Synthesis of (Choice-Free) Reset Nets

Raymond Devillers

Département d'Informatique, Université Libre de Bruxelles,
B-1050 Brussels, Belgium (rdevil@ulb.ac.be)

Abstract. Instead of synthesising a labelled transition system into a weighted Petri net, we shall here consider the larger class of nets with reset arcs, allowing to instanciate a larger class of transition systems. We shall also target an extension of choice-free nets with reset arcs, since choice-free nets appeared to be especially interesting in terms of properties, synthesis and implementation. In addition to a general algorithm, we shall analyse how to speed it up by reducing the number and complexity of the linear systems of constraints to be solved and how to set up a pre-synthesis phase. We shall also envisage how to implement the result of such a synthesis as a concurrent program.

Keywords: labelled transition systems; reset nets; choice-freeness; synthesis.

1 Introduction

In order to validate a system, instead of analysing a model of the latter to check if it satisfies a set of desired properties, the synthesis approach tries to build a model "correct by construction" directly from those properties, and then to implement it. In particular, if the behaviour of a system is specified by a finite labelled transition system (LTS for short), more or less efficient algorithms have been developed to build a bounded weighted Petri net with a reachability graph isomorphic to (or close to) the given LTS [2, 19]. It is also possible to target some subclasses of Petri nets [6], in particular choice-free nets and some of their specialisations [7, 8, 4, 13] which present interesting features.

On the contrary, in order to extend a bit the power of the technique, we shall here consider a superclass of the classical Petri nets, by allowing reset arcs [1]. When one extends Petri nets, it is often the case that properties which are decidable for the latter (albeit sometimes with a huge complexity) become undecidable. And indeed, for reset nets, boundedness and reachability (in particular) are undecidable [14]. This increases the interest to avoid analysis techniques in favour of synthesis ones.

The paper is organised as follows. After recalling classical definitions, notations and properties in Section 2, we present presynthesis phases in Section 3, and then general algorithms to synthesise (choice-free) reset nets in Section 4. In the next sections, we analyse how to speed up the synthesis and to implement the resulting models. As usual, we conclude in the last section.

2 Classical Definitions, Notations and Properties

Definition 1. LTS, SEQUENCES AND REACHABILITY

A *labelled transition system with initial state*, *LTS* for short, is a quadruple $TS = (S, \rightarrow, T, \iota)$ where S is the set of *states*, T is the set of *labels*, $\rightarrow \subseteq (S \times T \times S)$ is the *transition relation*, and $\iota \in S$ is the *initial state*.
A label t is *enabled* at $s \in S$, written $s[t\rangle$, if $\exists s' \in S\colon (s, t, s') \in \rightarrow$, in which case s' is said to be *reachable* from s by the firing of t, and we write $s[t\rangle s'$. Generalising to any (firing) sequences $\sigma \in T^*$, $s[\varepsilon\rangle$ and $s[\varepsilon\rangle s$ are always true, with ε being an empty sequence; and $s[\sigma t\rangle s'$, i.e., σt is *enabled* from state s and leads to s' if there is some s'' with $s[\sigma\rangle s''$ and $s''[t\rangle s'$.
A state s' is *reachable* from state s if $\exists \sigma \in T^* : s[\sigma\rangle s'$. The set of states reachable from s is noted $[s\rangle$. □ 1

Definition 2. SOME PROPERTIES OF LTS

$TS = (S, \rightarrow, T, \iota)$ is *fully reachable* if $S = [\iota\rangle$, i.e., each state is reachable from the initial one.
TS is *forward deterministic* if $\forall s \in S, \forall t \in T : s[t\rangle s' \wedge s[t\rangle s'' \Rightarrow s' = s''$. It is *backward deterministic* if $\forall s \in S, \forall t \in T : s'[t\rangle s \wedge s''[t\rangle s \Rightarrow s' = s''$. It is *deterministic* if it is both forward and backward deterministic, i.e., the successors or predecessors of a state are determined by the labels of the arcs.
TS is *quasi-persistent* if $\forall s, s_1, s_2 \in S\ \forall a \neq b \in T : s[a\rangle s_1 \wedge s[b\rangle s_2 \Rightarrow s_1[b\rangle \wedge s_2[a\rangle$, i.e., if there is a choice, it persists until both labels are performed
TS is *persistent* if $\forall s, s_1, s_2 \in S\ \forall a \neq b \in T : s[a\rangle s_1 \wedge s[b\rangle s_2 \Rightarrow s_1[b\rangle s'' \wedge s_2[a\rangle s''$ for some $s'' \in S$, i.e., it is quasi-persistent and the resulting states are the same.
TS is reversible if $\forall s \in [\iota\rangle : \iota \in [s\rangle$, i.e., every reachable state allows to go back to the initial state. □ 2

Definition 3. PETRI NETS AND REACHABILITY GRAPHS.

A (finite, place-transition) *weighted Petri net*, or *weighted net*, is a tuple $N = (P, T, W)$ where P is a finite set of *places*, T is a finite set of *transitions*, with $P \cap T = \emptyset$, and W is a *weight* function $W\colon ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}$ giving the weight of each arc.
A *Petri net system*, or *system*, is a tuple $\mathcal{S} = (N, M_0)$ where N is a net and M_0 is the *initial marking*, a marking being a member of $P \rightarrow \mathbb{N}$ (hence a member of $\mathbb{N}^P$) indicating the number of *tokens* in each place.
A transition $t \in T$ is *enabled by* a marking M, denoted by $M[t\rangle$, if for all places $p \in P$, $M(p) \geq W(p, t)$. If t is enabled at M, then t can *occur* (or *fire*) in M,

leading to the marking M' defined by $M'(p) = M(p) - W(p,t) + W(t,p)$; this is denoted by $M[t\rangle M'$. A marking M' is *reachable* from M if there is a sequence of firings leading from M to M'. The set of markings reachable from M is denoted by $[M\rangle$. The *reachability graph of* $\mathcal{S}$ is the labelled transition system $RG(\mathcal{S})$ with the set of vertices $[M_0\rangle$, the set of labels T, initial state M_0 and transitions $\{(M,t,M') \mid M,M' \in [M_0\rangle \wedge M[t\rangle M'\}$. $\square$ 3

A reset net (RPN for short) or system is an easy extension of the classical Petri nets and system, defined as follows:

Definition 4. Reset nets and systems.

A (finite, place-transition, weighted) *reset net* is a tuple $N = (P,T,W,R)$ where (P,T,W) is a Petri net and $R \subseteq P \times T$ is a set of (undirected) reset arcs. A reset system is a reset net provided with an initial marking M_0: (P,T,W,R,M_0).
A transition $t \in T$ is *enabled by* a marking M, denoted by $M[t\rangle$, if for all places $p \in P$, $M(p) \geq W(p,t)$, i.e., it is enabled in the underlying Petri net. If t is enabled at M, then t can *occur* (or *fire*) in M, leading to the marking M' defined by $M'(p) = M(p) - W(p,t) + W(t,p)$ if $(p,t) \notin R$ and $M'(p) = W(t,p)$ otherwise, denoted by $M[t\rangle M'$. The latter case may be interpreted as follows: first, t absorbs $W(p,t)$ tokens from p, then erases the rest of the tokens in p, and finally produces $W(t,p)$ new tokens in p.
Like for Petri nets, a marking M' is *reachable* from M if there is a sequence of firings leading from M to M' and the set of markings reachable from M is denoted by $[M\rangle$. The *reachability graph of a reset system* $\mathcal{S} = (P,T,W,R,M_0)$ is the labelled transition system $RG(\mathcal{S})$ with the set of vertices $[M_0\rangle$, the set of labels T, initial state M_0 and transitions $\{(M,t,M') \mid M,M' \in [M_0\rangle \wedge M[t\rangle M'\}$.
A (reset) net is *pure* if $\forall p \in P, t \in T : W(p,t) \cdot W(t,p) = 0$, i.e., no transition both checks the presence of tokens in a place and produces tokens in that place.
For any place $p \in P$, we shall denote $p^\bullet = \{t \in T | W(p,t) > 0\}$ (the set of transitions collecting tokens from p, also called successors or outputs of p) and $R(p) = \{t \in T | (p,t) \in R\}$ (the set of transitions resetting p). $\square$ 4

Definition 5. Boundedness

A (reset or Petri net) system $\mathcal{S}$ is *bounded* if $\exists k \in \mathbb{N}\ \forall p \in P\ \forall M \in [M_0\rangle : M(p) \leq k$. It is k-bounded if $\forall p \in P\ \forall M \in [M_0\rangle : M(p) \leq k$. $\square$ 5

A classical (and easy) result is that

Corollary 1. Bounded system

A (reset or Petri net) system $\mathcal{S}$ is bounded iff its reachability graph $RG(\mathcal{S})$ is finite. $\square$ 1

Among the very numerous subclasses of Petri net systems that have been considered in the literature, choice-free ones[1] (meaning there is no true choice to be

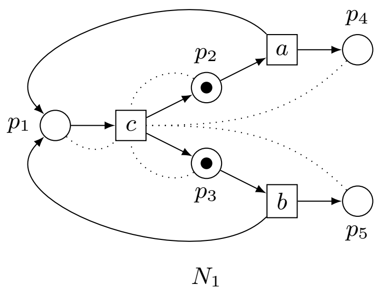
[1] not to be confused with free-choice nets [9]

performed when two or more transitions are enabled [20]; they have also been called output-nonbranching [5]) appeared very interesting in terms of properties, synthesis and implementation [8]. We shall thus introduce a similar subclass for reset nets.

Definition 6. Choice-free subclasses

A Petri net is said *choice-free* if $\forall p \in P : |p^\bullet| \leq 1$.
A reset net will be said choice-free if $\forall p \in P : (|p^\bullet \cup R(p)| \leq 1)$. That is, each place has at most one successor transition and at most one resetting transition, and if they are both present they must be the same. □ 6

In graphical representations, reset arcs will be drawn as (undirected) dotted lines, and as usual arcs with null weight are omitted. Figure 1 presents a reset net and the corresponding reachability graph for some initial marking. It is not choice free, while each place has a single output transition and a single reset arc, but not always the same: for instance p_2 has output a and is reset by c.



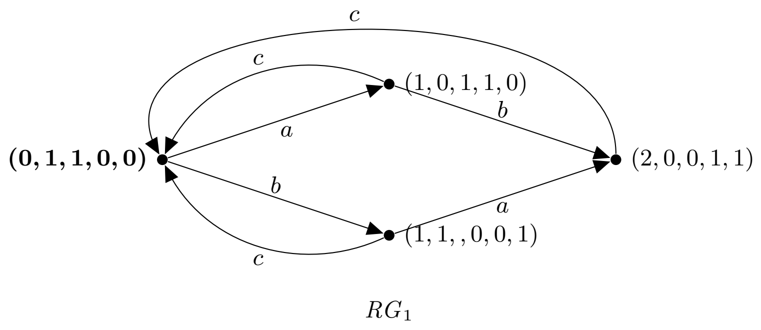


Fig. 1. A reset net system and its (finite) reachability graph for the initial marking specified in bold.

On the contrary, Figures 2 and 3 present bounded choice-free reset net systems, and their reachability graphs (the initial markings are still respresented in bold).

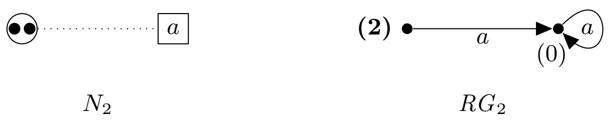


Fig. 2. A simple choice-free reset net and its reachability graph

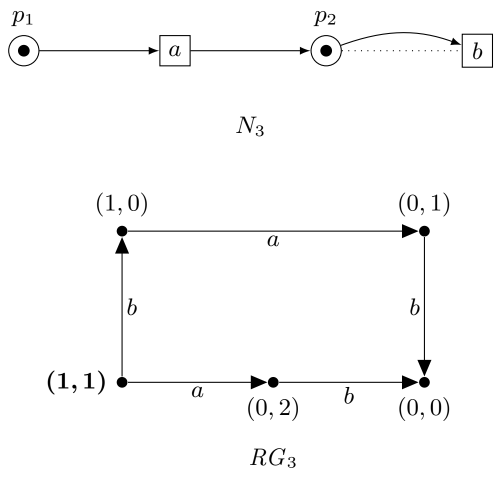


Fig. 3. Another bounded choice-free reset net and its reachability graph

Markings may be considered as a kind of vectors with indices in P, but we shall also consider vectors of transitions.

Definition 7. T-Vectors

A *T-vector* is an element of $\mathbb{N}^T$.
The *support* of a vector is the set of the indices of its non-null components.
A vector is called *prime* if the greatest common divisor of its components is one (i.e., its components do not have a common non-unit factor).
The *Parikh vector* $\Psi(\sigma)$ of a finite sequence $\sigma \in T^*$ of transitions is a T-vector counting the number of occurrences of each transition in σ, and the *support* of σ is the support of its Parikh vector, i.e., $supp(\sigma) = supp(\Psi(\sigma)) = \{t \in T \mid \Psi(\sigma)(t) > 0\}$. □ 7

Definition 8. Synthesis

Two LTS $TS_1 = (S_1, \to_1, T, \iota_1)$ and $TS_2 = (S_2, \to_2, T, \iota_2)$ are isomorphic if there is a bijection $\zeta\colon S_1 \to S_2$ with $\zeta(\iota_1) = \iota_2$ and $(s, t, s') \in \to_1 \Leftrightarrow (\zeta(s), t, \zeta(s')) \in \to_2$, for all $s, s' \in S_1$.
If an LTS TS is isomorphic to the reachability graph $RG(\mathcal{S})$ of some system $\mathcal{S}$, we say that $\mathcal{S}$ *solves* TS (or X-solves it, if X is the class of $\mathcal{S}$). A LTS is X-solvable if a system of class X solves it.
A synthesis is a procedure aimed at finding a solution from TS (when possible); it thus consists to keep the structure of the reachability graph of a system (dropping the exact values of the markings) in order to obtain a given LTS. □ 8

3 Presynthesis

The presynthesis phase consists in checking that the given LTS satisfies some structural properties common to all the reachability graphs of the target class. These properties need of course to be easy to check. If a check fails, we may immediately reject the synthesis and produce a reason, easy to understand in general.

For a reset net synthesis, we may use the following:

Proposition 1. General properties of reset nets

The reachability graph of a bounded reset net system is finite, totally reachable and forward deterministic.

Proof: Finiteness results from Corollary 1. Forward determinism results from the firing rule, and total reachability from the definition of a reachability graph. □ 1

Hence, if a LTS is not finite, or not totally reachable, or not forward deterministic (easy to check if the LTS is given explicitely), there is no reset net solution (and we know why). Note however that, contrary to what happens for usual Petri nets, it may happen that the reachability graph of a reset net is not backward deterministic. This may be observed on Figure 2 for instance (N_2 is even a choice-free reset net): there are two arcs labelled a arriving at node (0).

For a choice-free reset net synthesis, we may use the following:

Proposition 2. General properties of choice-free reset nets

The reachability graph of a bounded choice-free reset net system is finite, totally reachable, forward deterministic and quasi-persistent.

Proof: The first three properties result from Proposition 1.
Quasi-persistence results from the observation that, in a reset net system, for any place $p \in P$, the marking of p may only be decreased when firing t if $t \in p^\bullet \cup R(p)$.

Hence, if the system is choice-free, the marking of p may only be decreased by a single transition. As a consequence, if t is enabled by some marking, it remains so at least until t is fired. □ 2

However, contrary to what happened for usual Petri nets, it may happen that the reachability graph of a choice-free reset net system is not persistent: this is illustrated by Figure 3, where N_3 is a choice-free reset net system, a and b are initially enabled, but $M_0[ab\rangle$ and $M_0[ba\rangle$ lead to different markings.

Many general properties of choice-free net system have been discovered and proposed for a pre-synthesis phase (see [8]). However, very few of them remain valid for choice-free reset net systems.

For instance, it is known that bounded choice-free nets always have home states in their reachability graphs, i.e., states that remain reachable whatever the evolution of the system (this is due to Keller's theorem [18]). This is not true for choice-free reset nets, as illustrated by Figure 4.

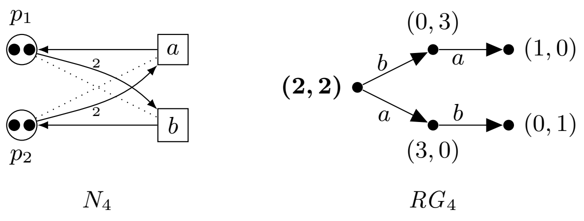


Fig. 4. A choice-free reset system and its reachability graph without home state

Next, if the reachability graph of a bounded choice-free net is acyclic, all paths between two reachable markings have the same Parikh vector: Figure 3 shows that this is no longer true for choice-free reset nets since there are two paths ab and bab from $(1,1)$ to $(0,0)$.

Moreover, it is known that, in the reachability graph of a bounded choice-free net, cycles are propagated Parikh-equivalently: if $s[\alpha\rangle s$ and $s[a\rangle s'$ for some $a \in T$ and $\alpha \in T^*$, then $s'[\beta\rangle s'$ with $\Psi(\alpha) = \Psi(\beta)$ for some $\beta \in T^*$. This is no longer true when we add reset arcs, as illustrated by Figure 5. We may observe in this example that, while the initial cycle (a simple loop a) is not transported on the next state, it is however transported on the last state. We may then wonder if any cycle is eventually transported if we go further enough. This is not true however, as illustrated by Figure 6: the initial cycle $babc$ is not transported if we perform the path ab since this leads to a dead end.

Finally, many interesting properties of bounded choice-free net systems are linked to the minimal Parikh vectors of non-empty cycles (a cycle with such a minimal Parikh vector is called *small*). However, when there are reset arcs, Parikh vectors are no longer characteristic of cycles. For instance, in Figure 2, RG_2 has two

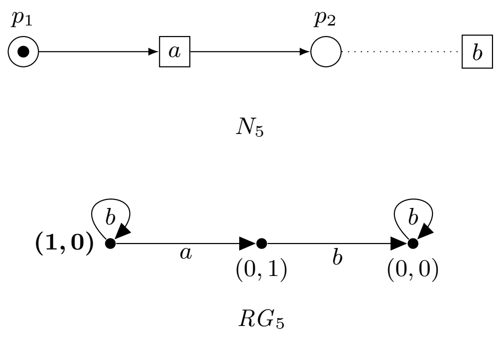


Fig. 5. A bounded choice-free reset net and its reachability graph, where cycles are not always pushed forward

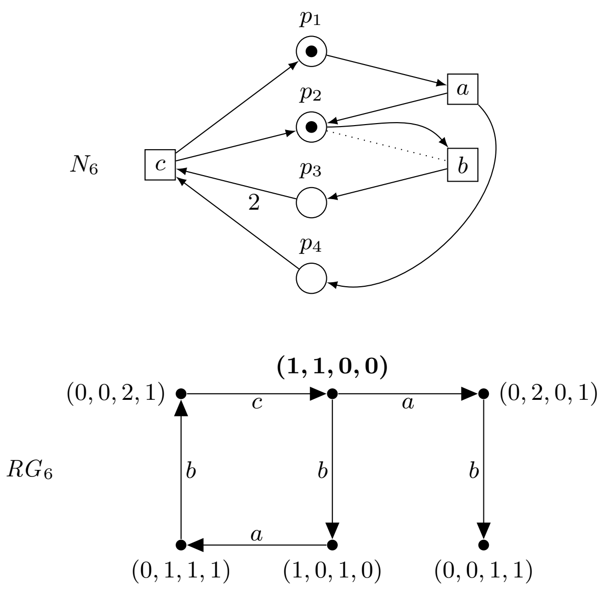
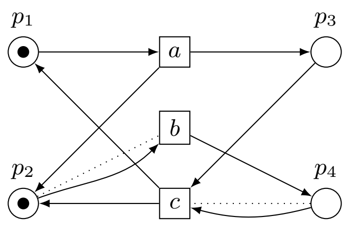


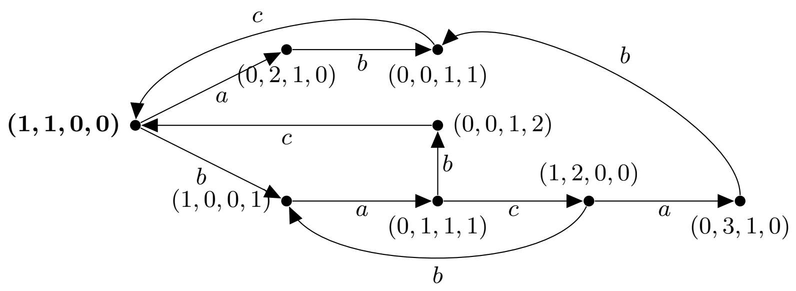
Fig. 6. A bounded choice-free reset net and its reachability graph, where a cycle is not at all pushed forward

paths a (hence with the same Parikh vector), but only one of them defines a cycle (in Figure 5, there are three paths b, but only two of them are cycles). Hence we may suspect that the properties of small cycles will be different for reset net systems.

For instance, in a bounded choice-free net system, any cycle has a Parikh vector which is a sum of Parikh vectors of small cycles, but this is no longer true if we add reset arcs, as illustrated in Figure 7: the cycles which do not visit twice some state are abc (and acb), $babc$ and $bacabc$, of which only the first one is small, but the Parikh vector $(1, 2, 1)$ of $babc$ is not a multiple of the minimal Parikh vector $(1, 1, 1)$ (note also that this example shows a bounded system where it is possible to reach a strictly larger marking: $(1, 2, 0, 0)$ may be reached from the initial marking $(1, 1, 0, 0)$; this explain why the classical Karp-Miller procedure [17] does not work for reset nets and reset arcs make boundedness undecidable).



N_7



RG_7

Fig. 7. A bounded choice-free reset net and its reachability graph, where cycles do not have a base of small cycles

The status of other classical properties of the reachability graphs of bounded choice-free nets is uncertain. For instance, it is not known if the Parikh vectors of small cycles remain prime, nor if they are either equal or disjoint.

4 General Algorithms

A classical technique to perform a Petri net synthesis is to start from a net with transitions only, then to add progressively new places in order to constrain the evolutions to get closer and closer to what is specified by the given LTS. Each added place must allow all the evolutions permitted by the specification, and exclude some forbidden situations that were allowed by the previously added places. The process leaves some freeness in the way these new places are added, so that the result is usually not unique, and it may happen that some additions are not possible, meaning there is no solution to the considered synthesis problem.

Let $TS = (S, \to, T, \iota)$ be a given labelled transition system. The theory of regions [2] characterises the solvability of an LTS through the solvability of a set of *separation problems*. In case the LTS is finite, we have to solve $\frac{1}{2}\cdot|S|\cdot(|S|-1)$ states separation problems and up to $|S|\cdot|T|$ event/state separation problems, as follows:

Definition 9. Separation problems

- A *(Petri net) region* of $(S, \to, T, \iota)$ is a triple $(\mathbb{M}, \mathbb{B}, \mathbb{F}) \in (S \to \mathbb{N}, T \to \mathbb{N}, T \to \mathbb{N})$ such that for all $s[t\rangle s' \in \to$, $\mathbb{M}(s) \geq \mathbb{B}(t)$ and $\mathbb{M}(s') = \mathbb{M}(s) - \mathbb{B}(t) + \mathbb{F}(t)$. A region models a place p, in the sense that $\mathbb{B}(t)$ models $W(p,t)$, $\mathbb{F}(t)$ models $W(t,p)$, and $\mathbb{M}(s)$ models the token count of p at the marking corresponding to s (and in particular, $\mathbb{M}(\iota)$ models the initial marking of p).
- A *states separation problem* (SSP for short) consists of a set of states $\{s, s'\}$ with $s \neq s'$, and it can be solved by a region (or place) distinguishing them, i.e., has a different number of tokens in the markings corresponding to the two states: $\mathbb{M}(s) \neq \mathbb{M}(s')$. There are $|S| \cdot (|S| - 1)/2$ such problems.
- An *event/state separation problem* (ESSP for short) consists of a pair $(s,t) \in S \times T$ with $\neg s[t\rangle$. For every such problem, one needs a region (or place) such that $\mathbb{M}(s) < \mathbb{B}(t)$. There are $|S| \cdot |T| - |\to|$ such problems. □ 9

If the LTS is infinite, also the number of separation problems (of each kind) becomes infinite, but we need to find a finite set of regions solving all of them. Other techniques must then be searched for, instead of considering each separation problem separately, but here we shall restrict our attention to finite LTSs, i.e., to bounded solutions. Then, [10] showed that a Petri net synthesis problem is solvable iff each separation problem has a solution, and a possible solution to the synthesis problem is obtained by gathering all the places corresponding to those separation problem solutions.

For reset net synthesis problems, the situation is similar, but we need to consider reset regions:

Definition 10. Reset region

A *reset region* (RPN-region for short) of $(S, \rightarrow, T, \iota)$ is a tuple $(\mathbb{M}, \mathbb{R}, \mathbb{B}, \mathbb{F}) \in (S \rightarrow \mathbb{N}, 2^T, T \rightarrow \mathbb{N}, T \rightarrow \mathbb{N})$ such that for all $s[t\rangle s' \in \rightarrow$, $\mathbb{M}(s) \geq \mathbb{B}(t)$ and $\mathbb{M}(s') = \mathbb{M}(s) - \mathbb{B}(t) + \mathbb{F}(t)$ if $t \notin \mathbb{R}$, $\mathbb{F}(t)$ otherwise. A region models a place p (see Figure 8), in the sense that $\mathbb{B}(t)$ models $W(p, t)$, $\mathbb{F}(t)$ models $W(t, p)$, $\mathbb{M}(s)$ models the token count of p at the marking corresponding to s, and $\mathbb{R}$ specifies which transitions belong to $R(p)$. □ 10

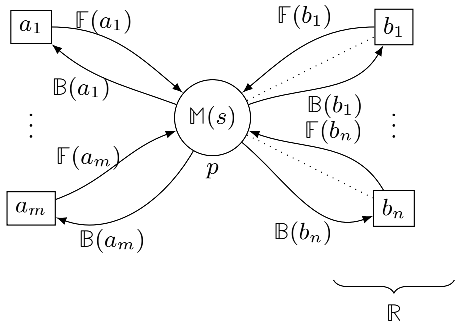


Fig. 8. A general reset region (pictured as a place p). $T = \{a_1, \ldots, a_m, b_1, \ldots, b_n\}$ and $\mathbb{R} = \{b_1, \ldots, b_n\}$. $\mathbb{M}(\iota)$ is the initial marking of p, and more generally $\mathbb{M}(s)$ is the marking of p corresponding to state s.

The proofs of [10] may be immediately adapted so that a finite, totally reachable, forward deterministic LTS admits a RPN-solutions iff each separation problem has a RPN-region solution, and a possible solution to the synthesis problem is obtained by gathering all the places corresponding to those separation problem solutions.

For each separation problem, we thus have to solve a system of $(1 + 2 \cdot |\rightarrow|)$ linear constraints: 1 to express the separation problem to consider (i.e., either $\mathbb{M}(s) \neq \mathbb{M}(s')$ or $\mathbb{M}(s) < \mathbb{B}(t)$), $|\rightarrow|$ constraints expressing that a transition is possible, and the same number to express the resulting marking. There are $(|S|+2\cdot|T|)$ variables in $\mathbb{N}$: $|S|$ variables $\mathbb{M}(s)$, $|T|$ variables $\mathbb{B}(t)$ and $|T|$ variables $\mathbb{F}(t)$. But for each case we may have to consider up to $2^{|T|}$ configurations for $\mathbb{R}$, as illustrated by Figure 8.

For the synthesis of choice-free reset nets, instead of considering $2^{|T|}$ configurations, we only have to consider $2 \cdot |T|$ of them, as illustrated[2] on Figure 9, each one having $1 + |T| + |S|$ variables. Note however that, for each event/state separation problem (s, t), we only have to consider 2 configurations since b must be t in this case. On the contrary, for a states separation problem (s, s'), b is not prescribed and we may need to consider all the possible configurations (in particular when the problem has no solution).

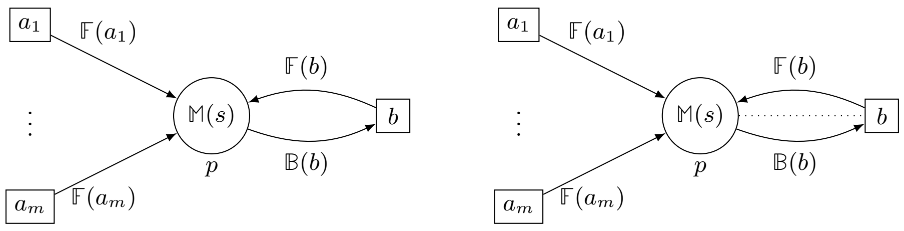


Fig. 9. The two general choice-free reset regions (pictured as a place p). $T = \{a_1, \ldots, a_m, b\}$ and $\mathbb{R}$ is either empty or $\{b\}$. $\mathbb{M}(\iota)$ is the initial marking of p, and more generally $\mathbb{M}(s)$ is the marking of p corresponding to state s.

For each separation problem and each configuration for $\mathbb{R}(p)$, the system of linear constraint to be solved is polynomial in the size of the LTS to be solved, and it is homogeneous (no independent term), so that instead of searching a solution in the integer domain we may work in the (non-negative) rational one and afterwards apply a multiplicative factor to get integer solutions. We may thus use a polynomial procedure, like the Karmarkar's one [16]. This may no longer be true if we add other constraints; for instance, if we search for a k-bounded solution, we may add $|S|$ constraints $\mathbb{M}(s) \leq k$ (for each state $s \in S$): the size of the system remains polynomial, but it is no longer homogeneous, and the problem is NP-complete (see also [3, 21]).

5 Acceleration

Since the complexity of a reset net synthesis may be quite high (still worse than for usual Petri net synthesis), it may be beneficial to use a divide and conquer strategy [11, 12] and decompose the given LTS as a product or an articulation of simpler components: those decompositions were developed in the context of Petri net synthesis, but they apply as well to reset nets.

[2] Note however that, since $\mathbb{B}(b)$ may be null, several configurations of the left kind may intersect.

Concerning the choice-free case, several accelerations have been exploited in [7] for instance, based on the analysis of their reachability graphs. Unfortunately, most of them are no longer valid when we add reset arcs.

For instance, it has been shown that, for choice-free synthesis, states separation problems are irrelevant: if a LTS satisfies the pre-synthesis checks and all the event-state separation problems, then the states separation problems are automatically satisfied too. This is no longer the case for choice-free reset synthesis, as illustrated by Figure 2: there is no event-state separation problems since there is a single label (a), enabled at each state; however, we need to separate the two states of the corresponding LTS.

Let us first consider the solution of the event-state separation problems for some event $b \in T$. It is not necessary to consider all of them in general, and it is sometimes possible to slightly simplify the systems of linear constraints to be solved.

Indeed, if we consider the two forms of places/regions detailed in Figure 9, we may see that, if there is a cycle in the given LTS without the label b, all the labels a_i occurring in it must have a weight $\mathbb{F}(a_i) = 0$ (otherwise, the marking of p strictly increases around the cycle). As a consequence, if $\mathbb{C}(b)$ denotes the set of labels occurring in cycles without b, $\forall a \in \mathbb{C}(b) : \mathbb{F}(a) = 0$. Moreover, if $s[a\rangle s'$ for $a \in \mathbb{C}(b)$, we have $\mathbb{M}(s) = \mathbb{M}(s')$. If $\mathbb{C}(b) \neq \emptyset$, both properties reduce the number of variables to find when searching a region of the kinds exhibited in Figure 9 (and it may happen that none is found, in which case the synthesis has no solution).

Now, if $s[a\rangle s'$ and $s[b\rangle$ with $a \neq b$, from the quasi-persistence (satisfied if the given LTS passed succesfully the presynthesis phase), we also have $s'[b\rangle$, so that we do not have to separate b from s' if we do not have to do it from s. If $a \in \mathbb{C}(b)$ and $s[a\rangle s'$, since $\mathbb{M}(s) = \mathbb{M}(s')$ for any region of the two kinds illustrated in Figure 9, if $\neg s[b\rangle$ any place separating s from b will also separate s' from b; we may deduce that if $s'[b\rangle$, the synthesis by a choice-free reset net is then impossible (this could then be incorporated in the presynthesis phase). If $a \in \mathbb{C}(b)$, $\neg s[b\rangle$ and $\neg s'[b\rangle$, it is equivalent to separate b from s and to s'. Let us thus define the equivalence relation $s_1 \sim_b s_2$ generated by $\exists a \in \mathbb{C}(b) : s[a\rangle s'$. If $s_1 \sim_b s_2$ and $\neg s_1[b\rangle$, we must also have $\neg s_2[b\rangle$, but we only have to separate a single state from b in each equivalence class where b must be excluded. Moreover, it is not always necessary to separate all such equivalence classes from b.

Indeed, if $s[a\rangle s'$ with $a \in T \setminus (\{b\} \cup \mathbb{C}(b))$ and $\neg s'[b\rangle$, $\mathbb{M}(s) \leq \mathbb{M}(s')$ for any region of the two kinds illustrated in Figure 9. As a consequence, any place separating b from s' will also separate it from s. Let us thus consider the graph whose nodes are the equivalence classes of $\sim_b$ which do not enable b, with an arc from c_1 to c_2 if there is $s_1[a\rangle s_2$ with $s_1 \in c_1$, $s_2 \in c_2$ and $a \in T \setminus (\{b\} \cup \mathbb{C}(b))$. From the discussion above, we only have to consider the event-state separation problems separating a member of a rightmost class (in this graph) from b.

More easily, when considering an event-state separation problem for b, it is always advisable to first look if some place previously devised for another separating problem for b does not already solve the present problem.

Let us now consider a states separation problem for $s \neq s'$. Again, we may first look if some of the regions constructed before does not already solve it.

From the discussion above, it occurs that if $s \sim_b s'$ for some $b \in T$, we should not search for a separating region with output or reset b, since then $\mathbb{M}(s) = \mathbb{M}(s')$: this reduces the burden of finding an adequate region, if any. And in particular, if $s \sim_b s'$ for any $b \in T$, it is not possible to separate s from s' and the synthesis fails (again, this could be incorporated in the presynthesis phase).

When a given LTS is reversible, it is known [5] that it has a classical choice-free solution iff it has a pure one, which reduces the number of unknowns in each linear system to be solved. We shall now see that this extends immediately when we add reset arcs.

Proposition 3. PREFIXED LTS

If each label occurring in some LTS occurs on a cycle around the initial state, then this LTS has a choice-free reset solution iff it has a pure one. Moreover, in the general schemes illustrated in Figure 9, we may always assume $\mathbb{F}(b) = 0$.

Proof: First, we may observe that, if $t \in T$ but t does not occur as a label in the LTS, this may be obtained by a (pure) place without input, without (initial) token, and with a unique output transition t. Hence, in the following, without loss of generality we shall assume that each transition labels some arc in the given LTS.

Now, let $b \in T$ and let us consider a reset region $(\mathbb{M}, \mathbb{R}, \mathbb{B}, \mathbb{F})$ of the kind illustrated in Figure 9, so that either $\mathbb{R} = \emptyset$ (non-resetting region) or $\mathbb{R} = \{b\}$ (region resetting b).

For any arc $s[b\rangle s'$ in the LTS, we may observe that $\mathbb{M}(s') \geq \mathbb{F}(b)$ in the non-resetting case, and $\mathbb{M}(s') = \mathbb{F}(b)$ in the resetting case. Moreover, if $s'[\sigma\rangle s''$ with $\sigma \in (T \setminus \{b\})^*$, $\mathbb{M}(s'') \geq \mathbb{M}(s')$. Since we assumed that, for some $s \in S$, $\exists \sigma \in T^*$: $s[b\sigma\rangle\iota$, we deduce $\mathbb{M}(\iota) \geq \mathbb{F}(b)$. And since $\forall s \in S\ \exists \sigma \in T^* : \iota[\sigma\rangle s$, we also have $\forall s \in S : \mathbb{M}(s) \geq \mathbb{F}(b)$.

Now, let $k = \min(\mathbb{F}(b), \mathbb{B}(b))$. Let us consider the object obtained from $(\mathbb{M}, \mathbb{R}, \mathbb{B}, \mathbb{F})$ by subtracting k from each $\mathbb{M}(s)$ as well as from $\mathbb{B}(b)$ and from $\mathbb{F}(b)$. It is easy to see from the previous property that this object is still a choice-free reset region, and that if the original region solves a (states or event/state) separation problem, the same is true for the new one. Since in the new region we may not have both $\mathbb{B}(b) > 0$ and $\mathbb{F}(b) > 0$, this leads to a pure solution if we apply this procedure to each region of a choice-free reset solution of the given LTS.

Finally, let us consider a pure non-resetting region for b (left of Figure 9). If $\mathbb{F}(b) > 0$, we have $\mathbb{B}(b) = 0$, but then we cannot have a cycle with b (the marking would increase indefinitely while following the cycle), which contradicts the hypotheses. For a pure resetting region $(\mathbb{M}, \mathbb{R}, \mathbb{B}, \mathbb{F})$ for b (right of Figure 9), if $\mathbb{F}(b) > 0$, we have $\mathbb{B}(b) = 0$ and (from the argument above) $\forall s \in S : \mathbb{M}(s) \geq \mathbb{F}(b)$. Let us then consider the object obtained by subtracting $\mathbb{F}(b)$ from each $\mathbb{M}(s)$ and replacing $\mathbb{F}(b)$ by 0. It is easy to see that this object is still a choice-free reset region for b, and that if the original region solved a states separation problem (it cannot solve an event/state separation problem since $\mathbb{B}(b) = 0$), the same is true for the new one. We thus may assume in any case that $\mathbb{F}(b) = 0$. □ 3

For instance, any LTS isomorphic to RG_6 in Figure 6 has a pure choice-free reset solution, for instance N_6. The same is true for RG_7 in Figure 7, and more generally we have the following corollary:

Corollary 2. PURE SOLUTIONS

If an LTS is reversible, it has a choice-free reset solution iff it has a pure one.

□ 2

6 Net Implementation

Synthesis may be considered as a simple step in an implementation process. From a behavioural specification (for instance in the form of a LTS), it allows to find (if possible) a model of a certain class, which may be considered as a structural specification presenting the adequate behaviour. It then remains to implement this model in a practical device, either hardware or software, or mixed.

In our case, since Petri nets and their extensions are especially devised to describe a distributed application, we may try to obtain a program, where each place corresponds to data structures (giving in particular the number of tokens in the place, but each token may provide other informations that will be used by the absorbing agent) and parallel processes for each transition. Since we consider models with 'black' tokens, the control flow will not rely on the information carried by the tokens, but this information may be exploited by the transition-process when it absorbs the needed tokens and fires. After or during the processing of these informations, the agent will produce some tokens in some places, possibly carrying some information that will be available in the future (but not for the control flow).

In general, a classical problem may occur when the agents check the availability of their needed tokens in a distributed way: it may happen that an agent observes that some input place (or all of them) has the needed tokens, but before the firing takes place and absorbs them, another agent does the same and absorbs the tokens before, disabling the first agent. In order to avoid this, a solution is to lock all accesses to memory when an agent tries to get its tokens, but this is not very distributed. Another solution is that each agent progressively locks all its input places, in some order compatible with the orders used by the other agents (putting all these local orders together must yield a – possibly partial – global order, to avoid deadlocks), but fixing this order is not exactly distributed either. During this locking, the transition may observe if the needed tokens are available, and if it is not the case it will be necessary to unlock all the locked places and retry later (this may induce some starvation phenomenon).

For choice-free nets, the situation is much more sympathetic, since there is no conflict in accessing the input places of each agent. Hence, if a transition observes that there are enough tokens in some place, this may not be changed by other transitions: the latter may only increase the set of tokens in the considered place. Note however that we could have problems if a transition is duplicated in several processes in order to implement some form of auto-concurrency [15]: we shall thus assume that auto-concurrency is not allowed in our systems. It may be necessary to lock accesses to each place however, in order to avoid intermixing absorptions and productions of tokens in the place by different parallel processes (classical problem when performing additions and/or subtractions in parallel, with local copies of the variables), but this may be done in a distributed way. Moreover, if a transition observes that there are not enough tokens in some input place, it is necessary to unlock the place and retry later, but it is never necessary to restart from the beginning. It is even possible to absorb needed tokens when their presence is observed, even if not all of them are there: it is never necessary to give back the absorbed tokens if the transition is blocked at some point, and starvations are equivalent to blockings. The productions and absorptions may be done concurrently in any order, provided the places are protected against simultaneous accesses, since additions and subtractions commute ($+i - j + k = -j + k + i$). This does not create problems and does not perturb the evolutions of the underlying Petri net (but spurious intermediate markings may be created).

For general reset nets, since this generalises Petri nets, we encounter the same problems, and the same non-distributed solutions.

For choice-free reset nets, we have the same separation of the input places as for usual choice-free Petri nets, hence we have the same fact that checking in a distributed way that the needed tokens are available is not destroyed by the other agents. However, we have a problem with the production phase of each agent, due to the fact that commuting a reset and a production is not innocuous.

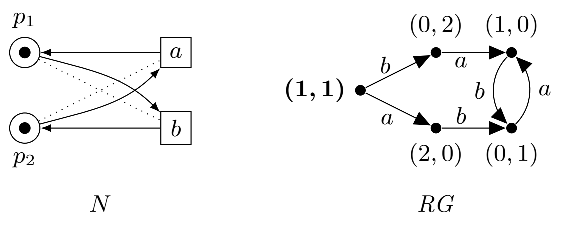


Fig. 10. A simple choice-free reset system and its reachability graph

This is illustrated by the example on Figure 10. Initially, both a and b are enabled. Let us assume they observe it simultaneously, absorb the needed tokens and proceed as follows: a produces a token in p_1, b resets p_1, b produces a token in p_2 and a resets p_2; we get the marking $(0, 0)$ which blocks the system, hence does not allow to still perform infinitely often ab or ba as specified by the reachability graph on the right.

Hence, for the reset and production phase of a transition, we must again either lock all the places, or progressively all its reset-output and output places in some well-defined order. However, in the latter case, when a place is locked by another transition, one only has to wait for its unlocking to proceed: it is never necessary to undo some modification, nor restart the phase, nor wait for an extra delay. When all the needed places have been modified we may unlock them and restart the input-phase.

The structure of each implemented transition could then be sketched as illustrated on Figure 11. There are variants of this schema however; for instance, the absorption of the input tokens in some place may be performed progressively, without waiting they are all present simultaneously.

```
repeat
    for each input place $p$ (i.e., such that $W(p,t) > 0$) in any order do
        end-collect = false
        repeat
            lock place $p$
            if there are not enough tokens ($M(p) < W(p,t)$)
            then unlock place $p$; wait for some time
            else collect the needed tokens ($M(p) = M(p) - W(p,t)$);
                 unlock place $p$; end-collect = true
        until end-collect
    process the action of the transition
            (possibly using the hidden information of the black tokens)
    for each output or reset place $p$ (i.e., such that $t \in R(p)$ or $W(t,p) > 0$)
        in an adequate order do
        lock place $p$
        if $p$ is reset by $t$ (i.e., $t \in R(p)$)
        then erase the remaining tokens of $p$, if any (then $M(p) = 0$)
        if $p$ is an output place for $t$
        then  produce $W(t,p)$ tokens (possibly with adequate hidden
              information, depending on the ones of the used tokens)
    for each output or reset place $p$, in any order do unlock place $p$
until we want to stop
```

Fig. 11. Sketch of a parallel subprogram implementing transition t

7 Concluding Remarks and Future Work

We succeeded in finding how to synthesise, when possible, a finite LTS into a Petri net when we allow reset arcs, either in the general case or in the choice-free case.

We explored how to realise a pre-synthesis phase, but some work has still to be accomplished. In particular, the status of two important properties has to be determined: the primality of small cycles and the disjointness of small cycles with non-identical Parikh vectors.

Region theory has been extended to cope with the addition of reset arcs, and the complexity of the separation problems has been delineated.

Some practical accelerations have been exhibited, but it is likely that some more could be discovered.

Finally, the way to implement a structural specification with reset arcs as a concurrent program has been analysed.

Of course, it should be possible to consider other superclasses of Petri nets (like the ones with inhibitor arcs or transfer arcs or a mixture of those various extensions), as well as other subclasses of those superclasses (similar to marked graphs or free-choice nets for instance) but the region approach assumes that the constraints are only linked to individual places, so the extensions may sometimes be delicate.

Acknowledgements

We want to thank the anonymous referees for their careful reading, as well as Eike Best for his encouragements, and some interesting examples.

References

1. Araki, T., Kasami, T.: Some decision problems related to the reachability problem for Petri nets. Theor. Comput. Sci. 3(1), 85–104 (1976)
2. Badouel, E., Bernardinello, L., Darondeau, P.: Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag (2015), `https://doi.org/10.1007/978-3-662-47967-4`
3. Badouel, E., Bernardinello, L., Darondeau, P.: The synthesis problem for elementary net systems is NP-complete. Theor. Comput. Sci. 186(1-2), 107–134 (1997), `https://doi.org/10.1016/S0304-3975(96)00219-8`
4. 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). pp. 161–172 (2014), `https://doi.org/10.1007/978-3-319-04921-2\_13`
5. Best, E., Devillers, R.: Synthesis of persistent systems. In: 35th International Conference on Application and Theory of Petri Nets and Concurrency (ICATPN 2014). pp. 111–129 (2014), `https://doi.org/10.1007/978-3-319-07734-5\_7`

6. Best, E., Devillers, R., Erofeev, E., Wimmel, H.: Target-oriented Petri net synthesis. Fundamenta Informaticae 175, 97–122 (2020), https://doi.org/10.3233/FI-2020-1949
7. Best, E., Devillers, R., Schlachter, U.: Bounded choice-free Petri net synthesis: algorithmic issues. Acta Inf. 55(7), 575–611 (2018)
8. Best, E., Devillers, R.R., 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. pp. 89–108 (2020), https://doi.org/10.1007/978-3-030-51831-8_5
9. Desel, J., Esparza, J.: Free Choice Petri Nets, Cambridge Tracts in Theoretical Computer Science, vol. 40. Cambridge University Press, New York, USA (1995)
10. Desel, J., Reisig, W.: The synthesis problem of Petri nets. Acta Inf. 33(4), 297–315 (1996)
11. 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). pp. 65–73 (2016), https://doi.org/10.1109/ACSD.2016.10
12. 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. pp. 113–126 (2019)
13. Devillers, R., Hujsa, T.: Analysis and synthesis of weighted marked graph Petri nets. In: Application and Theory of Petri Nets and Concurrency - 39th International Conference, PETRI NETS 2018, Bratislava, Slovakia, June 24-29, 2018, Proceedings. pp. 19–39 (2018)
14. Dufourd, C., Finkel, A., Schnoebelen, P.: Reset nets between decidability and undecidability. In: Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings. pp. 103–115 (1998), https://doi.org/10.1007/BFb0055044
15. Grabowski, J.: On partial languages. Fundam. Informaticae 4(2), 427–498 (1981)
16. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica 4(4), 373–396 (1984)
17. Karp, R., Miller, R.: Parallel program schemata. Journal of Computer and System Sciences 3(2), 147âĂŞ195 (1969)
18. Keller, R.M.: A Fundamental Theorem of Asynchronous Parallel Computation. In: Sagamore Computer Conference, August 20-23 1974, LNCS Vol. 24. pp. 102–112 (1975), https://doi.org/10.1007/3-540-07135-0_113
19. 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. pp. 296–307 (2018), https://doi.org/10.1007/978-3-319-77313-1_23
20. Teruel, E., Colom, J.M., Silva, M.: Choice-Free Petri Nets: a Model for Deterministic Concurrent Systems with Bulk Services and Arrivals. IEEE Transactions on Systems, Man, and Cybernetics, Part A 27(1), 73–83 (1997), http://dx.doi.org/10.1109/3468.553226
21. 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. pp. 127–147 (2019), https://doi.org/10.1007/978-3-030-21571-2_9