Algorithms for the Synthesis of Elementary Net Systems with Localities

Aishah Ahmed and Marta Pietkiewicz-Koutny

School of Computing, Newcastle University
Newcastle upon Tyne NE4 5TG, United Kingdom
{a.ahmad,marta.koutny}@ncl.ac.uk

Abstract. Elementary Net Systems with Localities (ENL-systems) is a class of Petri nets introduced to model GALS (globally asynchronous locally synchronous) systems, where some of the components might be considered as logically or physically close and acting synchronously, while others might be considered as loosely connected or residing at distant locations and communicating with the rest of the system in an asynchronous way. The specification of the behaviour of a GALS system comes very often in the form of a transition system. The automated synthesis, based on regions, is an approach that allows to construct Petri net models from their transition system specifications. While theory of regions is well developed, there is still lack of implemented tools capable of dealing with complex real-life system construction. In this paper we focus on developing algorithms and tool support for the synthesis of the ENL-systems from step transition systems, where arcs are labelled by steps (sets) of executed actions. The algorithms are implemented within the WORKCRAFT framework.

Keywords: theory of concurrency, Petri nets, localities, analysis and synthesis, step sequence semantics, theory of regions, transition systems, WORKCRAFT framework.

1 Introduction

A number of computational systems exhibit behaviour that follows the 'globally asynchronous locally (maximally) synchronous' paradigm. Examples can be found in hardware design, where a VLSI chip may contain multiple clocks responsible for synchronising different subsets of gates [12], and in biologically inspired membrane systems representing cells within which biochemical reactions happen in synchronised pulses [23]. To formalise such systems, [16] introduced *Place/Transition-nets with localities* (PTL-nets), where each locality defines a distinct set of events which must be executed synchronously, i.e., in a maximally concurrent manner (often called *local maximal concurrency*).

An attractive way of constructing complex computing systems is their automated synthesis from behavioural specifications given in terms of suitable transition systems. In such a case, the synthesis procedure is often based on the regions of a transition system, a notion introduced in [14], and later used to solve the synthesis problem for many different classes of Petri nets [2, 4, 7, 6, 21, 22, 24]. A comprehensive, systematic survey of the synthesis problem and region theory is presented in [3].

The vast majority of results in the area of synthesis of Petri nets use the standard transition systems, where the arcs are labelled with single events/actions, as initial specifications of systems' behaviour. In this paper, however, we follow the approach, used in [17–20, 24], employing step transition systems instead, where arcs are labelled with sets of executed events/actions.

This paper is concerned with finding efficient algorithms for deriving regions for the synthesis of ENL-systems. The nets with localities, as already mentioned, were first introduced in [16] using as a base a class of Place/Transition nets. The idea of actions' localities was later adapted to Elementary Net Systems (EN-systems) in [17], where a solution to the synthesis problem for ENL-systems was presented. Further advances in the area of synthesising nets with localities from step transition systems are the subjects of [18–20]. The papers [17–19] suitably adapted the classical theory of regions [4] to cope with local maximal concurrency in the context of three different classes of nets, while [20] concentrated on finding the rules for reducing the number of regions that are essential to sythesise ENL-systems. The mentioned sequence of papers built up a theory for the synthesis of ENL-systems. This paper concentrates on implementing existing theoretical results into a tool, with the main focus on computing regions of ENL-transition systems. The algorithms are implemented within the WORKCRAFT framework [25, 27]. Developing efficient algorithms for the synthesis of ENL-systems is challenging (note that the problem is NP-complete which can be shown following the argument made in [2]). Also, there are no implemented tools for the synthesis of Petri nets from step transition systems. The ones that exist, like Petrify [9, 15, 10], VipTool [5], ProM [1], Genet [8] or Rbminer [26] work with specifications that do not include the information about concurrency in the form of steps of simultaneously executed actions.

To explain the basic idea behind ENL-systems, let us consider the net in Figure 1 modelling two co-located consumers and one producer residing in a remote location. In the initial state, the net can execute the singleton step $\{c_4\}$. Another enabled step is $\{p_2\}$ which removes the token from b_1 and puts a token in both b_0 and b_2. In this new state, there are three enabled steps, viz. $\{p_1\}$, $\{c_1, c_4\}$ and $\{p_1, c_1, c_4\}$. The last one, $\{p_1, c_1, c_4\}$, corresponds to what is usually called *maximal concurrency* as no more activities can be added to it without violating the constrains imposed by the available resources (represented by tokens). However, the previously enabled step $\{c_4\}$ which is still *resource (or token) enabled* is disallowed by the control mechanism of ENL-systems. It rejects a resource enabled step like $\{c_4\}$ since we can add to it c_1 co-located with c_4 obtaining a step which is resource enabled. In other words, the control

mechanism employed by ENL-systems (and PTL-nets) is that of *local maximal concurrency* as indeed postulated by the GALS systems execution rule.

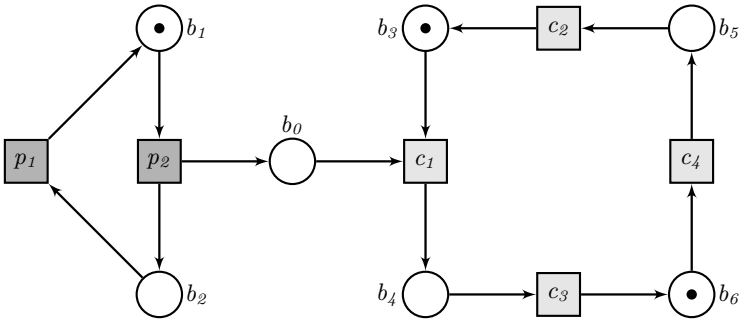

Fig. 1: A one producer/two co-located consumers system (shading of boxes indicates the co-location of events they represent).

The paper is organised as follows. The next section recalls some basic notions concerning step transition systems, ENL-transition systems, ENL-systems and their synthesis. Section 3 introduces an algorithm for computing regions of a given step transition system (ENL-transition system) and provides theoretical results to support the algorithm. The paper ends with a conclusion that includes some directions for future work.

2 Preliminaries

Let E be a fixed finite non-empty set of *events*. A *co-location relation on* E is any equivalence relation $\mathrel{-\!\!\!\circ}$ on the set of events. Moreover, for an event e and a non-empty set of events U (called a *step*), we will denote $e \mathrel{-\!\!\!\circ} U$ whenever there is at least one event $f \in U$ satisfying $e \mathrel{-\!\!\!\circ} f$.

Definition 1. *A* step transition system on E *is a triple* $\mathfrak{ts} \stackrel{\text{df}}{=} (Q, A, q_0)$ *where* Q *is a non-empty finite set of* states, $A \subseteq Q \times (2^E \setminus \{\varnothing\}) \times Q$ *is a finite set of* transitions (arcs), *and* $q_0 \in Q$ *is the* initial state. *We will write* $q \xrightarrow{U} q'$ *(or simply* $q \xrightarrow{U}$*) whenever* (q, U, q') *is a transition. We will call* q *the* source *of transition* (q, U, q') *and* q' *its* target.

To ease the presentation, we will assume that each event of E occurs in at least one of the steps labelling the transitions of $\mathfrak{ts}$.

Definition 2. *A sequence of steps* $U_1, \ldots, U_k$ *forms a* step sequence *from* q *to* q', σ, *in a step transition system* $\mathfrak{ts} = (Q, A, q_0)$ *if there are states* $q = q_1, q_2, \ldots, q_{k+1} = q'$ *such that* $(q_i, U_i, q_{i+1}) \in A$ *for* $i = 1, \ldots, k$. *We will denote*

this $q \xrightarrow{U_1 \ldots U_k} q'$ *or* $q \xrightarrow{\sigma} q'$. *We will say that a step sequence* has no loops *if all its states are distinct (adjacent to at most two transitions).*

Let $\mathfrak{ts} = (Q, A, q_0)$ and $\mathfrak{ts}' = (Q', A', q_0')$ be step transition systems. We say that $\mathfrak{ts}$ and $\mathfrak{ts}'$ are *isomorphic*, $\mathfrak{ts} \cong \mathfrak{ts}'$, if there is a bijection $f : Q \longrightarrow Q'$ such that $f(q_0) = q_0'$ and $(q, U, q') \in A \Leftrightarrow (f(q), U, f(q')) \in A'$, for all $q, q' \in Q$ and $U \in 2^E \setminus \{\varnothing\}$.

In diagrams of bigger step transition systems we will write annotations of arcs for singleton steps without braces (*e.g.*, e instead of $\{e\}$). Also, by *thick arcs* in a step transition system we will mean arcs labelled by non-singleton steps.

In the following sub-sections we gather definitions introduced in [17, 19, 20].

2.1 ENL-systems

Definition 3 ([17]). *An* elementary net system with localities *(ENL-system) is a tuple*

$$\mathfrak{enl} \stackrel{\text{df}}{=} (B, E, F, \doteq, c_0)$$

such that B *is a finite set of* conditions *disjoint from the events,* $F \subseteq (B \times E) \cup (E \times B)$ *is the* flow relation, $\doteq$ *is a co-location relation on* E, *and* $c_0 \subseteq B$ *is the* initial case *(in general, any subset of* B *is a* case*).*

In diagrams, conditions (local states) are represented by circles, events (actions) by boxes, the flow relation by directed arcs, and each case (global state) by tokens (small black dots) placed inside those conditions which belong to this case. Moreover, boxes representing co-located events are shaded in the same way (see Figure 1).

For every event e, its *pre-conditions* and *post-conditions* are given respectively by

$$^\bullet e \stackrel{\text{df}}{=} \{b \mid (b, e) \in F\} \text{ and } e^\bullet \stackrel{\text{df}}{=} \{b \mid (e, b) \in F\}$$

(both sets are assumed non-empty and disjoint). Two events are *in conflict* (or conflicting) if they share a pre-condition, or share a post-condition. The dot-notation extends to sets of events in the usual way, e.g., $^\bullet U \stackrel{\text{df}}{=} \bigcup\{^\bullet e \mid e \in U\}$.

The semantics of $\mathfrak{enl}$ is based on steps of simultaneously executed events. We first define *potential steps of* $\mathfrak{enl}$ as all non-empty sets of non-conflicting events. A potential step U is then *resource enabled* at a case c if $^\bullet U \subseteq c$ and $U^\bullet \cap c = \varnothing$, and *control enabled* if, in addition, there is no event $e \notin U$ such that $e \doteq U$ and the step $U \cup \{e\}$ is resource enabled at c. A control enabled step U can be *executed* leading from c to the case $c' = (c \setminus {}^\bullet U) \cup U^\bullet$. We denote this by $c[U\rangle c'$ (or $c[U\rangle$). The step execution mechanism employed by ENL-systems can therefore be called *local maximal concurrency.*

The *step transition system* of $\mathfrak{enl}$ is given by:

$$\mathfrak{ts}_{\mathfrak{enl}} \stackrel{\text{df}}{=} \big(C\,,\, \{(c, U, c') \in 2^B \times 2^E \times 2^B \mid c \in C \wedge c[U\rangle c'\}\,, c_0\big)\,,$$

where C — the set of *reachable* cases — is the least set of cases containing c_0 and closed w.r.t. the step execution relation. To ease the presentation, we will assume that $\mathfrak{enl}$ does not have *dead events*, i.e., each event occurs in at least one of the steps labelling the arcs of the step transition system $\mathfrak{ts}_{\mathfrak{enl}}$.

ENL-systems can be considered as Elementary Net Systems (EN-systems) equipped with an explicit notion of localities for their events. The local maximal semantics of their execution means that certain properties enjoyed by EN-systems do not hold for ENL-systems, like the following monotonicity property:

Fact 1 *The step monotonicity for EN-system states that: If $c[U\rangle$ and $U' \subseteq U$ then $c[U'(U \setminus U')\rangle$.*

2.2 ENL-transition systems

To link the nodes (global states) of a step transition system $\mathfrak{ts}$ with the conditions (local states) of the hypothetical ENL-system corresponding to it, we use the notion of a *region*.

Definition 4 ([17]). *A* region *with explicit input and output events is a triple*

$$\mathfrak{r} \stackrel{\text{df}}{=} (in, r, out) \in 2^E \times 2^Q \times 2^E,$$

where sets in and out satisfy the following implication

$$(in = \varnothing \ \wedge \ out = \varnothing) \ \Rightarrow \ (r = Q \ \vee \ r = \varnothing)$$

and for every transition $q \xrightarrow{U} q'$, the following hold:

R1 *If $q \in r$ and $q' \notin r$ then $|U \cap out| = 1$.*
R2 *If $q \notin r$ and $q' \in r$ then $|U \cap in| = 1$.*
R3 *If $U \cap out \neq \varnothing$ then $q \in r$ and $q' \notin r$.*
R4 *If $U \cap in \neq \varnothing$ then $q \notin r$ and $q' \in r$.*

Each region is a triple: $\mathfrak{r} = (in, r, out)$, where r is a subset of states of the step transition system, *in* is a subset of its events E which are responsible for entering r, and *out* is a subset of events E which are responsible for leaving r. There are exactly two *trivial* regions satisfying $r = \varnothing$ or $r = Q$, viz. $(\varnothing, \varnothing, \varnothing)$ and $(\varnothing, Q, \varnothing)$. Moreover, (in, r, out) is a region iff so is its *complement* $(out, Q \setminus r, in)$. We will denote the complement of a region $\mathfrak{r}$ by $\bar{\mathfrak{r}}$. In general, a region, as defined above, cannot be identified only by its set of states; in other words, *in* and *out* may not be recoverable from r. However, if the step transition system is *thin*, i.e., for every event $e \in E$ there is a transition $q \xrightarrow{\{e\}} q'$ of $\mathfrak{ts}$, then different regions are based on different sets of states [17].

The set of all non-trivial regions of $\mathfrak{ts}$ will be denoted by $\mathfrak{R}_{\mathfrak{ts}}$ and, for every state q, $\mathfrak{R}_q$ is the set of all non-trivial regions (in, r, out) containing q,

$$\mathfrak{R}_q \stackrel{\text{df}}{=} \{\mathfrak{r} \in \mathfrak{R}_{\mathfrak{ts}} \mid q \in r\}.$$

The sets of *pre-regions*, ${}^\circ e$, and *post-regions*, e°, of an event e comprise all the non-trivial regions (in, r, out) respectively satisfying $e \in out$ and $e \in in$,

$$^\circ e \stackrel{\text{df}}{=} \{\mathfrak{r} \in \mathfrak{R}_{\mathfrak{ts}} \mid e \in out\} \qquad \text{and} \qquad e^\circ \stackrel{\text{df}}{=} \{\mathfrak{r} \in \mathfrak{R}_{\mathfrak{ts}} \mid e \in in\}. \text{ [17]}$$

This extends in the usual way to sets of events, e.g., $^\circ U \stackrel{\text{df}}{=} \bigcup\{ {}^\circ e \mid e \in U\}$.

The set of *potential steps of* $\mathfrak{ts}$ comprises all non-empty sets U of events such that $^\circ e \cap {}^\circ f = e^\circ \cap f^\circ = \varnothing$, for each pair of distinct events $e, f \in U$. A potential step U is then *region enabled at state* q if $^\circ U \subseteq \mathfrak{R}_q$ and $U^\circ \cap \mathfrak{R}_q = \varnothing$ [17, 19].

Definition 5 ([17, 19]). *A step transition system* $\mathfrak{ts} = (Q, A, q_0)$ *is an* ENL-transition system *w.r.t. a co-location relation* $\simeq$ *if the following hold:*

A1 *Each state is reachable from the initial state.*
A2 *For every event* e*, both* $^\circ e$ *and* e° *are non-empty.*
A3 *For all distinct states* q *and* q'*,* $\mathfrak{R}_q \neq \mathfrak{R}_{q'}$*.*
A4 *For every state* q *and step* U*, we have that* $q \xrightarrow{U}$ iff U *is region enabled at* q *and there is no event* $e \notin U$ *such that* $e \simeq U$ *and the step* $U \cup \{e\}$ *is region enabled at* q*.*

One can show (see [17]) that the step transition system of an ENL-system with the co-location relation $\simeq$ is an ENL-transition system w.r.t. $\simeq$.

2.3 Synthesis of ENL-systems

ENL-systems generate ENL-transition systems. The converse also is true, and the translation from ENL-transition systems to the corresponding ENL-systems is based on the regions of step transition systems.

Let $\mathfrak{ts} = (Q, A, q_0)$ be an ENL-transition system w.r.t. a (given) co-location relation $\simeq$. Then the net system *associated* with $\mathfrak{ts}$ is defined as [17, 19]:

$$\mathfrak{enl}^{\simeq}_{\mathfrak{ts}} \stackrel{\text{df}}{=} (\mathfrak{R}_{\mathfrak{ts}}\,,\, E, F_{\mathfrak{ts}}\,,\, \simeq,\, \mathfrak{R}_{q_0})$$

where $F_{\mathfrak{ts}} \stackrel{\text{df}}{=} \{(\mathfrak{r}, e) \in \mathfrak{R}_{\mathfrak{ts}} \times E \mid \mathfrak{r} \in {}^\circ e\} \cup \{(e, \mathfrak{r}) \in E \times \mathfrak{R}_{\mathfrak{ts}} \mid \mathfrak{r} \in e^\circ\}$. It turns out that such a construction always produces an ENL-system which generates a transition system isomorphic to $\mathfrak{ts}$. (Note that one does not have to use all the regions in $\mathfrak{R}_{\mathfrak{ts}}$ to construct a desired ENL-system, and a method to reduce their number is described in [20].)

Theorem 1 ([17]). *Let* $\mathfrak{ts}$ *be an ENL-transition system w.r.t. a co-location relation* $\simeq$*. Then* $\mathfrak{enl}^{\simeq}_{\mathfrak{ts}}$ *is an ENL-system and its step transition system is isomorphic to* $\mathfrak{ts}$*. Moreover, the isomorphism* ψ *between* $\mathfrak{ts}$ *and the step transition system of* $\mathfrak{enl}^{\simeq}_{\mathfrak{ts}}$ *is given by* $\psi(q) \stackrel{\text{df}}{=} \mathfrak{R}_q$*, for every state* q *of* $\mathfrak{ts}$*.*

The $\mathfrak{enl}^{\simeq}_{\mathfrak{ts}}$ net system obtained from the synthesis of the step transition system might contain many conditions which are *redundant* from the point of view of its behaviour, *i.e.*, deletion of such conditions (and their adjacent arcs) would lead to a net whose step transition system is still isomorphic to $\mathfrak{ts}$ [20]. This paper is not concerned with eliminating any redundant regions/conditions.

3 Algorithm approaches

In this section, we present the approaches that have been used for designing and implementing algorithms to generate non-trivial regions of a given step transition system (ENL-transition system) as defined in Section 2.2. We consider the execution time of the algorithms as the main measure to gauge their efficiency.

The initial idea was to focus on thin step transition systems, as in these transition systems, for every event $e \in E$, there is a transition $q \xrightarrow{\{e\}} q'$, for some $q, q' \in Q$. This, we believed, would allow us to ignore the thick arcs and base our algorithm solely on singleton arcs, which in turn would enable us to use some of the techniques used in the Petrify tool [10], which works with standard transition systems rather than with step transition systems. Although we discovered that in certain circumstances the thick arcs can be ignored, leading to some reduction of execution time when verifying regional conditions **R1**-**R4**, the rules for ignoring thick arcs were complicated, and the class of thin step transition systems was not a special case, for checking these rules easily. The results of this approach are reported on in section 3.2 and we will use some of them, which apply to all step transition system, in our algorithms. Therefore, we abandoned the special case of thin step transition systems and decided to seek a solution for the general class of step transition systems (ENL-transition systems).

3.1 A general approach

In the first instance, we have implemented (starting with a brute-force approach) an algorithm for extracting regions of ENL-transition systems as defined in [17] and recalled in Section 2. By using this approach, we could obtain the expected results. However, it was noticeable that the execution time of deriving regions took too long even for small step transition systems. For example, the algorithm took around 37 minutes (see Figure 5) to generate the non-trivial regions for the ENL-transition system in Figure 2 (see also its associated net in Figure 3). This was not surprising. As region is a triple $\mathfrak{r} = (in, r, out) \in 2^E \times 2^Q \times 2^E$, the computation of all non-trivial regions in such a way would involve generation of all sub-sets of the set of states Q, all subsets of the set of events E and then checking regional conditions, **R1**-**R4**, for all the arcs in the step transition system. Thus, optimising the implemented (brute-force) algorithm was essential for reducing its execution time and making it of any practical value.

The approach for reducing the number of potential *in* and *out* sets of regions is based on the information about causality and concurrency embedded in the step transition systems. This approach is presented in section 3.3.

The approach for reducing the number of candidates for the sets of states (r) for regions is based on source and targets sets. This idea was first used in the Petrify tool [10] and is adapted here to the context of step transition systems. This approach is explained in section 3.4.

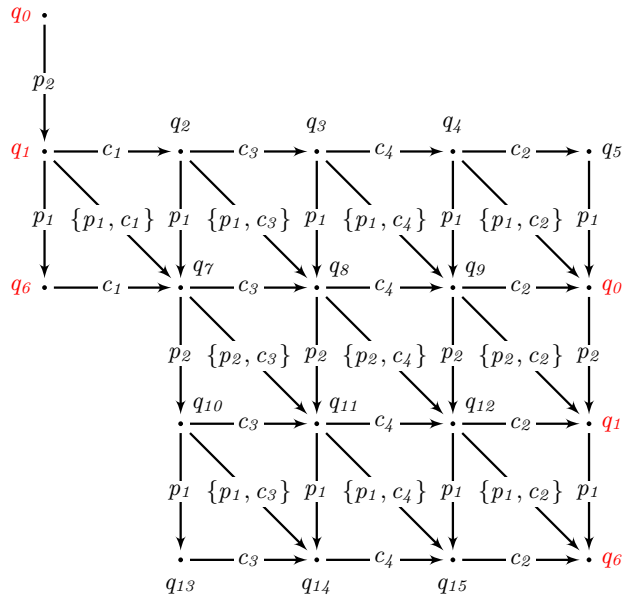

Fig. 2: An ENL-transition system $\mathfrak{ts}_3$, where p_1 and p_2 are co-located events, and c_1, c_2, c_3 and c_4 are co-located events. The graph should be glued on the states q_0, q_1, q_6 that appear twice in the picture.

3.2 Ignoring thick arcs

In this approach, we attempted to improve the execution time of extracting all the non-trivial regions by ignoring all the thick arcs in a given step transition systems in order to reduce the number of transitions ($q \xrightarrow{U} q'$) that need to be checked for every potential region (see Section 2.2).

The question is: which of the thick arcs can be ignored without changing the set of non-trivial regions? The thin step transition systems would be the first candidate of a class of systems to consider, because they contain the arcs labelled with all possible singleton steps, so all the events would be present in the set of steps labelling the remaining arcs of the step transition system after the removal of thick arcs. However, even in some of the thin transition systems removing thick arcs can be problematic as the Figure 4 shows. In that example removing the only thick arc, $\{e, f\}$, leads to the transition system being disconnected, meaning that previously ENL-transition system would not longer be classified as such on the grounds of not satisfying the axiom **A1**. The property of thinness is too weak to make a decision about the removal of thick arcs. We need to make much stronger assumptions as stated in the following proposition.

Proposition 1. *Let* $\mathfrak{ts} = (Q, A, q_0)$ *and* $\mathfrak{ts}' = (Q, A', q_0)$ *be two step transition systems, where* $\mathfrak{ts}'$ *is obtained from* $\mathfrak{ts}$ *by deleting a thick arc* $(q, U, q') \in A$ *labelled*

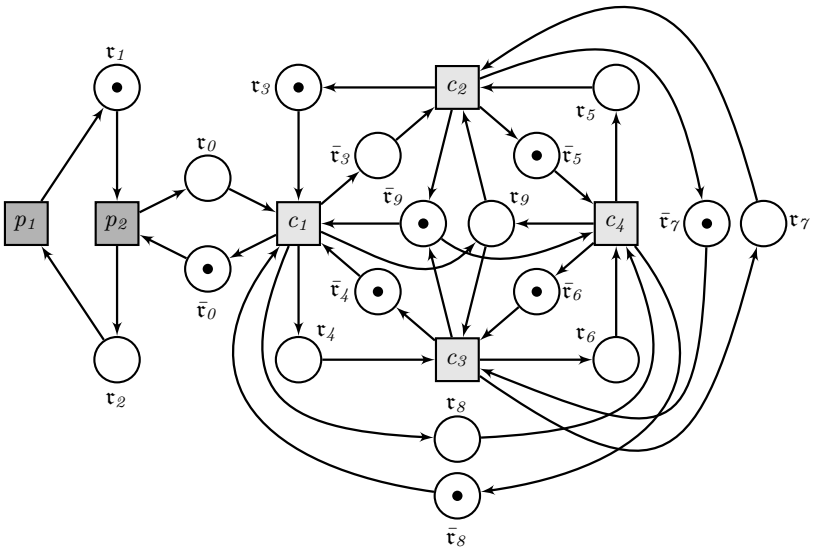

Fig. 3: The ENL-system resulting from the synthesis of the ENL-transition system $\mathfrak{ts}_3$ in Figure 2. Note that the synthesised net contains many redundant conditions.

by step U that satisfies: $U = U_1 \uplus \ldots \uplus U_k$ and there exists in $\mathfrak{ts}$ a step sequence $\sigma = U_{i_1} \ldots U_{i_k}$, leading from q to q', for any permutation of steps $U_1, \ldots, U_k$. Then $\mathfrak{R}_{\mathfrak{ts}} = \mathfrak{R}_{\mathfrak{ts}'}$.

Proof. The inclusion $\mathfrak{R}_{\mathfrak{ts}} \subseteq \mathfrak{R}_{\mathfrak{ts}'}$ holds trivially. We need to prove that $\mathfrak{R}_{\mathfrak{ts}'} \subseteq \mathfrak{R}_{\mathfrak{ts}}$. Let $\mathfrak{r} = (in, r, out) \in \mathfrak{R}_{\mathfrak{ts}'}$. We need to show that $\mathfrak{r} \in \mathfrak{R}_{\mathfrak{ts}}$, that means that for the arc (q, U, q') of $\mathfrak{ts}$ that was removed from $\mathfrak{ts}$ to obtain $\mathfrak{ts}'$ and $\mathfrak{r}$ the conditions **R1** - **R4** hold. We can assume, to the contrary, that one of the conditions does not hold for (q, U, q').
Case 1: **R1** does not hold for (q, U, q') and $\mathfrak{r}$. That means we have that $q \in r$ and $q' \notin r$ and $|U \cap out| \neq 1$. From assumptions, it follows that there is a step sequence in $\mathfrak{ts}'$, $\sigma = U_1 \ldots U_k$, where (q_1, U_1, q_2), $(q_2, U_2, q_3), \ldots, (q_k, U_k, q_{k+1}) \in A'$, $q = q_1$ and $q' = q_{k+1}$. As $\mathfrak{r} = (in, r, out) \in \mathfrak{R}_{\mathfrak{ts}'}$, $q \in r$ and $q' \notin r$, there exists a transition (q_i, U_i, q_{i+1}), $1 \leq i \leq k$, such that $q_i \in r$ and $q_{i+1} \notin r$ and for which **R1** is satisfied in $\mathfrak{ts}'$. Hence $|U_i \cap out| = 1$. Suppose that there exists another step, $U_j \neq U_i$, in the step sequence σ such that $(q_j, U_j, q_{j+1}) \in A'$ and $q_j \in r$ and $q_{j+1} \notin r$. From this and the assumptions it follows that there exists a permutation of steps of σ (which leads from q to q'), where U_i and U_j are consecutive steps, which is impossible as the sources of the arcs they label must belong to r and the targets must be outside r. We also cannot have a step U_j in σ labelling a transition (q_j, U_j, q_{j+1}), for which $q_j \notin r$ and $q_{j+1} \in r$, as in this case we would need to have either a loop in σ with two occurrences of U_i,

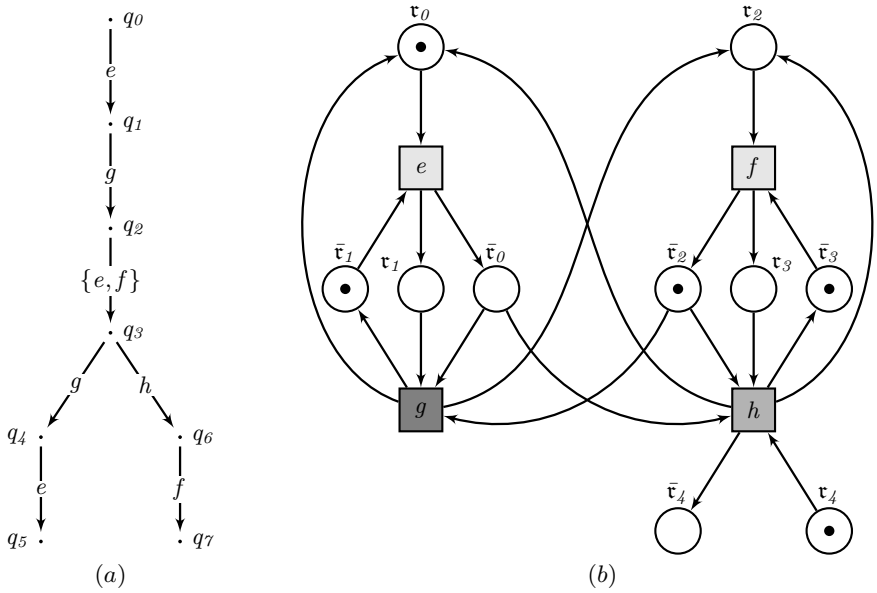

Fig. 4: An ENL-transition system $\mathfrak{ts}_2$ with co-located events e and f and non co-located events g and h (a); the ENL-system resulting from its synthesis (b).

or another step in σ, apart from U_i, which labels a transition exiting r, and we already proved that that is impossible. Hence, for all steps of σ apart from U_i, we have that they do not cross the border of r. As a consequence, from **R3**, which is satisfied for them, $|U_j \cap out| = 0$. As steps $U_1, \ldots, U_k$ are disjoint and $U = U_1 \uplus \ldots \uplus U_k$, we have $|U \cap out| = 1$, a contradiction. So, **R1** is satisfied for (q, U, q') and $\mathfrak{r}$.

Case 2: **R2** does not hold for (q, U, q') and $\mathfrak{r}$. The proof is similar to the proof of Case 1 and also leads to a contradiction. So, **R2** is satisfied for (q, U, q') and $\mathfrak{r}$.

Case 3: **R3** does not hold for (q, U, q') and $\mathfrak{r}$. That means $|U \cap out| \neq 0$ and $q \notin r$ or $q' \in r$. We can assume, without the loss of generality, that $q \notin r$. We now need to consider two cases:

1. If $q' \in r$ and $q \notin r$, there is a step U_i in σ, labelling a transition $(q_i, U_i, q_{i+1}) \in A'$, where $1 \leq i \leq k$ and $q_i \notin r$ and $q_{i+1} \in r$. Since **R3** holds for this transition and $\mathfrak{r}$ in $\mathfrak{ts}'$, we have $|U_i \cap out| = 0$. For other steps of σ, $U_j \neq U_i$, we must have $|U_j \cap out| = 0$, as otherwise we would have a transition, labelled with step U_j, exiting r, but that would mean that either σ has a loop with two occurrences of U_i, or there is another transition, labelled with a step, different from U_i, entering r again, and this is impossible according to the argument used in Case 1. So, $|U_i \cap out| = 0$ for $i = 1, \ldots, k$, and as a consequence $|U \cap out| = 0$, a contradiction.

2. If $q' \notin r$ and $q \notin r$, then again we have, for all steps U_i in σ, labelling transitions (q_i, U_i, q_{i+1}), $1 \leq i \leq k$, either $q_i, q_{i+1} \notin r$ or we have at least one pair of transitions, with one transition entering r (labelled U_i, for example) and one transition exiting r (labelled U_j, for example). The latter can be ruled out as according to the assumptions, there would need to be a step sequence between q and q' in $\mathfrak{ts}'$ with U_j and U_i being consecutive steps. However, that would mean that there must exist another two steps (different from U_i and U_j) in σ one labelling a transition entering r and another labelling a transition exiting r and we already know that this must be excluded as two steps labelling exiting transitions and two steps labelling entering transitions cannot be part of any permutation of steps leading from q to q'. So, for all transition (q_i, U_i, q_{i+1}), $1 \leq i \leq k$, we have $q_i, q_{i+1} \notin r$, and, by **R3**, which is satisfied for arcs of $\mathfrak{ts}'$, we have $|U_i \cap out| = 0$ for $i = 1 \ldots k$. As a consequence $|U \cap out| = 0$, a contradiction.

So, **R3** is satisfied for (q, U, q') and $\mathfrak{r}$.
Case 4: **R4** does not hold for (q, U, q') and $\mathfrak{r}$. The proof is similar to the proof of Case 3 and also leads to a contradiction. So, **R4** is satisfied for (q, U, q') and $\mathfrak{r}$. □

To check the conditions stated in Proposition 1 would be computationally expensive. However, there is a class of step transition systems, for which they are always satisfied as shown in the Theorem 2. But first, we introduce a concept of *co-located sequential events* in the context of step transition systems.

Definition 6. *Let $\mathfrak{ts} = (Q, A, q_0)$ be an ENL-transition system w.r.t. a co-location relation $\backsimeq$ and E be a set of events used in the steps labelling transitions from A. We say that $\backsimeq$ partitions the set of events E into sets of* co-located sequential events*, $E = E_1 \uplus \ldots \uplus E_m$, if the following holds: for all $i = 1, \ldots, m$ there is no $q \in Q$ and no step U in $\mathfrak{ts}$ such that $e, f \in E_i$ and $e, f \in U$ and $q \xrightarrow{U}$.*

Theorem 2. *Let $\mathfrak{ts} = (Q, A, q_0)$ be an ENL-transition system w.r.t. a co-location relation $\backsimeq$, which partitions its set of events E into sets of co-located sequential events ($E = E_1 \uplus \ldots \uplus E_m$). Then all thick arcs in $\mathfrak{ts}$ satisfy the conditions of Proposition 1.*

Proof. From the fact that $\backsimeq$ partitions the set of events into sets of co-located sequential events we have that every non-singleton step of $\mathfrak{ts}$ can contain only one event from any individual sub-set E_i, $i = 1, \ldots, m$ and therefore, there is no non-singleton step that can be executed as a consequence of the local maximal concurrency semantics (enforced concurrency). Hence, the sets of resource enabled steps and control enabled steps of $\mathfrak{ts}$ are the same. That in turn means that $\mathfrak{ts}$, which is an ENL-transition system, is a step transition system of some EN-system and in this class of nets the step monotonicity property (see Fact 1) is satisfied for every (resource) enabled step. So, for every thick arc of $\mathfrak{ts}$ the conditions of Proposition 1 are satisfied. □

3.3 Reducing the number of potential *in* and *out* sets of regions

For a given step transition system $\mathfrak{ts} = (Q, A, q_0)$, finding its regions involve finding the *in* and *out* sets of events that are part of their definition. To reduce the execution time of generating all the non-trivial regions we can look at the possibilities of minimising the number of *in* and *out* sets to be considered. For a transition system that has n events ($|E| = n$), the brute-force approach would consider 2^n sets for *in* and *out* sets. However, looking at transitions of $\mathfrak{ts}$ and their labels, we can eliminate safely many candidates for *in* and *out* sets as stated in the following propositions.

Proposition 2. *Let $\mathfrak{r} = (in, r, out)$ be a region in a step transition system $\mathfrak{ts} = (Q, A, q_0)$. If the target of any transition $q \xrightarrow{U'} q'$ in $\mathfrak{ts}$, where $e \in U'$, is the source of another transition $q' \xrightarrow{U''} q''$ in $\mathfrak{ts}$, where $f \in U''$ and $f \neq e$, then the following hold:*

1. $\{e, f\} \not\subseteq in$;
2. $\{e, f\} \not\subseteq out$.

Proof. 1. Assume, to the contrary, that we have two transitions in $\mathfrak{ts}$, $q \xrightarrow{U'} q'$ and $q' \xrightarrow{U''} q''$ and two events e and f, such that $e \neq f$, $e \in U'$ and $f \in U''$, and $\{e, f\} \subseteq in$ for a region $\mathfrak{r} = (in, r, out)$. Hence, since regional condition **R4** is satisfied for both transitions and $\mathfrak{r}$, we have: $q \notin r$ and $q' \in r$ and $q' \notin r$ and $q'' \in r$. We obtained a contradiction and so $\{e, f\} \not\subseteq in$.
2. The proof is similar, but we use **R3** regional condition here rather than **R4**.

□

Corollary 1. *Let $\mathfrak{r} = (in, r, out)$ be a region in a step transition system $\mathfrak{ts} = (Q, A, q_0)$. If the target of any transition $q \xrightarrow{\{e\}} q'$ in $\mathfrak{ts}$ is the source of another transition $q' \xrightarrow{\{f\}} q''$ in $\mathfrak{ts}$, for $f \neq e$, then $\{e, f\} \not\subseteq in$ and $\{e, f\} \not\subseteq out$.*

Proof. Follows directly from Proposition 2. □

Proposition 3. *Let $\mathfrak{r} = (in, r, out)$ be a region in a step transition system $\mathfrak{ts} = (Q, A, q_0)$, and let $q \xrightarrow{U} q'$ be a transition in $\mathfrak{ts}$, such that there are two events $e, f \in U$. Then the following hold:*

1. $\{e, f\} \not\subseteq in$;
2. $\{e, f\} \not\subseteq out$.

Proof. 1. Assume, to the contrary, that we have a transition $q \xrightarrow{U} q'$ in $\mathfrak{ts}$, where $e, f \in U$ and $\{e, f\} \subseteq in$. Hence, since regional condition **R2** is satisfied in $\mathfrak{ts}$ for $q \xrightarrow{U} q'$ and $\mathfrak{r}$, we have that $\neg(q \notin r$ and $q' \in r)$ is true, which means that either $q \in r$ or $q' \notin r$. However, as **R4** regional condition is also satisfied for $q \xrightarrow{U} q'$ and $\mathfrak{r}$, we have $q \notin r$ and $q' \in r$, but this contradicts the previous conclusion.
2. The proof is similar, but we use **R1** and **R3** regional conditions here rather than **R2** and **R4**.

□

3.4 Using source and target sets

After selecting the candidates for the sets *in* and *out* we need to turn our attention to discovering the sets of states (r) for regions of the form: $\mathfrak{r} = (in, r, out)$. When doing so we need to look at the possibilities of reducing the number of these sets (see Section 3.1). In the context of standard transition systems, where arcs are labelled by single events, these sets would solely define regions. So, although we are considering step transition systems in this paper rather than the standard transition systems, some of the ideas developed earlier can be re-used in our new setting. For this part of the algorithm, we use the ideas of excitation and switching regions introduced in [9, 10, 15], where an excitation region for an event e is the maximal set of states, which are the sources of transitions labelled by e, while a switching region for an event e is the maximal set of states, which are targets of transitions labelled by e. We generalise these ideas to our setting of step transition systems. Also, we take advantage of the fact that step transition systems contain explicit information about the concurrency of events. We use this information, as well as the information about the causality of events, to select potential *in* and *out* sets (see Section 3.3), which are going to be useful in this step of the algorithm. In what follows we will call excitation regions the source sets and switching regions the targets sets in the context of step transition systems.

Definition 7. *Let $\mathfrak{ts} = (Q, A, q_0)$ be a step transition system on E. We define the following two sets of states for every $e \in E$:*

- *The* source set $S_e = \{q \in Q \mid \exists q' \in Q : (q, U, q') \in A \ \wedge \ e \in U\}$.
- *The* target set $T_e = \{q' \in Q \mid \exists q \in Q : (q, U, q') \in A \ \wedge \ e \in U\}$.

Proposition 4. *Let $\mathfrak{r} = (in, r, out)$ be a region in a step transition system on E, $\mathfrak{ts} = (Q, A, q_0)$, and let $e \in E$. Then the following is satisfied:*

1. *If $e \in out$ then $S_e \subseteq r$ and $T_e \cap r = \varnothing$.*
2. *If $e \in in$ then $T_e \subseteq r$ and $S_e \cap r = \varnothing$.*

Proof. Let $(q, U, q') \in A$ be a transition in $\mathfrak{ts}$ and $e \in U$. Hence $q \in S_e$ and $q' \in T_e$.

1. If $e \in out$ then $e \in U \cap out$ and from the definition of a region (**R3**) we have: $q \in r$ and $q' \notin r$. So, $S_e \subseteq r$ and $T_e \cap r = \varnothing$.
2. If $e \in in$ then then $e \in U \cap in$ and from the definition of a region (**R4**) we have: $q \notin r$ and $q' \in r$. So, $T_e \subseteq r$ and $S_e \cap r = \varnothing$.

□

If we want to combine the discovery of sets *in*, *out* and r, when searching for all non-trivial regions of step transition systems, we can use the results given in Corollary 2, Propositions 5 and Corollary 3.

Corollary 2. *Let $\mathfrak{r} = (in, r, out)$ be a region in a step transition system on E: $\mathfrak{ts} = (Q, A, q_0)$. Then*

$$\bigcup_{e \in out} S_e \ \cup \ \bigcup_{e \in in} T_e \subseteq r \ .$$

Proof. Follows directly from Proposition 4. □

From Corollary 2, we see that the set of states r of any region $\mathfrak{r} = (in, r, out)$ in a step transition system $\mathfrak{ts} = (Q, A, q_0)$ can be represented as

$$r = \bigcup_{e \in out} S_e \cup \bigcup_{e \in in} T_e \cup F^{\mathfrak{r}}, \tag{1}$$

where $F^{\mathfrak{r}}$ is a set of states we will call the *filler set* for $\mathfrak{r}$. We will use the following denotations: $S^{\mathfrak{r}} = \bigcup_{e \in out} S_e$ and $T^{\mathfrak{r}} = \bigcup_{e \in in} T_e$.

Proposition 5. *Let $\mathfrak{r} = (in, r, out)$ be a region in a step transition system $\mathfrak{ts} = (Q, A, q_0)$ and $(q, U, q') \in A$. Then the following hold:*

1. *If $q \in r$ and $U \cap out = \varnothing$ then $q' \in r$.*
2. *If $q \notin r$ and $U \cap in = \varnothing$ then $q' \notin r$.*

Proof. 1. Follows directly from **R1** of the definition of a region.
2. Follows directly from **R2** of the definition of a region.

□

Corollary 3. *Let $\mathfrak{r} = (in, r, out)$ be a region in a step transition system $\mathfrak{ts} = (Q, A, q_0)$ and $(q, U, q') \in A$, where $U \subseteq E \setminus (in \cup out)$. Then the following hold:*

1. *If $q \in S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$ and $q' \notin S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$ then $q' \in F^{\mathfrak{r}}$.*
2. *If $q' \in S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$ and $q \notin S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$ then $q \in F^{\mathfrak{r}}$.*

Proof. Follows from the representation of r (see Eq. (1)) and Proposition 5. □

3.5 An algorithm for generating regions of ENL-transition systems

Before we propose an algorithm for computing regions of ENL-transition systems, we summarise a few useful facts:

1. For every region $\mathfrak{r} = (in, r, out)$ in a step transition system $\mathfrak{ts}$ we have: $in \cap out = \varnothing$. This follows from **R3** and **R4** of the definition of a region.
2. For any non-trivial region $\mathfrak{r} = (in, r, out)$ in a step transition system $\mathfrak{ts}$ we have: $S^{\mathfrak{r}} \cup T^{\mathfrak{r}} \neq \varnothing$. This follows from the definition of a region, which states that the sets *in* and *out* can only be both empty for trivial regions.
3. The set of potential *in* sets is the same as the one of *out* sets, as the *in* set of a region $\mathfrak{r}$ is the *out* set of its complement, $\bar{\mathfrak{r}}$, and the other way round.

Although the title of this section suggests that the input for the algorithm should be an ENL-transition system, the algorithm can take as its input just a step transition system as defined in Definition 1, and it will compute regions as defined in Definition 4. The definition of a region that is used implies the target class of nets and the synthesis problem, in which the regions are later used as conditions to build the synthesised net. The algorithm does not check the axioms **A1**-**A4** (see Definition 5) to decide whether the input step transition

Algorithm 1: Extract $\mathfrak{R}_{\mathfrak{ts}}$

Function *Extract*$\mathfrak{R}_{\mathfrak{ts}}$*(Step transition system* $\mathfrak{ts} = (Q, A, q_0)$*)*
  Initialise $\mathfrak{R}_{\mathfrak{ts}}$ to empty set
  InOut = Generate All Potential *in*/*out* Sets($\mathfrak{ts}$) ; `// |InOut| = n`
  for *i:= 1 to n* **do**
    for *j:= 1 to n* **do**
      Initialise $S^{\mathfrak{r}}$ and $T^{\mathfrak{r}}$ to empty sets
      `/* Consider` $(in_i, out_j) \in InOut \times InOut$`, and not`
        $(out_j, in_i) \in InOut \times InOut$ `*/`
      if $in_i \cap out_j = \varnothing$ *and* $in_i \cup out_j \neq \varnothing$ **then**
        for *every event* $e \in out_j$ **do**
          Generate set S_e
          Add S_e to $S^{\mathfrak{r}}$
        for *every event* $e \in in_i$ **do**
          Generate set T_e
          Add T_e to $T^{\mathfrak{r}}$
      else
        break ; `//` (in_i, out_j) `is not valid`
      Initialise S to $S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$; `// Initial states to consider`
      Initialise r to $S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$
      for *every* $q \in S$ **do**
        Initialise $F_{current}$ to empty set
        for *every arc* $(q, U, q') \in A$ **do**
          if $(in_i \cup out_j) \cap U = \varnothing$ *and* $q' \notin r$ **then**
            Add q' to r
            Add q' to $F_{current}$
        for *every arc* $(q', U, q) \in A$ **do**
          if $(in_i \cup out_j) \cap U = \varnothing$ *and* $q' \notin r$ **then**
            Add q' to r
            Add q' to $F_{current}$
        Set S to $F_{current}$
      Let $\mathfrak{r} = (in_i, r, out_j)$
      if $\mathfrak{r}$ *satisfies regional axioms* **then**
        Add $\mathfrak{r}$ to $\mathfrak{R}_{\mathfrak{ts}}$
        Add $\bar{\mathfrak{r}} = (out_j, Q \setminus r, in_i)$ to $\mathfrak{R}_{\mathfrak{ts}}$
  return $\mathfrak{R}_{\mathfrak{ts}}$

system is indeed an ENL-transition system and therefore synthesisable. In most of our experiments, however, we used ENL-transition systems as inputs for the algorithm for the testing purposes (see Section 3.6).

To explain the algorithm we can look at a step transition system of Figure 4(a). In the first step of the algorithm we obtain all potential *in* and *out* sets

using the techniques from Section 3.3. We can consider one such a pair: $in = \varnothing$ and $out = \{h\}$. This pair trivially satisfies the validity constraints as stated in Section 3.3. The algorithm now discovers r for a candidate region $\mathfrak{r} = (\varnothing, r, \{h\})$. The set r is defined as $r = S^{\mathfrak{r}} \cup T^{\mathfrak{r}} \cup F^{\mathfrak{r}} = \bigcup_{e \in out} S_e \cup \bigcup_{e \in in} T_e \cup F^{\mathfrak{r}}$ (see Equation (1)). First the algorithm computes the set $S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$, in this case containing a single element q_3. The most complex part of the algorithm is the one to compute the filler set $F^{\mathfrak{r}}$ (using Corollary 3). The set $F^{\mathfrak{r}}$ is initially empty. It does not feature in the algorithm explicitly. So, initially, $r = S = S^{\mathfrak{r}} \cup T^{\mathfrak{r}} = \{q_3\}$. We start the for-loop at line 19 (see lines 19-29). Below we go through its iterations.

1. $S = \{q_3\}$ and $F_{current} = \varnothing$ (this set records the states added to r during the current iteration of the loop). We consider all possible arcs adjacent to q_3 that are not labelled by steps containing events from $in \cup out = \{h\}$. We have two arcs to consider:
   - Arc $q_3 \xrightarrow{\{g\}} q_4$: we add q_4 to r to 'bury' this arc in r. Now we have: $F_{current} = \{q_4\}$ and $r = \{q_3, q_4\}$.
   - Arc $q_2 \xrightarrow{\{e,f\}} q_3$: we add q_2 to r to 'bury' this arc in r. Now we have: $F_{current} = \{q_4, q_2\}$ and $r = \{q_3, q_4, q_2\}$.
2. $S = \{q_2, q_4\}$ and $F_{current} = \varnothing$ and we consider all possible arcs adjacent to q_2 and q_4 that are not labelled by steps containing events from $in \cup out = \{h\}$ and are not yet 'buried' in r. We have two arcs to consider:
   - Arc $q_4 \xrightarrow{\{e\}} q_5$: we add q_5 to r to 'bury' this arc in r. Now we have: $F_{current} = \{q_5\}$ and $r = \{q_2, q_3, q_4, q_5\}$.
   - Arc $q_1 \xrightarrow{\{g\}} q_2$: we add q_1 to r to 'bury' this arc in r. Now we have: $F_{current} = \{q_5, q_1\}$ and $r = \{q_3, q_4, q_5, q_2, q_1\}$.
3. $S = \{q_5, q_1\}$ and $F_{current} = \varnothing$ and we consider all possible arcs adjacent to q_1 and q_5 that are not labelled by steps containing events from $in \cup out = \{h\}$ and are not yet 'buried' in r. We have one arc to consider:
   - Arc $q_0 \xrightarrow{\{e\}} q_1$: we add q_0 to r to 'bury' this arc in r. Now we have: $F_{current} = \{q_0\}$ and $r = \{q_3, q_4, q_5, q_2, q_1, q_0\}$.
4. $S = \{q_0\}$ and $F_{current} = \varnothing$. In this iteration we cannot add any more states to r that are not already there, so S is set to $\varnothing$ for the next iteration of the loop. The computation of r is completed.

The discovered candidate for a region is $\mathfrak{r} = (\varnothing, \{q_0, q_1, q_2, q_3, q_4, q_5\}, \{h\})$. The algorithm then checks the regional axioms for the candidate and, if they are satisfied, the region and its complement (in this case $\overline{\mathfrak{r}} = (\{h\}, \{q_6, q_7\}, \varnothing)$) are added to the set of discovered regions. These are two out of ten non-trivial regions of $\mathfrak{ts}_2$ in Figure 4(a) discovered by Algorithm 1 (see the screenshot from WORKCRAFT, for this example, in Figure 7, in the Appendix).

3.6 Results of the experiments

The machine used in the experiments was PC with 3.20 GHz Intel Core i7 CPU and 12GB RAM, running Windows 10 Pro. The algorithms were implemented in Java (JDK 1.8.0) on top of the WORKCRAFT framework (version 3.2.6).

The average execution time of each algorithm was calculated on the basis of 20 runs. All experiments reported in this paper were conducted in isolation in order to prevent any side effects caused by concurrently executing processes. We ran each of the experiments over approximately the same time period to ensure that the computer was placed under similar load. The same machine was used for conducting all the experiments to ensure fair performance comparisons.

Using an example of the ENL-transition system in Figure 2, $\mathfrak{ts}_3$, we now compare the execution time taken to derive regions when applying five different approaches described as follows:

Approach 1: We generate all sets of 2^Q and all sets of 2^E; we check regional axioms for all arcs of A.
Approach 2: We generate all sets of 2^Q and all sets of 2^E; we check regional axioms for selected arcs of A, after ignoring the thick arcs (see Theorem 2).
Approach 3: We generate all sets of 2^Q; we consider only a selection of sets from 2^E to generate potential *in* and *out* sets by using the results from Section 3.3; we check regional axioms for all arcs of A.
Approach 4: We generate only a selection of sets from 2^Q by using the source and target sets; we consider only a selection of sets from 2^E to generate potential *in* and *out* sets by using the results from Section 3.3; we check regional axioms for all arcs of A.
Approach 5: We generate only a selection of sets from 2^Q by using the source and target sets; we consider only a selection of sets from 2^E to generate potential *in* and *out* sets by using the results from Section 3.3; we check regional axioms for selected arcs of A, after ignoring the thick arcs.

All the algorithms, related to the five approaches described above, produced the same non-trivial regions for $\mathfrak{ts}_3$. Figure 5 shows the results of the comparison, where clearly the best savings are gained by reducing the number of potential *in* and *out* sets by using the results presented in Section 3.3.

To test Algorithm 1, we selected the following examples: $\mathfrak{ts}_1$ in Figure 8 (2 states, 4 events), $\mathfrak{ts}_2$ in Figure 4 (8 states, 4 events), $\mathfrak{ts}_3$ in Figure 2 (16 states, 6 events), $\mathfrak{ts}_4$ in Figure 9 (12 states, 6 events), the step transition system of an ENL-system in Figure 10 ($\mathfrak{ts}_5$). The last transition system has 64 states and 9 events. The results of the experiments are shown in Figure 6.

When testing Algorithm 1, our first aim was to check whether it generates correctly all the expected regions. We selected our examples for testing trying to consider ENL-transition systems representing nets with different characteristics: thin step transition systems ($\mathfrak{ts}_2$, $\mathfrak{ts}_3$, $\mathfrak{ts}_5$) or not thin ($\mathfrak{ts}_1$, $\mathfrak{ts}_4$); step transition systems of nets with conflicts ($\mathfrak{ts}_1$, $\mathfrak{ts}_2$) or without conflicts ($\mathfrak{ts}_3$, $\mathfrak{ts}_4$, $\mathfrak{ts}_5$); step transition systems of nets, where every locality represents a sequential subsystem ($\mathfrak{ts}_3$, $\mathfrak{ts}_5$) or not ($\mathfrak{ts}_1$, $\mathfrak{ts}_2$, $\mathfrak{ts}_4$). For all the chosen examples the algorithm produced the expected results. The first four examined transition systems were very small. The last step transition system, $\mathfrak{ts}_5$, was bigger. As it has 64 states it was already too big to test the brute-force version of the algorithm on it, but Algorithm 1 produced its non-trivial regions in terms of milliseconds. We

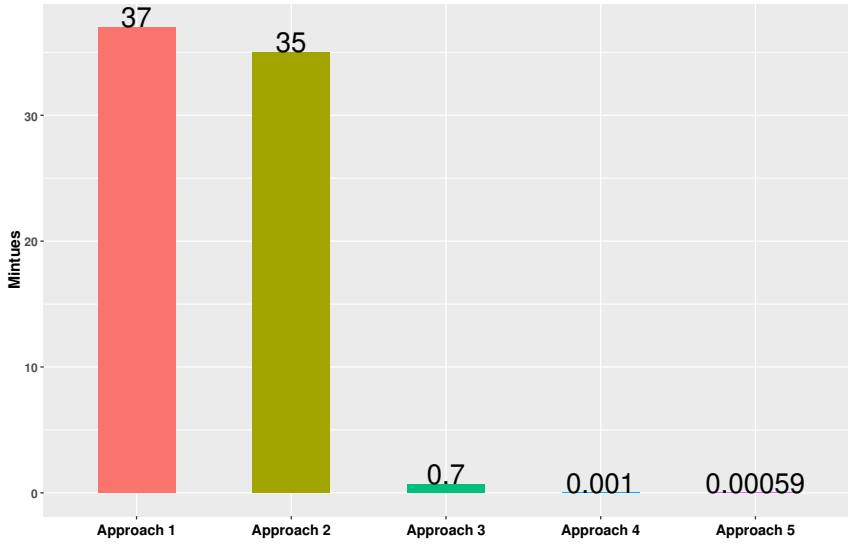

Fig. 5: A diagram showing the execution time taken to derive non-trivial regions of the ENL-transition system $\mathbb{ts}_3$ in Figure 2 when applying the five different approaches.

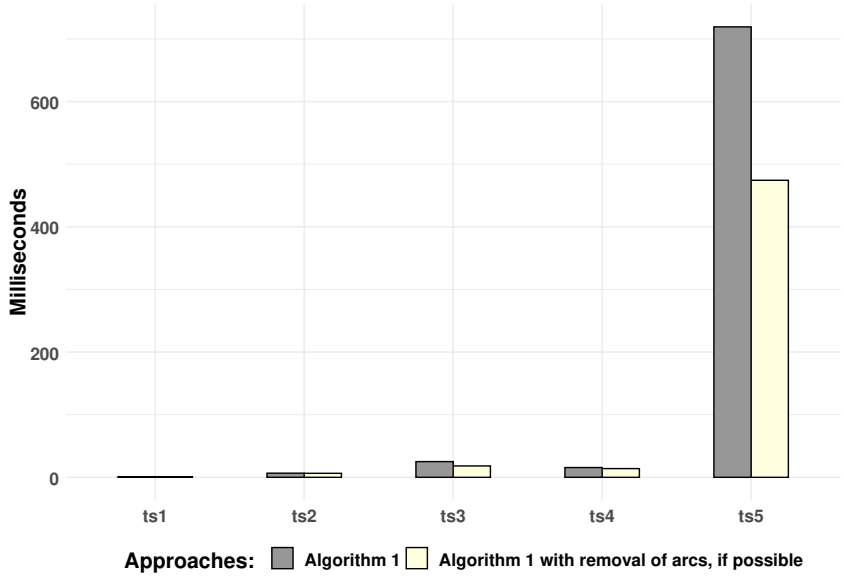

Fig. 6: A diagram showing the execution time taken to derive non-trivial regions of the ENL-transition systems: $\mathbb{ts}_1$, $\mathbb{ts}_2$, $\mathbb{ts}_3$, $\mathbb{ts}_4$ and $\mathbb{ts}_5$, when using Algorithm 1 without (or with) the removal of thick arcs, respectively

tested Algorithm 1 on ENL-transition systems with up to 90 states and it still performed very well. We found this promising.

The removal of thick arcs, applicable to $\mathfrak{ts}_3$, $\mathfrak{ts}_4$ and $\mathfrak{ts}_5$, gave the best results, in terms of execution time, for transition systems $\mathfrak{ts}_3$ and $\mathfrak{ts}_5$ as they satisfy the conditions of Theorem 2 allowing for the removal of all thick arcs.

4 Conclusions

In this paper we presented an algorithm for deriving non-trivial regions of ENL-transition systems. The algorithm was tested on a selection of small step transition systems. We plan to build a set of benchmarks to test its performance on bigger examples and see how it scales with the increasing sizes of inputs.

In the future, we plan to add new algorithms to our tool to produce a complete tool for the synthesis of ENL-systems, including algorithms for the minimization of the synthesised nets; and algorithms for synthesising ENL-systems with the assumption that the co-location relation is not known in advance and needs to be discovered as a part of solving the synthesis problem.

Acknowledgement The first author is grateful to the National Transitional Council of Libya for funding her PhD studentship and research.

References

1. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Rozinat, A., Weijters, T.: ProM: the process mining toolkit. In: de Medeiros, A.K.A., Weber, B. (eds.): Proc. of the Business Process Management Demonstration Track (BPMDemos 2009) vol. **489** CEUR
2. Badouel, E., Bernardinello, L., Darondeau, Ph.: The Synthesis Problem for Elementary Net Systems is NP-complete. Theoretical Computer Science **186** (1997) 107–134
3. Badouel, E., Bernardinello, L., Darondeau, P.: Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series. Springer (2015)
4. Badouel, E., Darondeau, Ph.: Theory of Regions. In: Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets I: Basic Models, Advances in Petri Nets. Lecture Notes in Computer Science **1491**. Springer-Verlag, Berlin Heidelberg New York (1998) 529–586
5. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Synthesis of Petri Nets from Scenarios with VipTool. Lecture Notes in Computer Science **5062**, Springer (2008) 388–398
6. Bernardinello, L., De Michelis, G., Petruni, K., Vigna, S.: On the Synchronic Structure of Transition Systems. In: Structures in Concurrency Theory, J.Desel (ed.) (1995) 69–84
7. Bernardinello, L.: Synthesis of Net Systems. In: Marsan, M.A. (ed.): Application and Theory of Petri Nets 1993. Lecture Notes in Computer Science **691**. Springer-Verlag, Berlin Heidelberg New York (1993) 89–105

8. Carmona, J., Cortadella, J., Kishinevsky, M.: Genet: A Tool for the Synthesis and Mining of Petri Nets. Proc. of ACSD'09, IEEE Computer Society (2009) 181–185
9. Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. IEICE Transactions on information and Systems **80** (1997) 315–325.
10. Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Logic Synthesis of Asynchronous Controllers and Interfaces. Springer (2002)
11. Darondeau, P., Koutny, M., Pietkiewicz-Koutny, M., Yakovlev, A.: Synthesis of Nets with Step Firing Policies. In: van Hee, K.M., Valk, R. (eds.): PETRI NETS 2008. Lecture Notes in Computer Science **5062**. Springer-Verlag, Berlin Heidelberg New York (2008) 112–131
12. Dasgupta, S., Potop-Butucaru, D., Caillaud, B., Yakovlev, A.: Moving from Weakly Endochronous Systems to Delay-Insensitive Circuits. Electronic Notes in Theoretical Computer Science **146** (2006) 81–103
13. Desel, J., Reisig, W.: The Synthesis Problem of Petri Nets. Acta Informatica **33** (1996) 297–315
14. Ehrenfeucht, A., Rozenberg, G.: Partial 2-structures; Part I: Basic Notions and the Representation Problem, and Part II: State Spaces of Concurrent Systems. Acta Informatica **27** (1990) 315–368
15. Kishinevsky, M., Kondratyev, A., Taubin, A., Varshavsky, V., Yakovlev, A., Napelbaum, E. and Reva, O.: Concurrent hardware: the theory and practice of self-timed design. John Wiley & Sons, Inc., (1994).
16. Kleijn, H.C.M., Koutny, M., Rozenberg, G.: Towards a Petri Net Semantics for Membrane Systems. In: Freund, R., Paun, G., Rozenberg, G., Salomaa, A. (eds.): WMC 2005. Lecture Notes in Computer Science **3850**. Springer-Verlag, Berlin Heidelberg New York (2006) 292–309
17. Koutny, M., Pietkiewicz-Koutny, M.: Transition Systems of Elementary Net Systems with Localities. In: Baier, C., Hermanns, H. (eds.): CONCUR 2006. Lecture Notes in Computer Science **4137**. Springer-Verlag, Berlin Heidelberg New York (2006) 173–187
18. Koutny, M., Pietkiewicz-Koutny, M.: Synthesis of Elementary Net Systems with Context Arcs and Localities. Fundamenta Informaticae **88** (2008) 307–328
19. Koutny, M., Pietkiewicz-Koutny, M.: Synthesis of Petri Nets with Localities. Scientific Annals of Computer Science **19** (2009) 1–23
20. Koutny, M., Pietkiewicz-Koutny, M.: Minimal regions of ENL-transition systems. Fundamenta Informaticae **101(1-2)** (2010) 45–58.
21. Mukund, M.: Petri Nets and Step Transition Systems. International Journal of Foundations of Computer Science **3** (1992) 443–478
22. Nielsen, M., Rozenberg, G., Thiagarajan, P.S.: Elementary Transition Systems. Theoretical Computer Science **96** (1992) 3–33
23. Păun, G.: Membrane Computing, An Introduction. Springer-Verlag, Berlin Heidelberg New York (2002)
24. Pietkiewicz-Koutny, M.: The Synthesis Problem for Elementary Net Systems with Inhibitor Arcs. Fundamenta Informaticae **40** (1999) 251–283
25. Poliakov, I., Khomenko, V., Yakovlev, A.: Workcraft - a framework for interpreted graph models. In: Franceschinis, G., Wolf, K. (eds.): Applications and Theory of Petri Nets 2009. Lecture Notes in Computer Science **5606**. Springer-Verlag, Berlin Heidelberg (2009) 333–342
26. Solé, M., Carmona, J.: Rbminer: A Tool for Discovering Petri Nets from Transition Systems. Lecture Notes in Computer Science **6252**, Springer (2010) 396–402
27. Workcraft. https://workcraft.org/ (2018)

Appendix

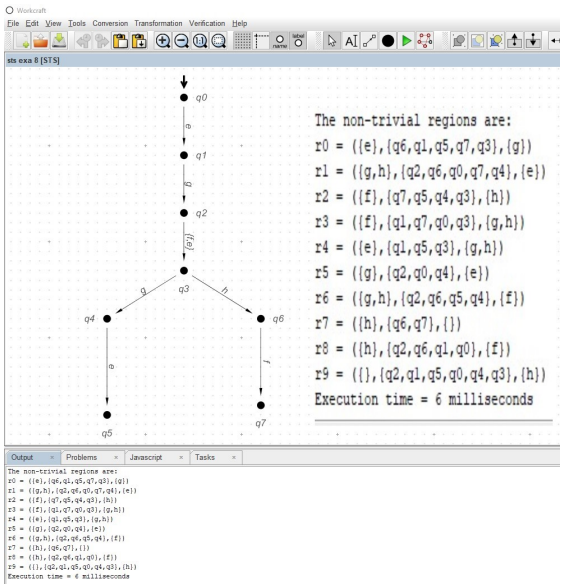

Fig. 7: An execution time and $\mathfrak{R}_{\mathfrak{ts}}$ of the ENL-transition system $\mathfrak{ts}_2$ in Figure 4(a) as shown in WORKCRAFT.

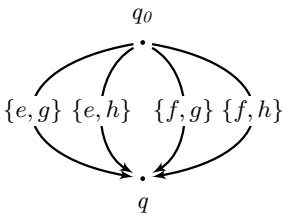

Fig. 8: An ENL-transition system $\mathfrak{ts}_1$, where e, f, g and h are co-located events.

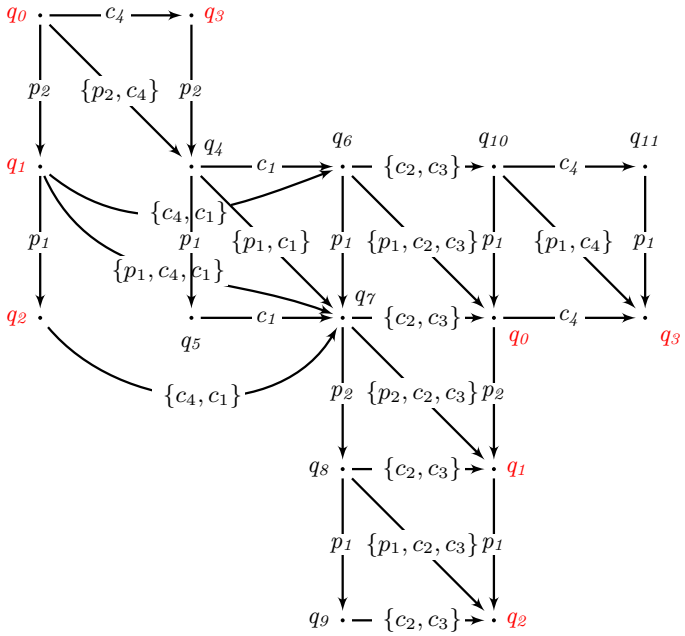

Fig. 9: An ENL-transition system $\mathfrak{ts}_4$, where p_1 and p_2 are co-located events, and c_1, c_2, c_3 and c_4 are co-located events. The graph should be glued on the states q_0, q_1, q_2, q_3 that appear twice in the picture. $\mathfrak{ts}_4$ is isomorphic to the step transition system of the ENL-system in Figure 1.

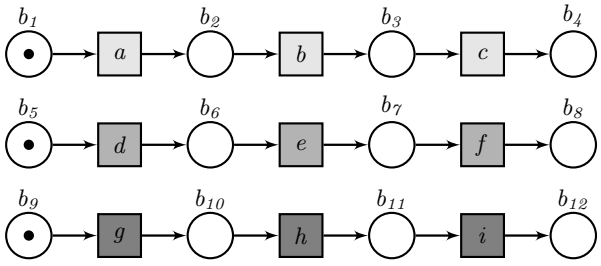

Fig. 10: An ENL-system, where events a, b, c are sharing a location; events d, e, f are in the second location; and events g, h and i are in the third location. It generates $\mathfrak{ts}_5$ step transition system considered in Figure 6.