow_2$ (here we allow $s_2 = s'_2$). In $TS_1 \otimes TS_2$, we then have that $(s_1, s_2)[a_1\rangle(s'_1, s_2)$, $(s_1, s'_2)[a_1\rangle(s'_1, s'_2)$ $(s_1, s_2)[a_2\rangle(s_1, s'_2)$ and $(s'_1, s_2)[a_2\rangle(s'_1, s'_2)$, so that $|adj(a_1) \cap adj(a_2)| > 1$ and, in any articulation decomposition of TS, a_1 must belong to the same component as any $a_2 \in T_2$. Symmetrically, some a_2 in T_2 must belong to the same component as any $a_1 \in T_1$ in any articulation of TS. As a consequence, any articulation of TS must be trivial (all the labels belong to the same component).

Let us now assume that $|S'_1| > 1 < |S'_2|$. From the previous point, we may not have $|S_1| > 1 < |S_2|$; without loss of generality, we shall assume $|S_1| = 1 < |S_2| = |S|$. Let $a_1 \in T_1$ (there must be one, since we assumed $T_1 \neq \emptyset$); then a_1 must occur as a loop around each state of $TS \equiv TS_1 \otimes TS_2$. Hence a_1 is adjacent to any state and we may only have $TS \equiv TS'_1 \lhd s' \rhd TS'_2$ if there is a single component in the articulation, contradicting $|S'_1| > 1 < |S'_2|$. □

Corollary 5.3. (General ambiguous form)
The only cases where we have a choice between a product and an articulation of transition systems have the form (up to a permutation of the roles of the three components)

$$(TS_1 \otimes \iota_{T_2}) \lhd (s_1, \iota) \rhd \iota_{T_3} \equiv (TS_1 \lhd s_1 \rhd \iota_{T_3}) \otimes \iota_{T_2}$$

where T_2, T_3 and the label set of TS_1, are pairwise disjoint and s_1 is a state of TS_1.
It is Petri net solvable iff so is TS_1 and a possible solution is a solution of TS_1 adequate for s_1, plus one transition for each label of T_3 connected by a side condition to the places marked by the marking corresponding to s_1 (with weights given by this marking), plus one isolated transition for each label of T_2.

When we have a choice between a factorisation and an articulation, the former should probably be preferred since a synthesis may then be put in the form of a sum of nets. But anyway, since at least one of the components of the product and at least one of the components of the articulation have only one state, the gain for the synthesis is low. Let us recall however that another interest of these decompositions is to exhibit an internal structure for the examined transition system.

For components with at least two states (hence not only composed of loops), there is no ambiguity in the decomposition, but we may alternate products and articulations, using the procedures detailed at the end of Sections 3 and 4. At the end, we shall obtain non decomposable components, but also possibly a choice between a product and an articulation if we get singleton components as in the examples in Figures 16 and 17.

6. Experiments

Let us now examine more closely the gain in efficiency the decompositions explored in this paper afford for the Petri net synthesis. Several remarks may be done before we start true experiments.

First, it is better to distinguish positive cases (where there is a solution) from negative ones. Indeed, in a negative case, we may stop when a necessary condition is not satisfied or when a SSP or ESSP problem may not be solved. In each case, the performance highly relies on the order in which sub-problems are considered. And even if we pursue in order to find all the obstacles, it may happen that finding that a problem has no solution has not the same complexity than finding a solution when there is one. Hence, in the following, we shall only consider positive examples. In the same spirit we shall not try to find an optimal solution (for instance with a minimal number of places, or with small weights, etc.).

Next, there is $|S|\cdot(|S|-1)/2$ SSP problems and $|S|\cdot|T| - |\rightarrow|$ $(=\sum_{s\in S}(|T| - \text{outdegree}(s))$ ESSP problems (where the outdegree of a state is the number of arcs originated from it), possibly after some pre-synthesis checks.

The pre-synthesis checks allow to quickly reject inadequate transition systems, without needing expensive solutions to SSP and ESSP problems. This includes the checks for total reachability and determinism, but also other ones if we search for solutions in a specific subclass of net systems. For instance, if we search for choice-free solutions (i.e., nets where each place has at most one output transition), many necessary structural conditions have been developed for a pre-synthesis phase [4, 18]. For positive cases, the pre-synthesis phase is usually negligible with respect to the time and memory requested by the solution of ESSP and SSP problems, but may provide interesting informations and data-structures for solving those separation problems.

In large reachability graphs, the number of states is usually much larger than the number of transitions. For instance, if we increase the number of initial tokens, the size of the graph may increase in a dramatic manner, while the number of transitions does not change. This leads to consider complexities in terms of the number of states $|S|$ for the positive synthesis of large transition systems, and to neglect the impact of the number of labels $|T|$.

To solve a separation problem, we have to find an adequate region $(\rho, \mathbb{B}, \mathbb{F})$. The number of unknowns is $|S| + 2\cdot|T|$ and for each arc $s[t\rangle s' \in\rightarrow$ we have two linear constraints: $\rho(s) \geq \mathbb{B}(t)$ and $\rho(s') = \rho(s) + \mathbb{F}(a) - \mathbb{B}(a)$. For a SSP$(s, s')$, we have to add the constraint $\rho(s) \neq \rho(s')$, and for an ESSP(s, t) we have to add the constraint $\rho(s) < \mathbb{B}(a)$. Hence, we have to solve systems of homogeneous linear constraints of the same size, and the complexity CS of all the separation problems is about the same. This leads to a positive synthesis complexity of the kind $A\cdot|S|^2\cdot CS + E\cdot|S|\cdot CS$ for some coefficients A and B.

However, this is not correct in general. First, for the synthesis of some subclasses of Petri nets, it is known that SSPs are irrelevant: the regions (or places) built to solve the ESSPs also solve all the SSPs. This is the case for the choice-free nets [4], hence also for all their subclasses. But even for general Petri net syntheses, this leads to the idea to first solve the ESSP problems and, for each SSP problem, check first if there is no region built previously which already solves it, which is very quick since there are generally few needed regions and the check is easy. This has been used for instance in the tool APT [3], and the experience shows that most of the time, no new system of linear constraints has to be solved for SSP problems, and otherwise only a very small number of them is needed. The same is true for ESSP problems: instead of systematically building and solving the corresponding system of linear constraints, we may first check if one of the regions built previously does not already solve the new separation problem (the efficiency of the procedure then relies on the order in which ESSP problems are considered). This may be interpreted in the following way: the global complexity is of the kind $(A \cdot |S|^2 + E \cdot |S|) \cdot CS$; in many cases $A = 0$; if this is not true, it may happen that $A \cdot |S|^2$ does not grow faster than $E \cdot |S|$; and if it does, most of the time A is so small that the dominance of $A \cdot |S|^2$ only occurs for synthesis problems that are so huge that, anyway, the problem is out of practical reach, due to memory and/or execution time overflow problems. Note that $E \cdot |S|$ also may grow much less than linearly. This is difficult to evaluate theoretically beforehand and experiments may be useful for that.

For the synthesis of weighted Petri nets, each system of linear constraints to be solved is homogeneous[3] and we may use the Karmarkar algorithm [19] for that, which has a known worse case complexity polynomial in the size of the system (here linear in $|S|$). The exact polynomial exponent is difficult to determine, but it is usually expected to be around 5, hence the degree 7 mentioned in the introduction section, to solve a quadratic number of SSP problems, but we know now that we should expect a much smaller number of separation problems to be solved in the form of a system of linear constraints of linear size. Moreover, it is known that, in practice, the Karmarkar algorithm is outperformed by the simplex algorithm [20], which however is exponential in principle [21]. The same arises with classical SMT solvers (APT uses SMTInterpol [22, 23]). This seems to be due to the fact that the families of problems leading to an exponential growth are too artificial and are not met in true applications. So again, experiments would be welcome to clarify the complexity of tools like APT.

If we now add decomposition algorithms, like the two ones described in this paper, in the pre-synthesis phase, we may expect some more improvements. The decomposition algorithms are very quick with respect to the proper synthesis (see for instance [9]) so that, even if no non-trivial components are detected, the loss due to the addition of new work in the preliminary phase is negligible. If non-trivial components are found, a post-processing phase must be added to recompose the various synthesised Petri net systems, but this is again very quick and negligible with respect to the various intermediate syntheses. The improvement will be more noticeable if the components are approximatively of the same size.

Proposition 6.1. (Improvement due to decomposition techniques)
Let us assume that, for a family of transition systems with increasing size $|S|$, the complexity of a positive synthesis is approximately proportional to $|S|^h$ for some $h \in \mathbb{R}_{>0}$.

If the transition systems may be factorised in k factors of approximately the same size, the gain is about

$$\frac{|S|^{h(1-1/k)}}{k}$$

[3]i.e., there is no independent terms; in that case, finding a solution in the integer domain is equivalent to finding one in the rational domain, since it is always possible to multiply a rational solution by an adequate factor to get an integer one.

If the transition systems may be articulated in k components of approximately the same size, the gain is about

$$k^{h-1}$$

Proof:

In the first case, each factor has an approximate size of $\sqrt[k]{|S|}$, and there are k of them, so that the gain is

$$\frac{|S|^h}{k \cdot |S|^{h/k}}$$

which leads to the first formula.

In the second case, each factor has an approximate size of $|S|/k$, and there are k of them, so that the gain is

$$\frac{|S|^h}{k \cdot (|S|/k)^h}$$

which leads to the second formula. □

Hence, we shall consider families of examples with an increasing number of copies of the same transition system as components. In a product of n copies of systems with $|S|$ states, the state space grows (exponentially) as $|S|^n$ (the number of labels grows linearly: $|T| \cdot n$). In an articulation of n copies of systems with $|S|$ states, the state space grows (linearly) as $|S| \cdot n - n + 1$ (the number of labels still grows linearly). Contrary to the product cases where there is no variant possible in the way the components are combined, this is not true for the articulations, where we can chose various ways of plugging a new component on the previous member of the family, and it could happen that the performance of the chosen tool (here APT) behaves differently on the different families. We shall here consider three ways of performing the plugging: the star-shape, the daisy-flower shape and the caterpillar shape, schematised in Figure 18 with 4 components each. The first one articulates all the components around the initial state. The second one articulates the components around different states of the first component. The last one plugs each component but the first one on a non-initial state of the previous component. In each case the time used by the synthesis of the n components is linear and takes n times the time to synthesise one of them.

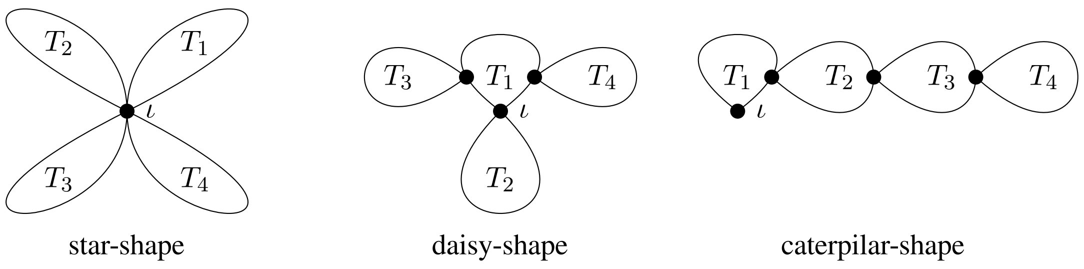


Figure 18. Three families of articulations.

Note however that it may happen that the performances are better than so expected. Indeed, when decomposing a complex transition system, it may happen that some (or all) components belong to a special case, to which the original system does not belong and for which a special implementation may speed up the synthesis. For instance, when decomposing a system into factors, if we get systems with a PN-solution which is a connected marked graph (i.e., each place has one input and one output transition, with weight 1), we may use the very effective synthesis described in [2]; the original system then also has a marked graph solution (an addition of marked graphs is a marked graph), but it is not connected and the speed up may not be applied. When decomposing a system with articulations, it may happen that some (or all) components have a choice-free solution, for which we know how to reduce the number and size [4] of ESSP problems to be solved (the SSP problems are then irrelevant). We shall not create such situations in our experiments however.

For the articulations, we shall use the system TS 1 in Figure 3, which has 23 states and no choice-free solution. We compared the obtained CPU time[4] for the three families mentioned above, up to 100 components. The results are schematised in Figure 19, where round dots are used for the star-shape

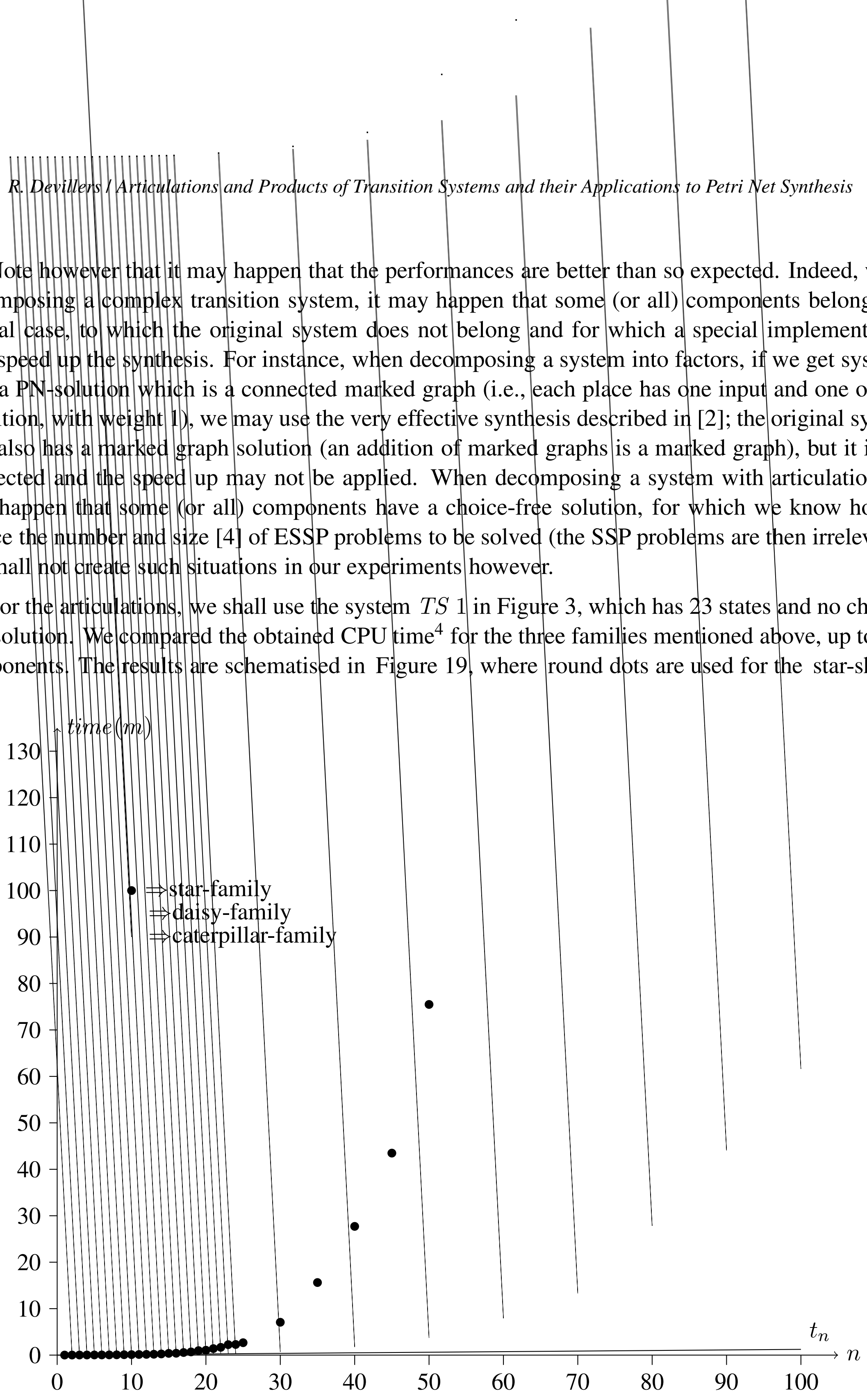


Figure 19. CPUtime for articulated families.

[4]The machine we used is based on processors Intel Xeon Gold 2.6 GHz

family, square dots for the daisy-shape family, and triangle dots for the caterpillar-shape family. It may be observed that the three families behave differently, while the systems have the same size for the same number of components. In particular, the star-shape family takes much more time, and beyond 50 components, APT crashed for memory exhaustion. From 50 components, the daisy-shape family takes more time than the caterpillar-shape one. Anyway, from $n = 30$ components, the performance is worse than synthesising n separate components (and recombining them, indicated by the line t_n; note however that if we use a computer system with several CPUs and/or cores, it is possible to launch several individual component syntheses in parallel, reducing seriously that indicator). Since the plot is crushed for the small numbers of components, we present in Figure 20 a zoom on the first results (here the time is given in seconds, instead of minutes). Curiously, up to 10 components (221 states) it is more effective to synthesise the articulated system than to desarticulate it and solve the components separately; and for the daisy and caterpillar families, this is still true up to 20 components (441 states).

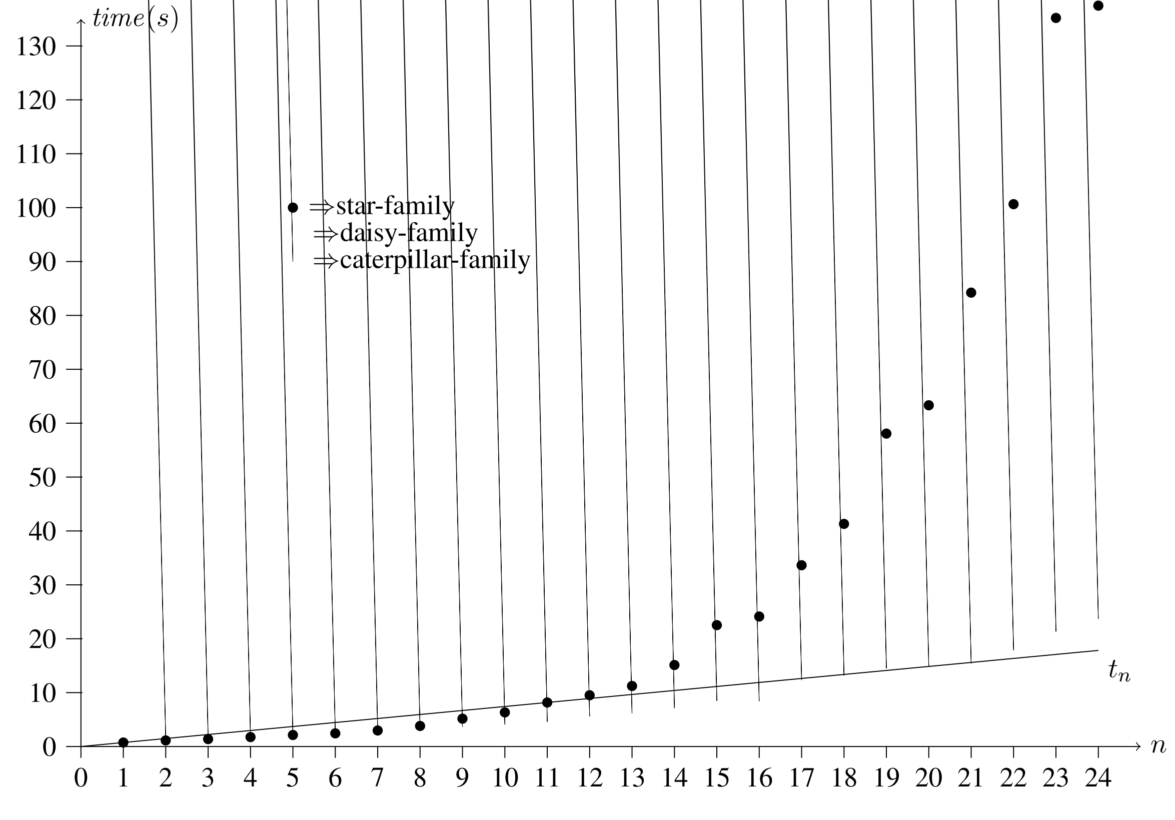


Figure 20. Zoom on CPUtime for articulated families.

We also performed a regression analysis to determine how the CPU time of the PN-synthesis grows in term of the number of components in an articulation. As already observed, the results are different for the different families of transition systems we considered, and it seems difficult to find growth rules compatible with all the figures we obtained. We then considered several subranges for the number of components.

First, up to 10 components, the three families behave about the same. The best fit corresponds to an exponential growth of the kind $0.9 \cdot 1.01^{|S|}$, thus with a base very close to 1; a power regression also gives a rather good fit with a formula of the kind $0.07 \cdot x^{0.77}$, lower than linear.
From $n = 10$ up to 50, the star-shape family becomes worse than the other two, which remain very similar. For the first one, the power regression gives rather good results, with a degree of 4.25, while for the next two families we get a lower degree around 2.6.
Beyond $n = 50$, the star-family disappears (as explained before) and the next two behave differently (with a strange, reproducible, hop for the daisy-flower family around 80 components, maybe due to a change in the usage of the cache memory) . For the daisy-family, we get a good power regression with a degree of 4.6; and for the caterpillar-family the power is 4.1. We thus have a polynomial-like behaviour by chunks, with an increasing power of $|S|$.

For the factorisation decomposition, where the size of the products grows very rapidly ($|S|^n$ states for a product of n components of the same size $|S|$), the performance is rapidly better when applying synthesis to the various components, as detailed in [9].

The previous analysis assumed that we consider synthesis problems where the target is the class of bounded weighted Petri nets. Thus all the nets we consider are bounded, but we do not fix these bounds beforehand. If we do, i.e., if we search for safe of k-safe solutions (with fixed k), then suddenly the problem becomes NP-complete (in the worst case, see for instance [7, 24]) instead of polynomial[5]. Corollaries 3.4 and 4.13 show that factorisations and articulations may be used to solve k-safe synthesis problems. That means that the synthesis algorithms we know for generating safe or k-safe solutions are exponential in the worst cases. If it occurs that P=NP, that means that we will be able to derive polynomial algorithms, but probably with a high polynomial degree. Then the gain obtained by a divide and conquer strategy (when it works) may be much larger than the ones we mention above.

7. Concluding remarks

We have developed a theory, algorithms and experiments around two (families of) pairs of operators acting on labelled transition systems and Petri nets. This allows both for structuring large transition systems and improving the efficiency of Petri net syntheses.

We may of course wonder if it will often be the case that transition systems needing a PN-synthesis present such structures. In fact, it depends where these systems come from. For instance, if they come from the organiser of a tool competition, needing a set of large positive examples: it may happen that they were obtained by combining smaller components (especially if the organiser does not know the decomposition algorithms developed in the present paper).

Other possible issues are to examine how this may be specialised for some subclasses of Petri nets and how these structures behave in the context of approximate solutions, devised when an exact synthesis is not possible, in the spirit of the notions and procedures developed in [5].

Finally, other kinds of operator pairs could be searched for, having interesting decomposition and recomposition procedures, allowing again to speed up synthesis problems.

[5]note that in this case we have to add constraints of the kind $\forall s \in S : M_s(p) \leq k$ to solve the various separation subproblems, rendering the linear systems non-homogeneous; hence we may not rely to a rational solution to derive one in the integer domain.

Acknowledgements

The author thanks Eike Best, Uli Schlachter, as well as the anonymous referees of the conference papers from which the present work is issued, for their useful remarks and suggestions. The remarks of anonymous referees also allowed to improve the present paper.

References

[1] Badouel E, Bernardinello L, Darondeau P. Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, 2015. ISBN:978-3-662-47967-4. doi:10.1007/978-3-662-47967-4.

[2] Best E, Devillers R. Characterisation of the State Spaces of Live and Bounded Marked Graph Petri Nets. In: 8th International Conference on Language and Automata Theory and Applications (LATA 2014). 2014 pp. 161–172. doi:10.1007/978-3-319-04921-2_13.

[3] Best E, Schlachter U. Analysis of Petri Nets and Transition Systems. In: Proceedings 8th Interaction and Concurrency Experience, ICE 2015, Grenoble, France, 4-5th June 2015. 2015 pp. 53–67. doi:10.4204/EPTCS.189.6.

[4] Best E, Devillers R, Schlachter U. Bounded choice-free Petri net synthesis: algorithmic issues. *Acta Inf.*, 2018. **55**(7):575–611. doi:10.1007/s00236-017-0310-9.

[5] Schlachter U. Over-Approximative Petri Net Synthesis for Restricted Subclasses of Nets. In: Language and Automata Theory and Applications - 12th International Conference, LATA 2018, Ramat Gan, Israel, April 9-11, 2018, Proceedings. 2018 pp. 296–307. doi:10.1007/978-3-319-77313-1_23.

[6] Badouel E, Bernardinello L, Darondeau P. Polynomial Algorithms for the Synthesis of Bounded Nets. In: TAPSOFT'95: Theory and Practice of Software Development, 6th International Joint Conference CAAP/FASE, Aarhus, Denmark. 1995 pp. 364–378. doi:10.1007/3-540-59293-8_207.

[7] Badouel E, Bernardinello L, Darondeau P. The Synthesis Problem for Elementary Net Systems is NP-Complete. *Theor. Comput. Sci.*, 1997. **186**(1-2):107–134. doi:10.1016/S0304-3975(96)00219-8.

[8] Devillers R. Factorisation of transition systems. *Acta Informatica*, 2018. **55**(4):339–362. doi:10.1007/s00236-017-0300-y.

[9] Devillers R, Schlachter U. Factorisation of Petri Net Solvable Transition Systems. In: Application and Theory of Petri Nets and Concurrency - 39th International Conference, PETRI NETS 2018, Bratislava, Slovakia. 2018 pp. 82–98. doi:10.1007/978-3-319-91268-4_5.

[10] Devillers R. Articulation of Transition Systems and its Application to Petri Net Synthesis. In: Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings. 2019 pp. 113–126. doi:10.1007/978-3-030-21571-2_8.

[11] Arnold A. Finite Transition Systems - Semantics of Communicating Systems. Prentice Hall international series in computer science. Prentice Hall, 1994. ISBN:978-0-13-092990-7.

[12] Desel J, Reisig W. The Synthesis Problem of Petri Nets. *Acta Inf.*, 1996. **33**(4):297–315. doi:10.1007/s002360050046.

[13] Devillers R. Products of Transition Systems and Additions of Petri Nets. In: Proc. 16th International Conference on Application of Concurrency to System Design (ACSD 2016) J. Desel and A. Yakovlev (eds). 2016 pp. 65–73. doi:10.1109/ACSD.2016.10.

[14] Keller RM. A Fundamental Theorem of Asynchronous Parallel Computation. In: Sagamore Computer Conference, August 20-23 1974, LNCS Vol. 24. 1975 pp. 102–112. doi:10.1007/3-540-07135-0_113.

[15] Hopcroft JE, Tarjan RE. Efficient Algorithms for Graph Manipulation [H] (Algorithm 447). *Commun. ACM*, 1973. **16**(6):372–378. doi:10.1145/362248.362272.

[16] Westbrook J, Tarjan RE. Maintaining Bridge-Connected and Biconnected Components On-Line. *Algorithmica*, 1992. **7**(5&6):433–464. doi:10.1007/BF01758773.

[17] Best E, Devillers R, Koutny M. The Box Algebra = Petri Nets + Process Expressions. *Inf. Comput.*, 2002. **178**(1):44–100. doi:10.1006/inco.2002.3117.

[18] Best E, Devillers RR, Erofeev E. A New Property of Choice-Free Petri Net Systems. In: Application and Theory of Petri Nets and Concurrency - 41st International Conference, PETRI NETS 2020, Paris, France, June 24-25, 2020, Proceedings. 2020 pp. 89–108. doi:10.1007/978-3-030-51831-8_5.

[19] Karmarkar N. A new polynomial-time algorithm for linear programming. *Combinatorica*, 1984. **4**(4):373–396. doi:10.1007/BF02579150.

[20] Dantzig G. Maximization of a linear function of variables subject to linear inequalities. In: Koopmans T (ed.), Activity Analysis of Production and Allocation, Proceedings. Wiley, New York, 1951 p. 339–347.

[21] Klee V, Minty G. How Good Is the Simplex Algorithm? In: Shisha O (ed.), Inequalities III, Proceedings. Academic Press, New York, 1951 p. 159–175.

[22] SMTInterpol, an Interpolating SMT Solver. https://ultimate.informatik.uni-freiburg.de/smtinterpol/.

[23] Kroening D, Leroux J, Rümmer P. Interpolating Quantifier-Free Presburger Arithmetic. In: Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings. 2010 pp. 489–503. doi:10.1007/978-3-642-16242-8_35.

[24] Tredup R. Hardness Results for the Synthesis of b-bounded Petri Nets. In: Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings. 2019 pp. 127–147. doi:10.1007/978-3-030-21571-2_9.