Algorithms and Tool Support for the Synthesis of Elementary Net Systems with Localities

Aishah Ahmed

School of Computing
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

Computing Science July 2023

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

This thesis includes some works of a paper [4] that has been published in the International Workshop on Petri Nets and Software Engineering 2020 (PNSE'20), a paper [3] that has been published in the Theoretical Computer Science journal 2022, and a paper [5] that has been published in the International Workshop ATAED (Algorithms & Theories for the Analysis of Event Data) 2022.

Aishah Ahmed
July 2023

Acknowledgements

I would like to say a special thank you to my supervisor, Marta Pietkiewicz-Koutny. Her support, guidance and overall insights in this field have made this an inspiring experience for me. She continuously provided encouragement and was always willing and enthusiastic to assist in any way she could throughout the research project. Also, words cannot even come close to expressing my gratitude to my family, especially my husband Ibrahim, who deserve the credit for whatever positive that I have achieved in my life. They have always supported me in all my endeavours and have patiently waited for the completion of my doctorate.

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 a shortage of implemented tools capable of dealing with complex real-life system construction. In this research project, we focus on developing algorithms and tool support for the synthesis of ENL-systems from step transition systems (ST-systems), where arcs are labelled by steps (sets) of executed actions. We present an algorithm for deriving non-trivial regions of ST-systems, which is a fundamental algorithm for the synthesis of ENL-systems. We introduce two algorithms for verifying whether a given step transition system can be synthesised to an ENL-system. Also, we present an algorithmic solution to the synthesis problem for ENL/LC-systems - a special subclass of ENL-systems, where conflicts between events are localised. Then, we focus on the minimisation of the synthesised nets. In particular, we discuss the properties of minimal, companion, and complementary regions, and their role in the process of minimisation of ENL-systems. Furthermore, we propose a strategy to eliminate redundant regions. Our theoretical results are backed by experiments. The algorithms are implemented within the WORKCRAFT framework.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

A number of computational systems exhibit behaviour which can be best understood as 'globally asynchronous locally (maximally) synchronous'. Examples can be found in hardware design, where a VLSI chip may contain multiple clocks responsible for synchronising different subsets of gates [19, 22], and in biologically inspired membrane systems representing cells within which biochemical reactions happen in synchronised pulses [35]. To formalise such systems, [24] introduced *Place/Transition-nets with localities* (PTL-nets), where each locality defines a distinct set of events/actions which must be executed synchronously, i.e., in a maximally concurrent manner (often called *local maximal concurrency*).

The notion of locality is crucial in this research project. From the formal point of view the concept of locality can be introduced by defining a locality mapping, which assigns to each event its locality (a natural number), like it was done in [25]. Another possibility is to define a co-location relation over the set of events like it was done in [27]. In this research we adopted the approach of [27], which stipulates that a co-location relation must be an equivalence relation and then localities are the equivalence classes of the chosen co-location relation. In this second approach the localities are not explicitly 'named' like in the first approach, but it is stated here which events are co-located and which ones belong to different localities.

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 [21], and later used to solve the synthesis problem for many different classes of Petri nets [6, 9, 13, 12, 33, 34, 37]. A comprehensive, systematic survey of the synthesis problem and region theory is presented in [7].
A majority of results in the area of synthesis of Petri nets use the standard transition systems, where the transitions are labelled with single events/actions, as initial specifications of

systems' behaviour. In this research project, however, we follow the approach, used in [25–28, 37], employing step transition systems instead, where transitions are labelled with sets of executed events/actions.

Nets with localities, as already mentioned, were first introduced in [24] using as a base a class of Place/Transition nets (PT-nets). The idea of actions' localities was later adapted to elementary net systems (EN-systems) in [25], 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 [26–28]. The papers [25–27] suitably adapted the classical theory of regions [9] to cope with local maximal concurrency in the context of three different classes of nets, while [28] concentrated on finding the rules for reducing the number of regions that are essential to synthesise ENL-systems.

The above list of papers built up a theory for the synthesis of ENL-systems. In this research project, we developed algorithms and tool support for the synthesis of ENL-systems from step transition systems. We concentrated on implementing existing and new theoretical results into our tool. The new theoretical results were introduced to support and justify the optimisations used in the developed algorithms. In our research, we were concerned with finding efficient algorithms for the synthesis of ENL-systems. The most important among them is an algorithm for deriving regions which are used in the synthesis procedure. Also, we developed two algorithms for verifying whether a given step transition system can be synthesised to an ENL-system. Furthermore, we presented an algorithmic solution to the synthesis problem for ENL/LC-systems - a special subclass of ENL-systems, where conflicts between events are localised (conflicting events belong to the same locality). The synthesis procedure for the general class of ENL-systems assumes that the co-location relation for events is known in advance. However, for the subclass of ENL/LC-systems whose conflicts are localised, one can calculate their 'canonical' co-location relations as a part of the synthesis procedure [27]. Any other valid co-location relations [1] can be obtained from such a co-location relation. Note that the class of ENL/LC-systems is particularly important since distributed conflicts (conflicting events belong to different localities) cannot be implemented without prior modifications (see [8]). To further augment our tool with new algorithms, we continued the work started in [28] and focused on the minimisation of ENL-systems. The nets obtained from the synthesis procedure, called saturated nets, contain many conditions that are redundant from their behaviour point of view. Removing such conditions is important to get more manageable and readable solutions to the synthesis problem. The approaches to remove redundant conditions from nets were investigated in the literature and implemented in

[1]By a valid co-location relation we mean a co-location relation that can be successfully used in the synthesis procedure.

several tools [1, 11]. Many synthesis procedures concentrate on returning smaller solutions based on so-called minimal regions [10, 14, 16, 17, 32]. In our approach, minimal regions, as defined for our class of step transition systems, are also important for building smaller solutions to the synthesis problem. Furthermore, following [13], we were interested in the role of minimal regions in defining state-machine components of the synthesised and minimised ENL-systems. We observed/proved that the saturated ENL-systems that result from the synthesis procedure and their minimal versions with all non-minimal regions removed are state machine decomposable (built from a set of connected sequential subsystems).

The algorithms are implemented within the WORKCRAFT framework [39, 48]. Developing efficient algorithms for the synthesis of ENL-systems is a challenging problem (note that the problem is NP-complete which can be shown following the argument made in [6]). Also, there are no implemented tools for the synthesis of Petri nets from step transition systems. The existing ones, like PETRIFY [16, 23, 17], VIPTOOL [11], PROM [1], GENET [14] or RB-MINER [40], work with specifications that do not include the information about concurrency in the form of steps of simultaneously executed actions.

1.1 Elementary net systems with localities

To explain the basic idea behind ENL-systems, let us consider the net in Figure 1.1 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.

The local maximal concurrency semantics used by ENL-systems addresses a general problem of modelling synchronous behaviour of events within localities that is needed in both application areas of these systems: membrane systems and VLSI circuits. However, in the context of designing VLSI circuits some additional requirements might be also important. One of such requirements is to guarantee that the steps allowed by a Petri net model satisfy step persistence property as defined in [22] (no step which became enabled can subsequently

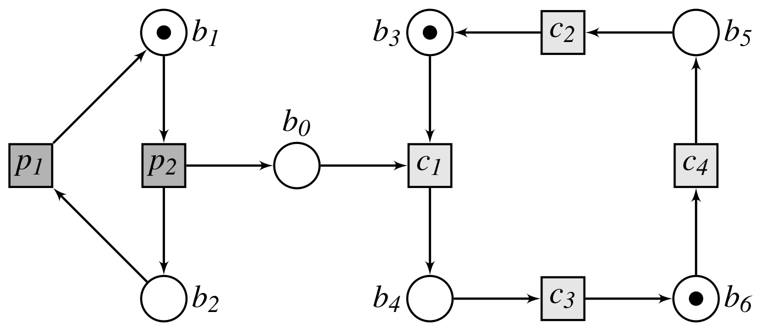


Fig. 1.1 A one producer/two co-located consumers system (shading of boxes indicates the co-location of events they represent).

be prevented from being executed by the occurrence of any other step). The local maximal concurrency semantics of ENL-systems is not strong enough to guarantee the step persistence property as can be seen in the example discussed above. The step $\{c_4\}$, which is enabled in the initial state, becomes disabled after the execution of the step $\{p_2\}$. Instead $\{c_1, c_4\}$ is enabled. In this thesis, we concentrate on ENL-systems with the local maximal concurrency semantics of execution and augmenting it to guarantee the satisfaction of the step persistence property is out of the scope of this thesis.

1.2 Automated synthesis of systems

Within the domain of rigorous system design, automated *synthesis* from behavioural specifications is an attractive and yet still underdeveloped approach of constructing computational systems. Synthesis procedures guarantee that the resulting systems are correct, and so the time consuming process of system verification becomes redundant. The main problem in the area of automated synthesis is a shortage of mature techniques and implemented tools capable of dealing with complex real-life system construction. In particular, there are no mature algorithms and implemented tools aimed at the synthesis of Boolean nets with localities from concurrent specifications. Addressing this issue is the overall goal of the proposed research project.

1.3 Tool support and implementation

The algorithms developed within this PhD project are implemented as tool support for the synthesis of ENL-systems. The tool is implemented as a Java plugin within a framework called WORKCRAFT. The WORKCRAFT is designed to provide a flexible framework for the development and analysis of interpreted graph models [39], including capturing, visual editing, simulation, synthesis, and verification of such models. There are several modules that have been implemented and supported by the WORKCRAFT framework so far. For instance, modules for Petri nets, Signal Transition Graph (STG) [52], Conditional Partial Order Graphs (CPOG) [31], and Structured Occurrence Nets (SON) [30]. The framework is built using a plugin-based architecture, which makes it easily extendible to new graph-based formalisms, as well as new analysis and verification modules. In addition, WORKCRAFT provides a GUI environment that facilitates model entry, supports interactive visual simulation, and convenient 'single-click' verification. Being such an extensible and flexible environment, WORKCRAFT allows researchers to define, model and analyse new model types [39, 38, 41]. Therefore, we decided to utilise WORKCRAFT to implement our algorithms and tool support for the synthesis of ENL-systems.

WORKCRAFT software engineers must be aware of the technologies that were used to develop this framework before embarking on making any further extensions to it. First of all, the WORKCRAFT framework requires Java JDK 8 or later version and the installation and inclusion of the WORKCRAFT library to a Java path. Furthermore, Eclipse IDE [46] was used as an integrated environment for developing and debugging WORKCRAFT graph models [48]. Taking these into consideration, it was possible to implement our proposed algorithms as WORKCRAFT plugins in such a development environment. There were some existing features in the WORKCRAFT plugins framework that we were able to utilise and adapt to implement our tool such as features for creating some components (conditions, transitions, arcs), for example. A part from that, our implemented tool for the synthesis of ENL-systems introduced new two plugins into the WORKCRAFT. As the synthesis procedure involves two different graph models: one plugin was developed to represent ENL-systems, and another one to represent step transition systems (ST-systems).

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 10 runs. All experiments reported in this thesis 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.

1.4 Main contributions

The results of this research project contribute to a wider field of the theory of synthesis of concurrent systems and focus on developing efficient algorithms for the synthesis of ENL-systems. The detailed contributions are listed below:

- Investigation and analysis of different efficiency measures for the implementation of algorithms to extract regions of ST-systems.
- Implementation of novel algorithms for extraction of regions of ST-systems based on the proved theoretical results.
- Experimental evaluation of the proposed and implemented efficiency measures and identification of an efficiency measure with highest gains.
- Design of the two algorithms for the synthesis of ENL-systems and experimental evaluation of their efficiency.
- Implementation of the existing theoretical results for the synthesis of ENL/LC-systems.
- Design of a strategy to eliminate redundant regions from the synthesised ENL-systems.
- Design and implementation of the algorithms for the minimisation of the synthesised nets.
- Investigation into state machine decomposability of the synthesised ENL-systems.
- Development and introduction of two new plugins into WORKCRAFT framework as a tool support for the synthesis of ENL-systems.

Some of the results contained in this Thesis were presented at two international workshops, [4, 5], and published as a journal article [3].

1.5 Outline of the thesis

The thesis is organised as follows. The next chapter recalls some basic notions concerning step transition systems, ENL-systems, the synthesis of ENL-systems, and summaries the related works. Chapter 3 introduces an algorithm for computing regions of a given step transition system and provides formal results to support the algorithm. Chapter 4 outlines two algorithms for verifying whether a given step transition system can be synthesised to an

$\text{ENL}_{\bumpeq}$-system and provides results to compare the efficiency of the two approaches for the synthesis procedure. Chapter 5 investigates ENL-systems with localised conflicts and outlines algorithms for computing ‘canonical’ co-location relations for them, from which any further valid co-location relations can be obtained. Chapter 6 discusses the minimisation of the solutions to the synthesis problems, and presents the experimental results. Chapter 7 presents the selected case studies that have been used throughout our research. Chapter 8 explains the WORKCRAFT framework and the implemented tool. Finally, Chapter 9 concludes the Thesis by giving directions for future work.

Chapter 2

Background and related works

2.1 Preliminaries

In this section, we recall suitably adapted notions and results from [25, 27, 28].

Throughout the thesis, E is a fixed finite nonempty set of *events*. A *step* is a nonempty set of events, and a *co-location relation* $\simeq$ is any equivalence relation over E. For every event $e \in E$, $[e]_{\simeq}$ is the equivalence class of $\simeq$ to which e belongs (i.e., the *locality* of e). For an event e and a step U, we denote $e \simeq U$ whenever there is at least one event $f \in U$ satisfying $e \simeq f$.

Definition 2.1.1. *A* step transition system *(or* ST*-system) on E is a triple* $\mathfrak{ts} \overset{\mathrm{df}}{=} (Q, A, q_0)$ *where Q is a nonempty finite set of* states*, $A \subseteq Q \times (\mathbb{P}(E) \setminus \{\varnothing\}) \times Q$ is a finite set of* transitions (arcs)*, and $q_0 \in Q$ is the* initial state [2]. $\diamondsuit$

For a transition $\mathfrak{t} = (q, U, q') \in A$, we will call q the *source* of t (q, U, q') and q' the *target* of t. Furthermore, $\mathfrak{t}$ is *thick* if $|U| \geq 2$.
Moreover, for every state q of the ST-system $\mathfrak{ts}$, we assume that:

- $allSteps_q$ is the set of all steps labelling transitions outgoing from q.
- $minSteps_q$ is the set of all minimal steps (w.r.t. set inclusion) belonging to $allSteps_q$.
- E_q is the union of all the steps in $allSteps_q$.

In diagrams, ST-systems are represented as if they were labelled directed graphs, and singleton steps annotating transitions are denoted without brackets ($e.g., e$ instead of $\{e\}$). To ease the presentation, we assume that each event of E occurs in at least one of the steps

[2]$\mathbb{P}(E)$ is the set of all subsets of E (the power set of a set E). It is also often denoted by 2^E.

labelling the transitions of $\mathfrak{ts}$.
$\mathfrak{ts}$ is called *thin* if, for every event $e \in E$, there is $(q,\{e\},q') \in A$.
A sequence of transitions $(q_1,U_1,q_2)(q_2,U_2,q_3)\dots(q_k,U_k,q_{k+1})$ is a *path* from q_1 to q_{k+1}. In such a case, $\sigma = U_1U_2\dots U_k$ is a *step sequence* from q_1 to q_{k+1}. A state q is *reachable* if there is a path from q_0 to q.
We say that two distinct events, e and f, are in *conflict* in $\mathfrak{ts}$ if there is $q \in Q$ such that $e,f \in E_q$ and, for all $U \in allSteps_q$, $\{e,f\} \not\subseteq U$. We denote this by $e \leftrightharpoons_{\mathfrak{ts}} f$ and $e \leftrightharpoons^{q}_{\mathfrak{ts}} f$ (if mentioning q is important).

Let $\mathfrak{ts} = (Q,A,q_0)$ and $\mathfrak{ts}' = (Q',A',q'_0)$ be two ST-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 $U \in \mathbb{P}(E) \setminus \{\varnothing\}$.

2.1.1 ENL-systems

Definition 2.1.2. *An* elementary net system with localities w.r.t. a co-location relation $\simeq$ *(or* ENL$_{\simeq}$*-system) is a tuple*

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

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, *and* $c_0 \subseteq B$ *is the* initial case *(in general, any subset of B is a* case*).*
For every event e, its pre-conditions *and* post-conditions *are given respectively by*

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

(both sets are assumed nonempty and disjoint).
We will also say that $\mathfrak{enl}$ *is an elementary net system with localities (or* ENL*-system) if mentioning* $\simeq$ *is not important.* ♢

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 1.1).

The dot-notation used in Definition 2.1.2 for pre- and post-conditions extends to sets of events in the usual way, e.g., $^{\bullet}U \overset{\text{df}}{=} \bigcup\{^{\bullet}e \mid e \in U\}$. Furthermore, two distinct events, e and f, are *in conflict* (or conflicting) if they share a pre-condition, or share a post-condition. We denote this by $e \leftrightharpoons_{\mathfrak{enl}} f$.
The semantics of $\mathfrak{enl}$ is based on steps of simultaneously executed events, and can be understood as *local maximal concurrency*. We first define *potential steps of* $\mathfrak{enl}$ as all nonempty sets of mutually 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 $e \simeq U$ and the step $U \cup \{e\}$ is resource enabled at c. We denote these respectively by $U \in \mathit{resenabled}(c)$ and $U \in \mathit{enabled}(c)$. A control enabled step $U \in \mathit{enabled}(c)$ can be *executed* leading from c to the case $c' = (c \setminus {}^\bullet U) \cup U^\bullet$. We denote this by $c[U\rangle c'$.
For example, for the ENL-system in Figure 1.1 with the initial case $c_0 = \{b_1, b_3, b_6\}$, we have $\mathit{resenabled}(c_0) = \mathit{enabled}(c_0) = \{\{p_2\}, \{c_4\}, \{p_2, c_4\}\}$. At c_0 we can execute the step $\{p_2\}$ moving to $c = \{b_0, b_2, b_3, b_6\}$ (i.e., $c_0[\{p_2\}\rangle c$). At the case c, $\mathit{enabled}(c) = \{\{p_1\}, \{c_1, c_4\}, \{p_1, c_1, c_4\}\}$. The steps containing c_1 and c_4 separately, like $\{c_1\}$, $\{c_4\}$, $\{p_1, c_1\}$ and $\{p_1, c_4\}$, are not control enabled at the case c due to local maximal concurrency semantics. They are, however, resource enabled at c as we have $\mathit{resenabled}(c) = \{\{c_1\}, \{c_4\}, \{p_1, c_1\}, \{p_1, c_4\}\} \cup \mathit{enabled}(c)$ (see also Figure 3.6, on p. 31).

The set of *reachable* cases of $\mathfrak{enl}$, denoted $\mathit{reach}_{\mathfrak{enl}}$, is the least set of cases containing c_0 such that if $c \in \mathit{reach}_{\mathfrak{enl}}$ and $c[U\rangle c'$, then $c' \in \mathit{reach}_{\mathfrak{enl}}$.

The ST-system *generated* by $\mathfrak{enl}$ is given by:

$$\mathfrak{ts}_{\mathfrak{enl}} \stackrel{\mathrm{df}}{=} (\mathit{reach}_{\mathfrak{enl}}, A, c_0),$$

where $A = \{(c, U, c') \mid c \in \mathit{reach}_{\mathfrak{enl}} \wedge c[U\rangle c'\}$. Note that its arcs are labelled only by control enabled steps (see Figure 3.6, on p. 31, depicting an ST-system that is isomorphic to the ST-system generated by the ENL-system in Figure 1.1, on p. 4).

$\mathfrak{enl}$ is a *net realisation* of an ST-system $\mathfrak{ts}$ if $\mathfrak{ts}_{\mathfrak{enl}} \cong \mathfrak{ts}$.

To ease the presentation, we assume that $\mathfrak{enl}$ does not have *dead events*, i.e., for each event e, there are $c \in \mathit{reach}_{\mathfrak{enl}}$ and $U \in \mathit{enabled}(c)$ such that $e \in U$.

Definition 2.1.3. *Let* $\mathfrak{enl} = (B, E, F, \simeq, c_0)$ *be an* ENL$_\simeq$*-system. We say that* $\mathfrak{enl}$ *is a* state machine ENL$_\simeq$*-system iff:*

1. $\forall e \in E : \; |{}^\bullet e| = 1 = |e^\bullet|$;
2. $|c_0| = 1$. $\diamondsuit$

Definition 2.1.4. *Let* $\mathfrak{enl} = (B, E, F, \simeq, c_0)$ *be an* ENL$_\simeq$*-system. A* subsystem *of* $\mathfrak{enl}$ *is an* ENL$_\simeq$*-system* $\mathfrak{enl}' = (B', E', F', \simeq', c'_0)$ *such that the following conditions hold:*

1. $B' \subseteq B$ *and* $E' \subseteq E$;
2. $\forall b \in B' \, \forall e \in E : \; ((b, e) \in F \vee (e, b) \in F) \implies e \in E'$;
3. $F' = F \cap ((B' \times E') \cup (E' \times B'))$; $\simeq' = \simeq \cap E' \times E'$ *and* $c'_0 = c_0 \cap B'$.

A subsystem $\mathfrak{enl}'$ *is* connected *if the graph* $(B' \cup E', F')$ *is connected. Also, point 2 above says that the subsystem* $\mathfrak{enl}'$ *is* generated by the subset of conditions B' *of the* ENL$_{\frown}$*-system* $\mathfrak{enl}$. ♢

Definition 2.1.5. *Let* $\mathfrak{enl} = (B,E,F,\frown,c_0)$ *be an* ENL$_{\frown}$*-system. We say that* $\mathfrak{enl}$ *is a* state machine decomposable ENL$_{\frown}$*-system iff there exists a set of connected subsystems of* $\mathfrak{enl}$, $\mathfrak{enl}_i = (B_i,E_i,F_i,\frown_i,c_0^i)$ *(*$i = 1,\dots,m$*), satisfying the following:*

1. $\forall i \in \{1,\dots,m\}$: $\mathfrak{enl}_i$ *is a state machine* ENL$_{\frown_i}$*-system;*

2. $B = \bigcup_i B_i$, $E = \bigcup_i E_i$ *and* $F = \bigcup_i F_i$.

The $\mathfrak{enl}_i$ *are called* state machine (or sequential) components *of* $\mathfrak{enl}$. ♢

Elementary net systems (or EN-systems) can be considered as ENL-systems where no two distinct events are co-located. The local maximal semantics of ENL-systems means that certain properties enjoyed by EN-systems do not hold for the general class of ENL-systems, e.g., the following *step monotonicity* property.

Fact 1. *Let* $\mathfrak{enl}$ *be such that no two distinct events are co-located. If* U *is a step enabled at a case* c *and* U' *is a nonempty proper subset of* U*, then there is a case* c' *such that* $c[U'\rangle c'$ *and* $U \setminus U' \in enabled(c')$. ♢

A resource enabled step of an ENL-system is also control enabled whenever it is covered by control enabled sub-steps.

Fact 2. *Let* $U, U' \in enabled(c)$ *and* $U \cup U' \in resenabled(c)$*. Then* $U \cup U' \in enabled(c)$. ♢

Furthermore, every resource enabled step of an ENL$_{\frown}$-system can be extended to a control enabled step.

Fact 3. *If* $U \in resenabled(c)$ *then there is* $W \in enabled(c)$ *such that* $U \subseteq W$ *and* $e \frown U$*, for every* $e \in W \setminus U$. ♢

The general synthesis problem considered in this thesis can be formulated in the following way.

Problem 1. *Given an* ST*-system* $\mathfrak{ts}$ *and a co-location relation* $\frown$*, find an effective way of checking whether there is an* ENL$_{\frown}$*-system which is a net realisation of* $\mathfrak{ts}$*. If the answer is positive construct such an* ENL$_{\frown}$*-system.* ♢

2.1.2 ENLST-systems

Problem 1 can be approached by considering a link between the nodes (global states) of an ST-system with the conditions (local states) of a hypothetical ENL-system realising it, captured by the notion of regions with explicit input and output events.

Definition 2.1.6. *A* region (with explicit input and output events) *of an* ST*-system* $\mathfrak{ts} = (Q, A, q_0)$ *is a triple*

$$\mathfrak{r} = (in, r, out) \in \mathbb{P}(E) \times \mathbb{P}(Q) \times \mathbb{P}(E),$$

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, U, q') *of* $\mathfrak{ts}$*, 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 $q' \in r$.*

◇

In a region $\mathfrak{r} = (in, r, out)$, the set *in* comprises events responsible for entering the set of states r, and *out* comprises events 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, $\mathfrak{r}$, cannot be identified only by its set of states r; in other words, *in* and *out* may not be recoverable from r. For example, the non-trivial regions of ST-system in Figure 2.1(a) are: $\mathfrak{r}_1 = (\varnothing, \{q_0\}, \{e\})$, $\mathfrak{r}_2 = (\{e\}, \{q\}, \varnothing)$, $\mathfrak{r}_3 = (\varnothing, \{q_0\}, \{f\})$, $\mathfrak{r}_4 = (\{f\}, \{q\}, \varnothing)$. Note that $\mathfrak{r}_1$ and $\mathfrak{r}_3$ are based on the same set of states and differ only by their *out* sets, and $\mathfrak{r}_2$ and $\mathfrak{r}_4$ are based on the same set of states and differ only by their *in* sets.

However, if $\mathfrak{ts}$ is *thin*, then different regions are based on different sets of states. For example, the non-trivial regions of the thin ST-system in Figure 2.2(a) are: $\mathfrak{r}_1 = (\varnothing, \{q_0, q_2\}, \{e\})$, $\mathfrak{r}_2 = (\{e\}, \{q_1, q_3\}, \varnothing)$, $\mathfrak{r}_3 = (\varnothing, \{q_0, q_1\}, \{f\})$, $\mathfrak{r}_4 = (\{f\}, \{q_2, q_3\}, \varnothing)$.

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,

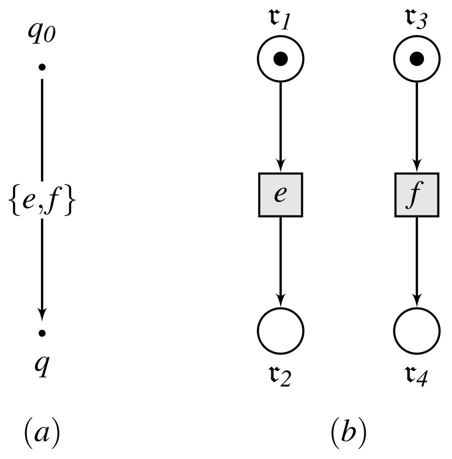


Fig. 2.1 An ENLST-system, where events e and f are co-located (a); and the ENL-system resulting from its synthesis (b).

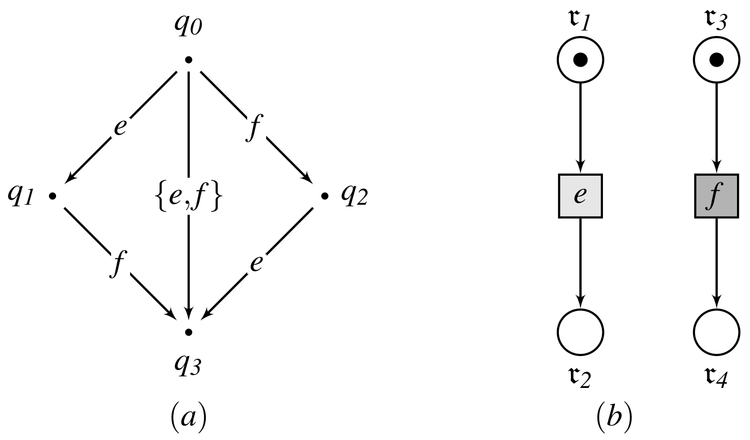


Fig. 2.2 An ENLST-system, where events e and f are not co-located (a); and the ENL-system resulting from its synthesis (b).

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

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 $e \in in$,

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

This extends in the usual way to sets of events, e.g., $^\circ U \overset{\text{df}}{=} \bigcup\{^\circ e \mid e \in U\}$.
For example, among the non-trivial regions of the ST-system in Figure 3.6 we have the

following two regions: $\mathfrak{r} = (\{c_1\}, \{q_6, q_7, q_8, q_9\}, \{c_3\})$ and $\mathfrak{r}' = (\{c_1\}, \{q_6, q_7, q_8, q_9\}, \{c_2\})$. We observe that $\mathfrak{r}, \mathfrak{r}' \in c_1{}^{\circ}$, $\mathfrak{r} \in {}^{\circ}c_3$ and $\mathfrak{r}' \in {}^{\circ}c_2$. Note that these two regions are based on the same set of states and differ only by their *out* sets. The need for having both $\mathfrak{r}$ and $\mathfrak{r}'$ is the consequence of the fact that all the transitions outgoing from the set of states $\{q_6, q_7, q_8, q_9\}$ are labelled by steps containing both c_2 and c_3 (see Definition 2.1.6(**R1**)).
The set of *potential steps of* $\mathfrak{ts}$ comprises all nonempty 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* $q \in Q$ if ${}^{\circ}U \subseteq \mathfrak{R}_q$ and $U^{\circ} \cap \mathfrak{R}_q = \varnothing$. We denote this by $U \in regenabled(q)$.

Definition 2.1.7. *An* ST*-system* $\mathfrak{ts} = (Q, A, q_0)$ *is an* ENL step transition system *w.r.t. a co-location relation* $\simeq$ *(or* ENLST$_{\simeq}$*-system) if the following hold:*

A1 *Each state is reachable.*

A2 *For every event e, both* ${}^{\circ}e$ *and* e° *are nonempty.*

A3 *For all distinct states q and* q', $\mathfrak{R}_q \neq \mathfrak{R}_{q'}$.

A4 *For every state q and step U,* $U \in allSteps_q$ iff $U \in regenabled(q)$ *and there is no event* $e \notin U$ *such that* $e \simeq U$ *and* $U \cup \{e\} \in regenabled(q)$.

We will also say that $\mathfrak{ts}$ *is an* ENLST*-system (if mentioning* $\simeq$ *is not important).* ♢

The **A1** axiom implies that all the states in ST-system are reachable from the initial state. **A2** will ensure that every event in a synthesised ENL-system will have at least one input condition and at least one output condition. **A3** was used for other transition systems as well, and is usually called the state separation property [9, 34], and it guarantees that $\mathfrak{ts}$ is deterministic. **A4** is a variation of the forward closure property [34] or the event/state separation property [9], and ensures that every step in a transition system is indeed a maximal step w.r.t. localities of the events it comprises (see [25]).

One can show (see [25]) that the ST-system generated by an ENL$_{\simeq}$-system is an ENLST$_{\simeq}$-system.

2.1.3 Synthesis of ENL-systems

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

Definition 2.1.8. *The tuple* associated *with an* ENLST$_{\simeq}$*-system* $\mathfrak{ts}$ *is given as*

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

where q_0 is the initial state of $\mathfrak{ts}$ *and* $F_{\mathfrak{ts}} = \{(\mathfrak{r}, e) \in \mathfrak{R}_{\mathfrak{ts}} \times E \mid \mathfrak{r} \in {}^{\circ}e\} \cup \{(e, \mathfrak{r}) \in E \times \mathfrak{R}_{\mathfrak{ts}} \mid \mathfrak{r} \in e^{\circ}\}$. $\diamondsuit$

The above construction always produces an ENL$_{\hat{=}}$-system which generates an ST-system isomorphic to $\mathfrak{ts}$ [25].

Theorem 2.1.9. *Let* $\mathfrak{ts}$ *be an* ENLST$_{\hat{=}}$*-system. Then* $\mathfrak{enl}^{\hat{=}}_{\mathfrak{ts}}$ *is an* ENL$_{\hat{=}}$*-system such that* $\mathfrak{ts} \cong \mathfrak{ts}_{\mathfrak{enl}^{\hat{=}}_{\mathfrak{ts}}}$. *Moreover, the unique isomorphism* ψ *between* $\mathfrak{ts}$ *and* $\mathfrak{ts}_{\mathfrak{enl}^{\hat{=}}_{\mathfrak{ts}}}$ *is given by* $\psi(q) = \mathfrak{R}_q$, *for every state* q *of* $\mathfrak{ts}$.

The ENL-system $\mathfrak{enl}^{\hat{=}}_{\mathfrak{ts}}$ may contain conditions which are *redundant* from the point of view of its behaviour, i.e., deleting such conditions would result in an ENL-system generating the same ST-system [28].

2.2 Related works

In Chapter 1 we introduced our research topic in the context of the existing literature of the area. In this section, we focus on the selected papers that were the most important for this thesis.

Paper [25] was a source of the initial definitions and the starting point for our research. It introduced and investigated ENL-system with localities (ENL-systems), and provided a solution to the corresponding synthesis problem (from step transition systems to ENL-systems), by adapting the classical theory of regions [9] to cope with local maximal concurrency.

Sources [16, 17], where theoretical foundations for the Petrify tool were described, were important when we were developing our algorithms for extracting regions of ST-systems. In particular, we found the ideas of excitation and switching regions, which were introduced there for the standard transition systems, to be very useful and we generalised these ideas to our setting of ST-systems.

Paper [27], provided theoretical foundations for the implementation of the algorithms for the synthesis of ENL/LC-systems, where instead of assuming that localities are given at the outset, they are discovered as part of the synthesis procedure.

Paper [28] was a starting point for developing algorithms for the minimisation of the synthesised nets. We implemented three reduction rules introduced there, which allowed us to design experiments to find out the most suitable strategy to eliminate redundant regions (i.e., the most suitable according to our set of criteria). When working on the minimisation of nets, we re-defined certain notions of [28]: in particular, the notion of minimal regions.

Paper [13] was an inspiration for us when reasoning about state machine decomposability of the synthesised nets. Our new definition of minimal regions allowed us to prove new results regarding state machine decomposability of the synthesised ENL-systems, similar to the results of [13] obtained for the Elementary Net Systems.

Papers [39, 38, 41] were the initial source of information about the WORKCRAFT tool, which we decided to use as a framework for our algorithms for the synthesis of ENL-systems.

Chapter 3

Finding non-trivial regions

In this chapter, we discuss how one might generate non-trivial regions of an ST-system in an effective way. As regions are triples $\mathfrak{r} = (in, r, out) \in \mathbb{P}(E) \times \mathbb{P}(Q) \times \mathbb{P}(E)$ satisfying Definition 2.1.6(**R1**-**R4**), a brute force approach is too costly even for small ST-systems like the ENLST-system in Figure 3.1 (see also its associated ENL-system in Figure 3.2) [4]. We therefore focus on improving the computational effort by reducing the number of the transitions considered in Definition 2.1.6 as well as the in/out sets and the r sets. Also, we are interested in comparing the improvements resulting from these three ways aimed at efficiency gains. We consider the execution time of the algorithms as the main measure to gauge their efficiency.

The initial idea was to focus on thin ST-systems, as in these transition systems, for every event $e \in E$, there is a transition $(q, \{e\}, q')$, for some $q, q' \in Q$. This, we believed, would allow us to ignore the thick transitions and base our algorithm solely on singleton transitions, which in turn would enable us to use some of the techniques used in the Petrify tool [17], which works with standard transition systems rather than with ST-systems. Although we discovered that in certain circumstances the thick transitions can be ignored, leading to some reduction of execution time when verifying regional conditions **R1**-**R4**, the rules for ignoring thick transitions were complicated, and the class of thin ST-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 ST-system, in our algorithms. Therefore, we abandoned the special case of thin ST-systems and decided to seek a solution for the general class of ST-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 ENLST-systems as defined in [25] and recalled in Section 2.1. 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 ST-systems. For example, the algorithm took around 37 minutes to generate the non-trivial regions for the ENLST-system in Figure 3.1 (see [4]). This was not surprising. As region is a triple $\mathfrak{r} = (in, r, out) \in \mathbb{P}(E) \times \mathbb{P}(Q) \times \mathbb{P}(E)$, the computation of all non-trivial regions in such a way would involve generation of all subsets of the set of states Q, all subsets of the set of events E and then checking regional conditions, **R1**-**R4**, for all the transitions in the ST-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 ST-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 target sets. This idea was first used in the Petrify tool [17] and is adapted here to the context of ST-systems. This approach is explained in Section 3.4.

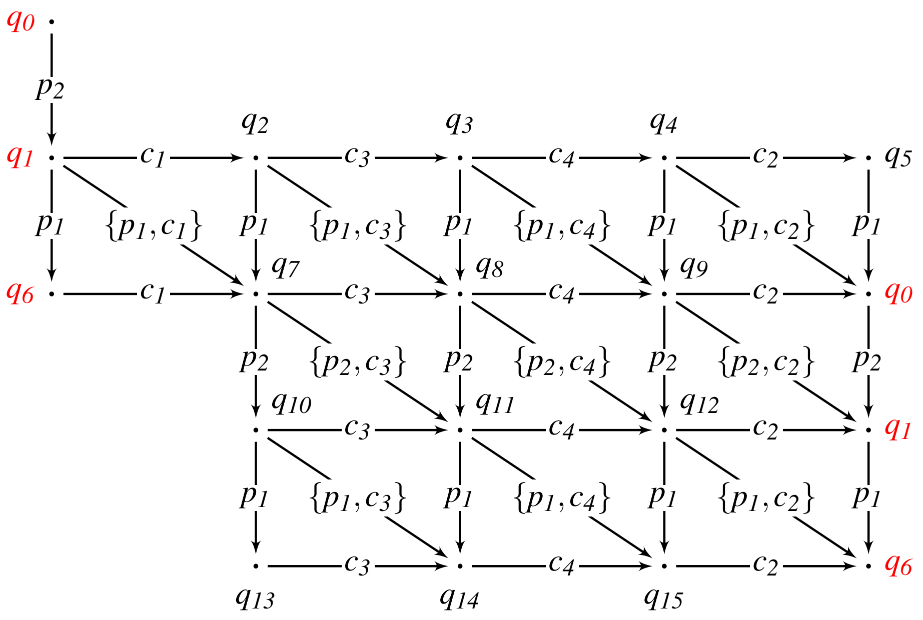


Fig. 3.1 An ENLST-system $\mathfrak{ts}_3$, where p_1 and p_2 are co-located events, and c_1, c_2, c_3 and c_4 are co-located events. The graph should be glued on the states q_0, q_1, q_6 that appear twice in the picture.

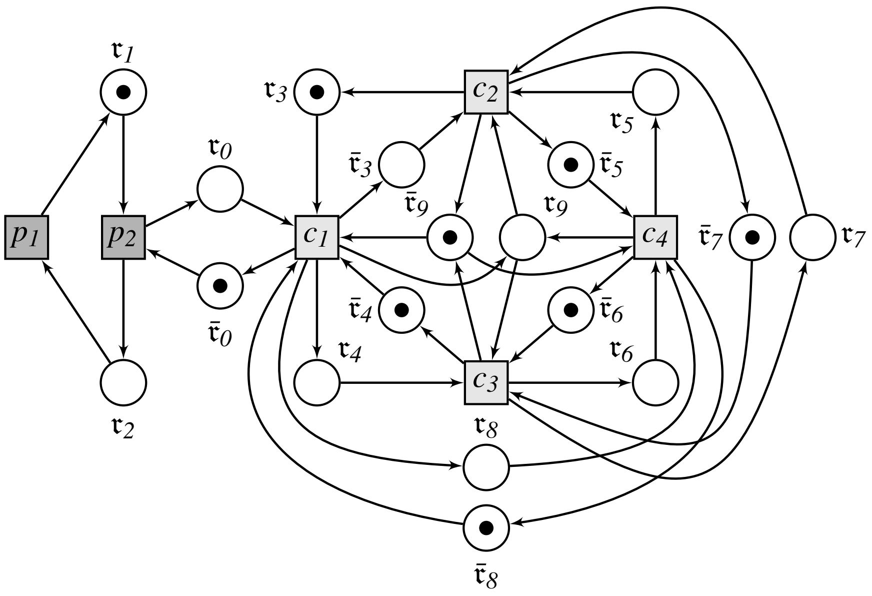


Fig. 3.2 The ENL-system resulting from the synthesis of the ENLST-system $\mathfrak{ts}_3$ in Figure 3.1. Note that the synthesised net contains many redundant conditions (see Section 6.1, p. 65, for definition).

3.2 Ignoring thick transitions

In this approach, we attempted to improve the execution time of extracting all the non-trivial regions by ignoring some of 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 2.1.2).

A key problem here is which of the thick transitions can be ignored without changing the set of non-trivial regions. Thin ST-systems are the first candidate class of ST-systems to consider, as they contain transitions labelled with all possible singleton steps, so all the events are present in the steps labelling the non-thick transitions. However, even in some of the thin ST-systems removing thick transitions can be problematic as the example in Figure 3.3 shows. In that example removing the only thick transition, $(q_2, \{e, f\}, q_3)$, leads to a disconnected ST-system which is not an ENLST-system as Definition 2.1.7(**A1**) no longer holds. Thus the property of being a thin ST-system is too weak to make a decision about the removal of thick transitions. We need to make stronger assumptions, e.g., as those stated in the next proposition.

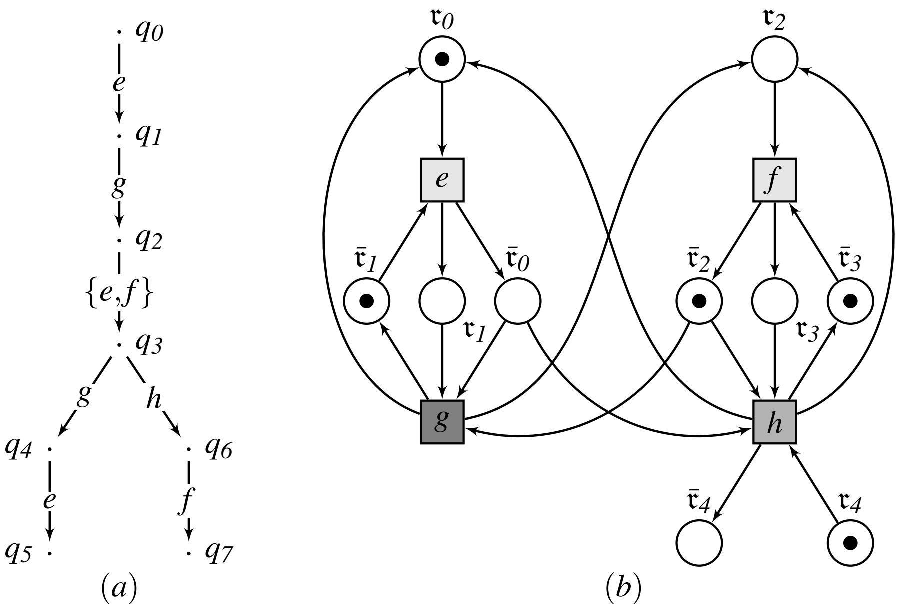


Fig. 3.3 An ENLST-system $\mathfrak{ts}_2$ with co-located events e and f and non co-located events g and h (a); the ENL-system resulting from its synthesis (b).

Proposition 3.2.1. *Let $\mathfrak{t} = (q,U,q')$ be a thick transition of an* ST*-system $\mathfrak{ts}$, and $U_1,\dots,U_k$ ($k \geq 2$) be a partition of U such that $U_{i_1}\dots U_{i_k}$ is a step sequence from q to q', for every permutation $i_1,\dots,i_k$ of $1,\dots,k$. Then $\mathfrak{R}_{\mathfrak{ts}} = \mathfrak{R}_{\mathfrak{ts}'}$, where $\mathfrak{ts}'$ is the* ST*-system obtained from $\mathfrak{ts}$ by deleting $\mathfrak{t}$.*

Proof: We first observe that the assumptions imply the existence of a path $(q_1,U_1,q_2)(q_2,U_2,q_3)\dots(q_k,U_k,q_{k+1})$ in $\mathfrak{ts}'$ with $q = q_1$ and $q' = q_{k+1}$.
The inclusion $\mathfrak{R}_{\mathfrak{ts}} \subseteq \mathfrak{R}_{\mathfrak{ts}'}$ holds trivially (see Definition 2.1.6). To prove $\mathfrak{R}_{\mathfrak{ts}'} \subseteq \mathfrak{R}_{\mathfrak{ts}}$, let $\mathfrak{r} = (in,r,out) \in \mathfrak{R}_{\mathfrak{ts}'}$. To show $\mathfrak{r} \in \mathfrak{R}_{\mathfrak{ts}}$ it suffices to demonstrate that Definition 2.1.6(**R1**–**R4**) hold for $\mathfrak{t}$ and $\mathfrak{r}$ in $\mathfrak{ts}$. Assume to the contrary that one of the conditions does not hold for $\mathfrak{t}$ and $\mathfrak{r}$.

Suppose first that Definition 2.1.6(**R1**) does not hold (if Definition 2.1.6(**R2**) does not hold, we proceed in a similar way). Then $q \in r$ and $q' \notin r$ and $|U \cap out| \neq 1$. As $\mathfrak{r} \in \mathfrak{R}_{\mathfrak{ts}'}$ and $q \in r$ and $q' \notin r$, there is 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 Definition 2.1.6(**R1**) is satisfied in $\mathfrak{ts}'$. Hence $|U_i \cap out| = 1$, and so, as $|U \cap out| \neq 1$, there exists $j \neq i$ such that $|U_j \cap out| \geq 1$. From the assumptions it follows that there is a path $(q'_1,U'_1,q'_2)(q'_2,U'_2,q'_3)\dots(q'_k,U'_k,q'_{k+1})$ from $q = q'_1$ and $q' = q'_{k+1}$ in $\mathfrak{ts}'$ such that $U'_1 = U_i$ and $U'_2 = U_j$. Since $\mathfrak{r}$ is a region of $\mathfrak{ts}'$ and $U_j \cap out \neq \varnothing \neq U_i \cap out$, it follows from Definition 2.1.6(**R3**) that $q'_2 \notin r$ and $q'_2 \in r$, yielding a contradiction.
Suppose now that Definition 2.1.6(**R3**) does not hold (if Definition 2.1.6(**R4**) does not

hold, we proceed in a similar way). Then $U \cap out \neq \varnothing$ and $q \notin r \vee q' \in r$. We can assume without the loss of generality that $q \notin r$. Moreover, let i be such that $U_i \cap out \neq \varnothing$. From the assumptions it follows that there is a path $(q'_1, U'_1, q'_2)(q'_2, U'_2, q'_3) \dots (q'_k, U'_k, q'_{k+1})$ from $q = q'_1$ to $q' = q'_{k+1}$ in $\mathfrak{ts}'$ such that $U'_1 = U_i$. Since $\mathfrak{r}$ is a region of $\mathfrak{ts}'$ and $U_i \cap out \neq \varnothing$, it follows from Definition 2.1.6(**R3**) that $q'_1 \in r$, yielding a contradiction with $q = q'_1$ and $q \notin r$. □

Checking the conditions stated in Proposition 3.2.1 is computationally expensive. However, there is a class of ST-systems, for which they are always satisfied as shown in Theorem 3.2.3. But first, we introduce a concept of *co-located sequential events* in the context of step transition systems.

Definition 3.2.2. *Let* $\mathfrak{ts}$ *be an* ST*-system, and* $\simeq$ *be a co-location relation with the equivalence classes* $E_1, \dots, E_m$. *Then* $\simeq$ *partitions the set of events into sets of* co-located sequential events *if, for all states* q *of* $\mathfrak{ts}$, *steps* $U \in allSteps_q$, *and* $1 \leq i \leq m$, $|U \cap E_i| \leq 1$. ◇

Theorem 3.2.3. *Let* $\mathfrak{ts}$ *be an* ENLST$_\simeq$*-system such that* $\simeq$ *partitions the set of events into sets of co-located sequential events. Then all thick transitions in* $\mathfrak{ts}$ *satisfy the conditions of Proposition 3.2.1 with each step* U_i *being a singleton.*

Proof: From Definition 3.2.2 it follows that the ENL$_\simeq$-system $\mathfrak{enl}_{\mathfrak{ts}}^{\simeq}$ can be treated as an elementary net system for which the step monotonicity property (see Fact 1) is satisfied for every (resource) enabled step. Hence, for every transition $(q, \{e_1, \dots, e_k\}, q')$, there is a step sequence $\{e_1\} \dots \{e_k\}$ from q to q' in $\mathfrak{ts}$. As a result, for every thick transition of $\mathfrak{ts}$, the conditions of Proposition 3.2.1 are satisfied with each step U_i being a singleton. □

Note that if a co-location relation $\simeq$ partitions the set of events into sets of *co-located sequential events*, then an ENLST$_\simeq$-system $\mathfrak{ts}$ satisfies the substep property (see Definition 2.1.7(**A4**)).

The last result provides sufficient conditions for removing all the thick transitions from an ENLST$_\simeq$-system $\mathfrak{ts}$ when computing its non-trivial regions. Its practical application in a synthesis procedure for an arbitrary ST-system is discussed in Chapter 4.

The next result shows that in the case of ENLST-systems the test for removing transitions can be simplified.

Proposition 3.2.4. *Let* $\mathfrak{t} = (q, U, q')$ *be a thick transition of an* ENLST*-system* $\mathfrak{ts}$ *which satisfies the conditions in Proposition 3.2.1. Then there are two paths from* q *to* q', $(q, U', s)(s, U'', q')$ *and* $(q, U'', t)(t, U', q')$, *such that* $U = U' \uplus U''$.

Proof: Follows from Fact 2. □

Thus, if one is interested, e.g., in re-synthesising ENL-systems (cf. [7]), then one only needs to apply Proposition 3.2.1 for $k = 2$. Moreover, if $\mathfrak{ts}$ is an arbitrary ST-system such that Proposition 3.2.1 can be applied for some $k \geq 3$ but not for $k = 2$, then $\mathfrak{ts}$ is not an ENLST-system. This provides an additional test for being an ENLST-system.

3.3 Reducing the number of *in*/*out* sets

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 structure and labelling of ST-systems.
Finding non-trivial regions of ST-systems involves 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 now look at the possibilities of minimising the number of the *in*/*out* sets to be considered. The following results present cases in which one can do this safely.

Proposition 3.3.1. *Let $\mathfrak{r} = (in, r, out)$ be a region of an* ST*-system $\mathfrak{ts} = (Q, A, q_0)$. If the target of any transition (q, U', q') in $\mathfrak{ts}$, where $e \in U'$, is the source of another transition (q', 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, U', q') and (q', 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 3.3.2. *Let $\mathfrak{r} = (in, r, out)$ be a region of an* ST*-system $\mathfrak{ts} = (Q, A, q_0)$. If the target of any transition $(q, \{e\}, q')$ in $\mathfrak{ts}$ is the source of another transition $(q', \{f\}, q'')$ in $\mathfrak{ts}$, for $f \neq e$, then $\{e, f\} \not\subseteq in$ and $\{e, f\} \not\subseteq out$.*

Proof. Follows directly from Proposition 3.3.1. □

Proposition 3.3.3. *Let $\mathfrak{r} = (in, r, out)$ be a region of an* ST*-system $\mathfrak{ts} = (Q, A, q_0)$, and let (q, U, q') be a transition in $\mathfrak{ts}$, such that there are two events $e, f \in U$. Then $\{e, f\} \not\subseteq in \cup out$, which can be split into three sub-cases:*

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

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

3. $(e \in in \Rightarrow f \notin out)$ *and* $(f \in in \Rightarrow e \notin out)$*.*

Proof. 1. Assume, to the contrary, that we have a transition (q,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,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,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. We prove that $e \in in$ implies $f \notin out$ (the second implication has just e and f swapped). Suppose, to the contrary, that $e \in in$ and $f \in out$. Since (q,U,q') is labelled by a step U, where $e,f \in U$, and $\mathfrak{r} = (in,r,out)$ is a region, we have that regional condition **R3** is satisfied for (q,U,q') and f and regional condition **R4** is satisfied for (q,U,q') and e, leading to a contradiction. So, the implication $e \in in \Rightarrow f \notin out$ holds.

□

3.4 Reductions based on 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 transitions are labelled by single events, these sets would solely define regions. So, although we are considering step transition systems here 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 [16, 17, 23], 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 ST-systems. Also, we take advantage of the fact that ST-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 target sets in the context of ST-systems.

Definition 3.4.1. *Let* $\mathfrak{ts} = (Q, A, q_0)$ *be an* ST*-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 $T_e = \{q' \in Q \mid \exists q \in Q : (q, U, q') \in A \ \wedge \ e \in U\}$.

$\diamondsuit$

Proposition 3.4.2. *Let* $\mathfrak{r} = (in, r, out)$ *be a region of an* ST*-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 $e \in U \cap in$ and from the definition of a region (**R4**) we have: $q \notin r$ and $q' \in r$. So, $T_e \subseteq r$ and $S_e \cap r = \varnothing$.

$\square$

If we want to combine the discovery of sets *in*, *out* and *r*, when searching for all non-trivial regions of ST-systems, we can use the results given in Corollary 3.4.3, Proposition 3.4.4 and Corollary 3.4.5 below.

Corollary 3.4.3. *Let* $\mathfrak{r} = (in, r, out)$ *be a region of an* ST*-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 3.4.2. $\square$

From Corollary 3.4.3, we see that the set of states r of any region $\mathfrak{r} = (in, r, out)$ in an ST-system $\mathfrak{ts} = (Q, A, q_0)$ can be represented as

$$r = \bigcup_{e \in out} S_e \cup \bigcup_{e \in in} T_e \cup \Phi^{\mathfrak{r}}\,, \tag{3.1}$$

where $\Phi^{\mathfrak{r}}$ 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 3.4.4. *Let* $\mathfrak{r} = (in, r, out)$ *be a region of an* ST*-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 3.4.5. *Let* $\mathfrak{r} = (in, r, out)$ *be a region of an* ST*-system on* E $\mathfrak{ts} = (Q, A, q_0)$ *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* $q' \in \Phi^{\mathfrak{r}}$.
2. *If* $q' \in S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$ *and* $q \notin S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$ *then* $q \in \Phi^{\mathfrak{r}}$.

Proof. Follows from the representation of r (see Eq. (3.1)) and Proposition 3.4.4. □

3.5 Extracting regions

Before we propose an algorithm for computing non-trivial regions of an ST-system $\mathfrak{ts}$, we summarise a few useful facts:

1. For every region $\mathfrak{r} = (in, r, out)$ of an ST-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 $\mathfrak{r} = (in, r, out)$ of an ST-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.

Algorithm 1 takes as input any ST-system as defined in Definition 2.1.1, and computes regions according to Definition 2.1.6. 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 2.1.7) to decide whether the input ST-system is an ENLST-system and therefore synthesisable. To explain the algorithm we can look at the ST-system of Figure 3.3(a). In this case, as already observed, the only thick transition cannot be removed. The algorithm then obtains all the potential in/out sets using the results from Section 3.3. Let us consider one (in, out) pair with $in = \varnothing$ and $out = \{h\}$, which 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 will satisfy $r = S^{\mathfrak{r}} \cup T^{\mathfrak{r}} \cup \Phi^{\mathfrak{r}} = \bigcup_{e \in out} S_e \,\cup\, \bigcup_{e \in in} T_e \cup \Phi^{\mathfrak{r}}$ (see Eq.(3.1)). First, the algorithm computes $S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$, in this case containing a single state q_3. The most involved part of the algorithm is the one to compute the (implicit) filler set $\Phi^{\mathfrak{r}}$ (using Corollary 3.4.5). The set $\Phi^{\mathfrak{r}}$ is initially empty. So, initially, $r = S = S^{\mathfrak{r}} \cup T^{\mathfrak{r}} = \{q_3\}$. The algorithm starts the while-loop at line 19 (see lines 19-27). Below we go through its iterations.

1. $S = \{q_3\}$ and $\Phi_{current} = \varnothing$ (this set records the states added to r during the current iteration of the loop). We consider all possible transitions adjacent to q_3 that are not labelled by steps containing events from $in \cup out = \{h\}$. We have two transitions to consider:

    - Transition $(q_3, \{g\}, q_4)$: we add q_4 to r to 'bury' this transition in r. Now we have: $\Phi_{current} = \{q_4\}$ and $r = \{q_3, q_4\}$.
    - Transition $(q_2, \{e, f\}, q_3)$: we add q_2 to r to 'bury' this transition in r. Now we have: $\Phi_{current} = \{q_4, q_2\}$ and $r = \{q_3, q_4, q_2\}$.

2. $S = \{q_2, q_4\}$ and $\Phi_{current} = \varnothing$ and we consider all possible transitions 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 transitions to consider:

    - Transition $(q_4, \{e\}, q_5)$: we add q_5 to r to 'bury' this transition in r. Now we have: $\Phi_{current} = \{q_5\}$ and $r = \{q_2, q_3, q_4, q_5\}$.
    - Transition $(q_1, \{g\}, q_2)$: we add q_1 to r to 'bury' this transition in r. Now we have: $\Phi_{current} = \{q_5, q_1\}$ and $r = \{q_3, q_4, q_5, q_2, q_1\}$.

3. $S = \{q_5, q_1\}$ and $\Phi_{current} = \varnothing$ and we consider all possible transitions 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 transition to consider:

    - Transition $(q_0, \{e\}, q_1)$: we add q_0 to r to 'bury' this transition in r. Now we have: $\Phi_{current} = \{q_0\}$ and $r = \{q_3, q_4, q_5, q_2, q_1, q_0\}$.

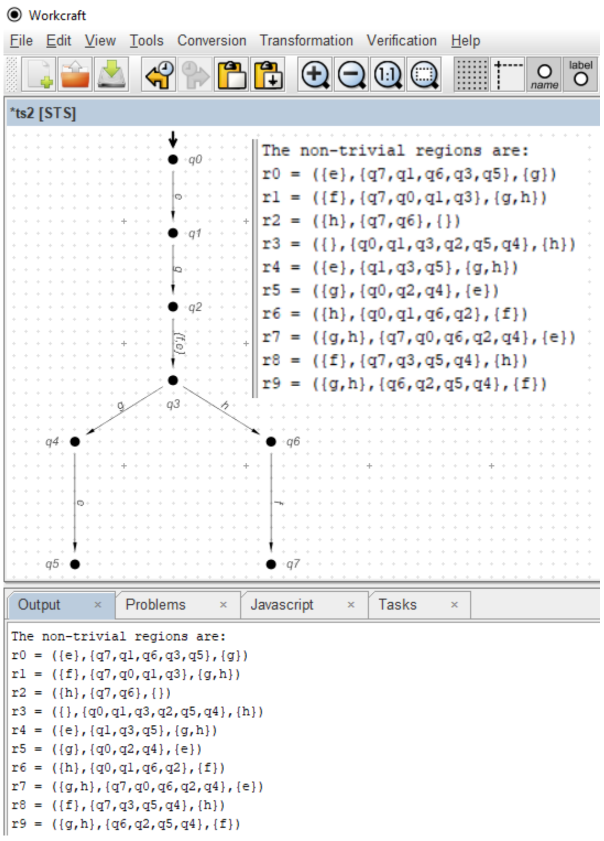
Algorithm 1 Extracting $\mathfrak{R}_{\mathfrak{ts}}$ for an ST-system $\mathfrak{ts} = (Q, A, q_0)$.

```
1: function EXTRACT_REGIONS($\mathfrak{ts}$)
2:     remove selected thick transitions from $A$                    ▷ see Section 3.2
3:     initialise $\mathfrak{R}_{\mathfrak{ts}}$ to $\varnothing$
4:     initialise $InOut$ to all potential $in/out$ sets             ▷ see Section 3.3
5:     initialise $InOutPairs$ to $InOut \times InOut$
6:     for all $in, out \in InOut$ do
7:         remove either $(out, in)$ or $(in, out)$ from $InOutPairs$
8:     for every $(in, out) \in InOutPairs$ do                       ▷ see Section 3.4
9:         initialise $S^{\mathfrak{r}}$ and $T^{\mathfrak{r}}$ to $\varnothing$
10:        if $in \cap out = \varnothing$ and $in \cup out \neq \varnothing$ then
11:            for every $e \in in$ do
12:                calculate $T_e$ and add to $T^{\mathfrak{r}}$
13:            for every $e \in out$ do
14:                calculate $S_e$ and add to $S^{\mathfrak{r}}$
15:        else
16:            break                                                 ▷ $(in, out)$ is invalid
17:        initialise $S$ to $S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$    ▷ initial states to consider
18:        initialise $r$ to $S^{\mathfrak{r}} \cup T^{\mathfrak{r}}$
19:        while $S \neq \varnothing$ do
20:            initialise $\Phi_{current}$ to $\varnothing$
21:            for every $(q, U, q') \in A$ do
22:                if $(in \cup out) \cap U = \varnothing$ then
23:                    if $q \in S$ and $q' \notin r$ then
24:                        add $q'$ to $r$ and $\Phi_{current}$
25:                    if $q' \in S$ and $q \notin r$ then
26:                        add $q$ to $r$ and $\Phi_{current}$
27:            set $S$ to $\Phi_{current}$
28:        if $\mathfrak{r} = (in, r, out)$ satisfies regional axioms then
29:            add $\mathfrak{r}$ and $\bar{\mathfrak{r}} = (out, Q \setminus r, in)$ to $\mathfrak{R}_{\mathfrak{ts}}$
30:    return $\mathfrak{R}_{\mathfrak{ts}}$
```

4. $S = \{q_0\}$ and $\Phi_{current} = \varnothing$. In this iteration we cannot add any more states to r that are not already there, so S is set to $\varnothing$ making it the last 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 $\bar{\mathfrak{r}} = (\{h\}, \{q_6, q_7\}, \varnothing)$) are added to the set of discovered regions. These are two out of ten non-trivial regions of $\mathfrak{ts}_2$ in Figure 3.3(a) discovered by

Algorithm 1. See also the screenshot from WORKCRAFT, for this example, in Figure 3.4. Note that each time the algorithm is run the naming of the generated regions might be different. So, for example, $\mathfrak{r}_0$ in Figure 3.4 is $\mathfrak{r}_1$ in Figure 3.3(b).



```
The non-trivial regions are:
r0 = ({e},{q7,q1,q6,q3,q5},{g})
r1 = ({f},{q7,q0,q1,q3},{g,h})
r2 = ({h},{q7,q6},{})
r3 = ({},{q0,q1,q3,q2,q5,q4},{h})
r4 = ({e},{q1,q3,q5},{g,h})
r5 = ({g},{q0,q2,q4},{e})
r6 = ({h},{q0,q1,q6,q2},{f})
r7 = ({g,h},{q7,q0,q6,q2,q4},{e})
r8 = ({f},{q7,q3,q5,q4},{h})
r9 = ({g,h},{q6,q2,q5,q4},{f})
```

```
The non-trivial regions are:
r0 = ({e},{q7,q1,q6,q3,q5},{g})
r1 = ({f},{q7,q0,q1,q3},{g,h})
r2 = ({h},{q7,q6},{})
r3 = ({},{q0,q1,q3,q2,q5,q4},{h})
r4 = ({e},{q1,q3,q5},{g,h})
r5 = ({g},{q0,q2,q4},{e})
r6 = ({h},{q0,q1,q6,q2},{f})
r7 = ({g,h},{q7,q0,q6,q2,q4},{e})
r8 = ({f},{q7,q3,q5,q4},{h})
r9 = ({g,h},{q6,q2,q5,q4},{f})
```

Fig. 3.4 Extracting non-trivial regions of the ENLST-system $\mathfrak{ts}_2$ in Figure 3.3(a) in WORKCRAFT.

3.6 Results of the experiments

To test Algorithm 1, we selected the following examples: $\mathfrak{ts}_1$ in Figure 3.5 (2 states, 4 events), $\mathfrak{ts}_2$ in Figure 3.3 (8 states, 4 events), $\mathfrak{ts}_3$ in Figure 3.1 (16 states, 6 events), $\mathfrak{ts}_4$ in Figure 3.6 (12 states, 6 events), and the ST-system generated by the ENL-system in Figure 3.7 ($\mathfrak{ts}_5$). The last ST-system has 64 states and 9 events. The selected test cases consider ST-systems representing nets with different characteristics: thin ST-systems ($\mathfrak{ts}_2$, $\mathfrak{ts}_3$, $\mathfrak{ts}_5$) or not thin ($\mathfrak{ts}_1$, $\mathfrak{ts}_4$); ST-systems generated by nets with conflicts ($\mathfrak{ts}_1$, $\mathfrak{ts}_2$) or without conflicts ($\mathfrak{ts}_3$, $\mathfrak{ts}_4$, $\mathfrak{ts}_5$); ST-systems generated by nets, where every locality represents a sequential subsystem ($\mathfrak{ts}_3$, $\mathfrak{ts}_5$) or not ($\mathfrak{ts}_1$, $\mathfrak{ts}_2$, $\mathfrak{ts}_4$).

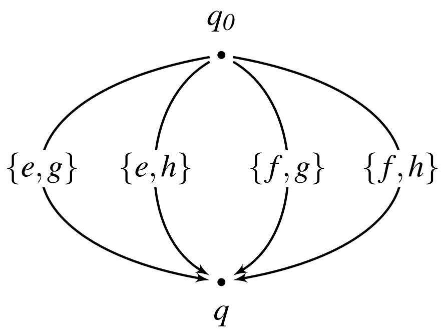


Fig. 3.5 An ENLST-system $\mathfrak{ts}_1$, where e, f, g and h are co-located events.

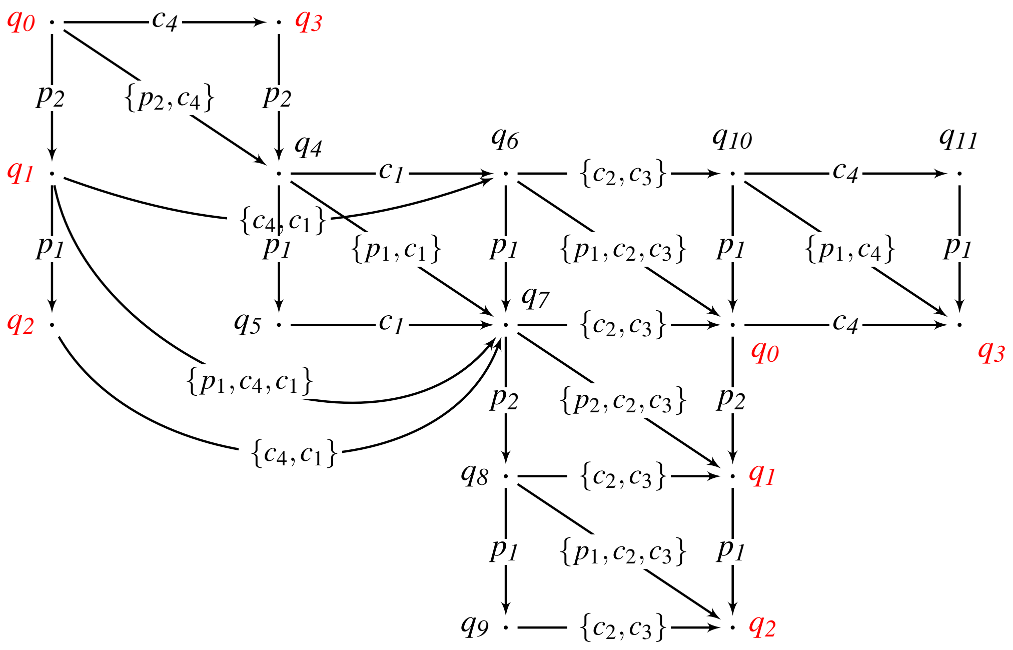


Fig. 3.6 An ENLST-system $\mathfrak{ts}_4$, where p_1 and p_2 are co-located events, and c_1, c_2, c_3 and c_4 are co-located events. The graph should be glued on the states q_0, q_1, q_2, q_3 that appear twice in the picture. $\mathfrak{ts}_4$ is isomorphic to the ENLST-system generated by the ENL-system in Figure 1.1 on p. 4.

When testing Algorithm 1, the aim was to check whether it generates correctly all the expected regions. The experiments confirmed that Algorithm 1 computed correctly all the expected regions for these small examples.

Secondly, we were interested in comparing the improvements to Algorithm 1 resulting from the three efficiency measures described in Sections 3.2, 3.3, and 3.4. The results are summarised in Table 3.1, where:

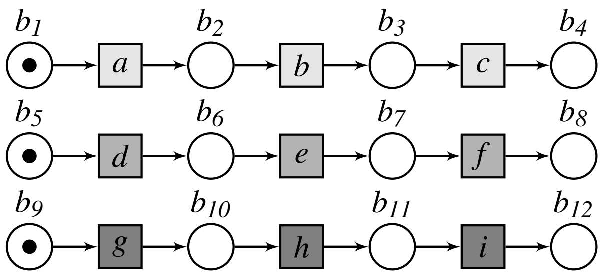


Fig. 3.7 An ENL-system, where events a, b, c are sharing a locality; events d, e, f are in the second locality; and events g, h and i are in the third locality. It generates ENLST-system $\mathfrak{ts}_5$ considered in Table 3.1 which is also considered in Table 3.2 and Table 5.1 as $\mathfrak{ts}_{3,3}$.

- A1(3.2) means that we applied only the measures described in Section 3.2, i.e., the algorithm removed some selected thick transitions from ST-systems according to Propositions 3.2.1 and 3.2.4. All potential *in*, *r* and *out* sets were considered in the computation.
- A1(3.3) means that the algorithm used only pre-selected *in*/*out* sets as described in Section 3.3, but considered all the transitions of ST-systems and all possible candidates for the *r* sets.
- A1(3.3,3.4) means that the algorithm used only pre-selected *in*/*out* sets (see Section 3.3), and only selected *r* sets (see Section 3.4), but considered all the transitions of ST-systems.
- A1 means that the algorithm employed all the efficiency measures as described in Section 3.2 (Propositions 3.2.1 and 3.2.4), Section 3.3 and Section 3.4 (see Algorithm 1).

Table 3.1 clearly shows that the best savings are gained by reducing the number of potential *in*/*out* sets, using the results of Section 3.3.

Table 3.1 Comparing execution times of Algorithm 1 when using different combinations of efficiency measures.

$\mathfrak{ts}$	**|Q|**	**|E|**	**A1(3.2)**	**A1(3.3)**	**A1(3.3,3.4)**	**A1**
$\mathfrak{ts}_1$	2	4	14.8 ms	11.7 ms	3.6 ms	3.55 ms
$\mathfrak{ts}_2$	8	4	175.1 ms	24.1 ms	6.5 ms	6.55 ms
$\mathfrak{ts}_3$	16	6	2123926.5 ms	208.5 ms	24.3 ms	22.30 ms
$\mathfrak{ts}_4$	12	6	99278.1 ms	206.2 ms	14.2 ms	12.80 ms
$\mathfrak{ts}_5$	64	9	memory overflow	memory overflow	263.8 ms	195.40 ms

To test how the performance of Algorithm 1 scales with the increasing sizes of inputs, we used ST-systems generated by nets composed of several sequential subsystems like the ST-system $\mathfrak{ts}_{3,3}$ generated by the ENL-system in Figure 3.7. In the notation $\mathfrak{ts}_{i,j}$, the index i denotes the number of sequential subsystems, and the index j denotes the number of events in each of the line-like sequential subsystems. The experimental results are summarised in Table 3.2.

The last column of Table 3.2 contains the results for a variation of Algorithm 1, denoted as Algorithm 1^*.

Algorithm 1^* takes as its input not only an ST-system $\mathfrak{ts}$, but also a co-location relation $\simeq$. In its implementation it differs from Algorithm 1 only in the part concerning the removal of thick transitions (see line 2 of Algorithm 1 and the description of the approach A1(3.2) of Table 3.1 for Algorithm 1).

The additional information about the co-location relation is used in Algorithm 1^* to check whether it satisfies the conditions for partitioning the set of events into sets of co-located sequential events (see Definition 3.2.2). If so, the Algorithm 1^* can remove all the thick transitions (rather than some selected ones) following Theorem 3.2.3. As this is possible for all the tested $\mathfrak{ts}_{i,j}$ ST-systems we can see the savings achieved in this way by comparing the columns for Algorithm 1 and Algorithm 1^* in Table 3.2.

Table 3.2 Execution time taken to derive non-trivial regions of selected ST-systems when using Algorithm 1 and Algorithm 1^*.

$\mathfrak{ts}$	$\lvert\mathbf{Q}\rvert$	$\lvert\mathbf{E}\rvert$	Algorithm 1	Algorithm 1^*
$\mathfrak{ts}_{3,3}$	64	9	0.1954 s	0.1718 s
$\mathfrak{ts}_{3,4}$	125	12	2.0029 s	1.3520 s
$\mathfrak{ts}_{3,5}$	216	15	13.9049 s	9.6727 s
$\mathfrak{ts}_{4,3}$	256	12	10.2716 s	7.5632 s
$\mathfrak{ts}_{4,4}$	625	16	172.0079 s	118.2612 s
$\mathfrak{ts}_{4,5}$	1296	20	1720.4560 s	1099.3429 s

Chapter 4

Using non-trivial regions in the synthesis procedure

4.1 Two approaches to the synthesis procedure

The synthesis procedure based on Definition 2.1.8 and Theorem 2.1.9 produces $\mathfrak{enl}^{\hat{=}}_{\mathfrak{ts}}$, where all the non-trivial regions are used as conditions (called the saturated net). It assumes that the initial ST-system is an ENLST-system. However, Algorithm 1, which extracts non-trivial regions of an ST-system takes as an input any ST-system. This means that another approach to the synthesis procedure can be designed supported by the results presented in this section. We will refer to the original approach to the synthesis procedure as Method I and to the alternative procedure as Method II.

Proposition 4.1.1. *Let $\mathfrak{ts}$ and $\mathfrak{ts}'$ be two isomorphic* ST*-systems, and $\hat{=}$ be a co-location relation. Then $\mathfrak{ts}$ is an* ENLST$_{\hat{=}}$*-system* iff *$\mathfrak{ts}'$ is an* ENLST$_{\hat{=}}$*-system.*

Proof: Follows directly from Definition 2.1.7(**A1–A4**). □

Consider an arbitrary ST-system $\mathfrak{ts} = (Q, A, q_0)$ and a co-location relation $\hat{=}$. After computing all the non-trivial regions and constructing $(\mathfrak{R}_{\mathfrak{ts}}, E, F_{\mathfrak{ts}}, \hat{=}, \mathfrak{R}_{q_0})$, we still do not know whether it is a solution to the synthesis problem for $\mathfrak{ts}$ w.r.t. $\hat{=}$. However, we have the following result, generalising one that holds for the elementary net systems [7].

Proposition 4.1.2. *Let $\mathfrak{ts}$ be an* ST*-system, and $\mathfrak{enl}$ be an* ENL$_{\hat{=}}$*-system which is a net realisation of $\mathfrak{ts}$. Then $\mathfrak{ts}$ is an* ENLST$_{\hat{=}}$*-system.*

Proof: It was shown in [25] that $\mathfrak{ts}_{\mathfrak{enl}}$ is an ENLST$_{\hat{=}}$-system. Hence, by Proposition 4.1.1, $\mathfrak{ts}$ is also an ENLST$_{\hat{=}}$-system. □

Corollary 4.1.3. *If an* ST*-system* $\mathfrak{ts}$ *has a net realisation which is an* ENL$_{\widehat{=}}$*-system, then* $\mathfrak{ts}$ *has also the regional net realisation* $\mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}}$.

Proof: By Proposition 4.1.2, $\mathfrak{ts}$ is an ENLST$_{\widehat{=}}$-system. Hence, by Theorem 2.1.9, $\mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}}$ is an ENL$_{\widehat{=}}$-system and the ST-system generated by it is isomorphic to $\mathfrak{ts}$. □

Hence, the lack of a regional solution for a given ST-system $\mathfrak{ts}$ and $\widehat{=}$ means that $\mathfrak{ts}$ cannot be synthesised to an ENL$_{\widehat{=}}$-system.

A consequence of this result is that we have two possible ways to design a synthesis procedure for $\mathfrak{ts}$ and $\widehat{=}$. In both cases we first compute all the non-trivial regions of $\mathfrak{ts}$, and then proceed in one of the following two ways:

Method I: We check the conditions captured by Definition 2.1.7(**A1**–**A4**) for $\mathfrak{ts}$ and $\widehat{=}$ (see Algorithm 3). If they are satisfied, $\mathfrak{ts}$ is an ENLST$_{\widehat{=}}$-system and $\mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}}$ is a solution to the synthesis problem for $\mathfrak{ts}$ and $\widehat{=}$ (see Theorem 2.1.9). In this approach, we only need to build $\mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}}$ (see Algorithm 2) when the axioms of Definition 2.1.7(**A1**–**A4**) are satisfied.

Method II: We build $\mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}}$ using the discovered non-trivial regions (see Algorithm 2), and then check whether $\mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}}$ is an ENL$_{\widehat{=}}$-system (i.e., the sets of pre- and post-conditions, for every $e \in E$, are nonempty and disjoint, see Algorithm 5). If this is the case, we check whether the ST-system generated by $\mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}}$ is isomorphic to $\mathfrak{ts}$ (this procedure is straightforward as $\mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}}$ is deterministic and the only candidate for an isomorphism is a mapping associating $\mathfrak{R}_q$ with each state q of $\mathfrak{ts}$; see Algorithm 4).

Algorithm 1, computing the non-trivial regions of an ST-system, is central to both approaches to solving the synthesis problem for $\mathfrak{ts}$ and $\widehat{=}$ described above.
In line 2, Algorithm 1 removes selected thick transitions from $\mathfrak{ts}$ using the rules described in Section 3.2. The general safe rule for transition removal is given in Proposition 3.2.1.
For a special subclass of ST-systems we can safely remove all the thick transitions following Theorem 3.2.3. However, for an ordinary ST-system, this can only be done when Algorithm 1 is invoked in the context of Method II. This is supported by the next result.

Theorem 4.1.4. *Let* $\mathfrak{ts} = (Q, A, q_0)$ *be an* ST*-system, and* $\widehat{=}$ *be a co-location relation which partitions the set of events into sets of co-located sequential events. Furthermore, let* $\mathfrak{ts}'$ *be the* ST*-system obtained from* $\mathfrak{ts}$ *by deleting its all thick transitions and* $\mathfrak{net} = (\mathfrak{R}_{\mathfrak{ts}'}, E, F_{\mathfrak{ts}'}, \widehat{=}, \mathfrak{R}_{q_0})$, *where* $F_{\mathfrak{ts}'} = \{(\mathfrak{r}, e) \in \mathfrak{R}_{\mathfrak{ts}'} \times E \mid \mathfrak{r} \in {}^{\circ}e\} \cup \{(e, \mathfrak{r}) \in E \times \mathfrak{R}_{\mathfrak{ts}'} \mid \mathfrak{r} \in e^{\circ}\}$. *Then* $\mathfrak{ts}$ *is an* ENLST$_{\widehat{=}}$*-system iff* $\mathfrak{net}$ *is an* ENL$_{\widehat{=}}$*-system such that* $\mathfrak{ts}_{\mathfrak{net}} \cong \mathfrak{ts}$.

Proof: ($\Longrightarrow$) From Theorem 3.2.3 and Proposition 3.2.1, we have $\mathfrak{R}_{\mathfrak{ts}} = \mathfrak{R}_{\mathfrak{ts}'}$. Hence, $\mathfrak{net} = (\mathfrak{R}_{\mathfrak{ts}}, E, F_{\mathfrak{ts}}, \widehat{=}, \mathfrak{R}_{q_0}) = \mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}}$ and from Theorem 2.1.9 we have that $\mathfrak{net}$ is an ENL$_{\widehat{=}}$-system and $\mathfrak{ts}_{\mathfrak{net}} \cong \mathfrak{ts}$.

($\Longleftarrow$) Since $\mathfrak{net}$ is an ENL$_{\bumpeq}$-system and $\mathfrak{ts}_{\mathfrak{net}} \cong \mathfrak{ts}$, it follows from Proposition 4.1.2 that $\mathfrak{ts}$ is an ENLST$_{\bumpeq}$-system. $\square$

The above theorem states that if the isomorphism check succeeds with the regions of $\mathfrak{ts}'$ and $\mathfrak{net}$ is an ENL$_{\bumpeq}$-system, then $\mathfrak{R}_{\mathfrak{ts}} = \mathfrak{R}_{\mathfrak{ts}'}$. Otherwise, it is not important that we computed the regions of $\mathfrak{ts}'$ in Algorithm 1 rather than those of $\mathfrak{ts}$, because $\mathfrak{ts}$ cannot be synthesised to any ENL$_{\bumpeq}$-system anyway.
We cannot invoke safely Algorithm 1 with the thick transitions removal as in Theorem 3.2.3 in the context of Method I. The reason is that we need to check the axioms in Definition 2.1.7(**A1**–**A4**) for the regions of $\mathfrak{ts}$ and not for $\mathfrak{ts}'$ and they may be different (in general, we only have $\mathfrak{R}_{\mathfrak{ts}} \subseteq \mathfrak{R}_{\mathfrak{ts}'}$).

We now present Algorithms 2, 3 and 4.

Algorithm 2 returns the computed $\mathfrak{enl}^{\bumpeq}_{\mathfrak{ts}}$ net as a data structure (for further processing) and produces its visualisation that can be viewed in the WORKCRAFT tool.

Algorithm 2 Constructing the saturated net $\mathfrak{enl}^{\bumpeq}_{\mathfrak{ts}}$ for an ST-system $\mathfrak{ts} = (Q, A, q_0)$ on E and a co-location relation $\bumpeq$.

```
 1: function BUILD_$\mathfrak{enl}^{\bumpeq}_{\mathfrak{ts}}$($\mathfrak{R}_{\mathfrak{ts}}$, $\bumpeq$)
 2:     initialise $E$ to the set of events of $\mathfrak{ts}$
 3:     for every $e \in E$ do
 4:         represent $e$ as an event (box) in the diagram of $\mathfrak{enl}^{\bumpeq}_{\mathfrak{ts}}$
 5:         set $e$ with its locality according to $\bumpeq$ ▷ boxes of co-located events will be shaded
    in the same way
 6:     for every $\mathfrak{r} = (in, r, out) \in \mathfrak{R}_{\mathfrak{ts}}$ do
 7:         represent $\mathfrak{r}$ as a condition (circle) in the diagram of $\mathfrak{enl}^{\bumpeq}_{\mathfrak{ts}}$
 8:         if $q_0 \in r$ then
 9:             place a token (small black dot) inside the $\mathfrak{r}$ condition
10:         for every $e \in out$ do
11:             link $\mathfrak{r}$ to $e$ by a directed arc          ▷ $\mathfrak{r}$ will be a pre-condition of $e$
12:         for every $e \in in$ do
13:             link $e$ to $\mathfrak{r}$ by a directed arc          ▷ $\mathfrak{r}$ will be a post-condition of $e$
14:     return $\mathfrak{enl}^{\bumpeq}_{\mathfrak{ts}}$
```

Figure 4.1 shows the screenshot from WORKCRAFT depicting constructed $\mathfrak{enl}^{\bumpeq}_{\mathfrak{ts}_1}$ for the ENLST-system $\mathfrak{ts}_1$ in Figure 3.5, on p. 31.

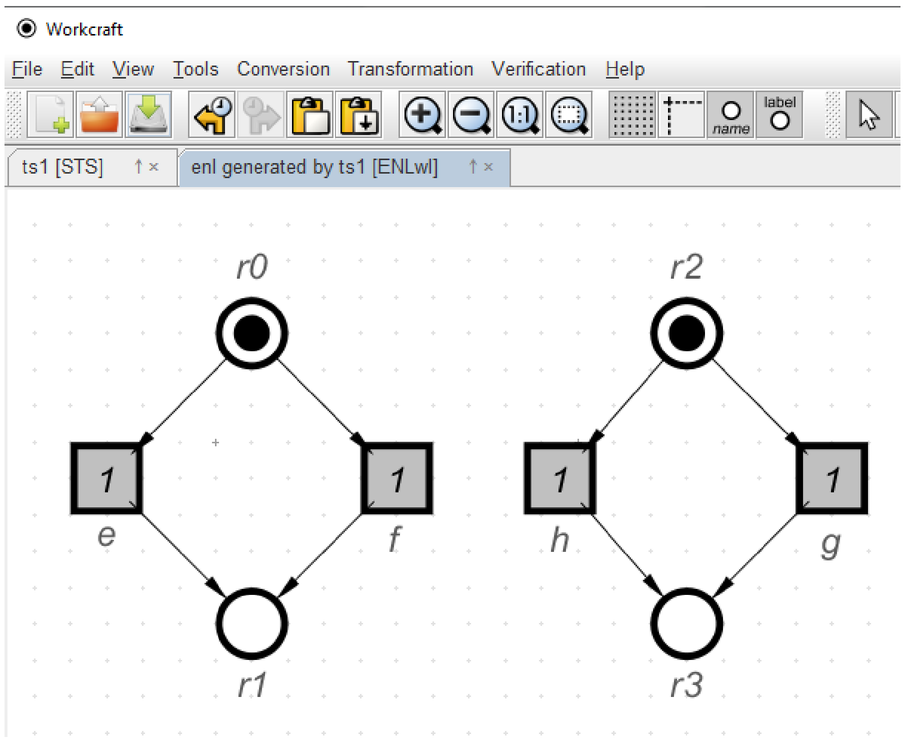

Workcraft
File Edit View Tools Conversion Transformation Verification Help
ts1 [STS]
enl generated by ts1 [ENLwl]
r0
r2
1
e
f
h
g
r1
r3

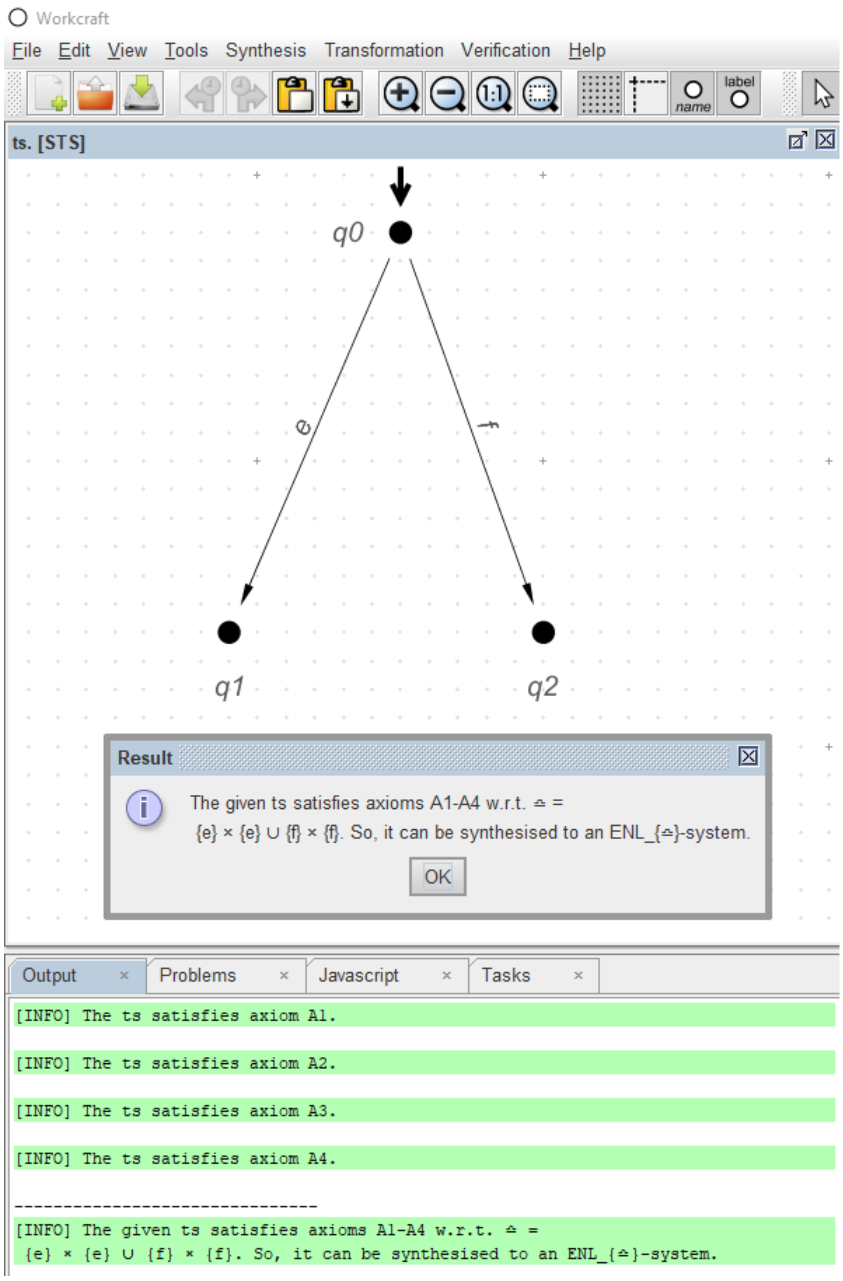
Fig. 4.1 Shows the screenshot from WORKCRAFT depicting the constructed $\mathfrak{enl}_{\mathfrak{ts}_1}^{\widehat{=}}$ for the ENLST-system $\mathfrak{ts}_1$ in Figure 3.5, on p. 31.

Algorithm 3 checks the axioms **A1–A4** (see Definition 2.1.7) for an ST-system $\mathfrak{ts}$ and a co-location relation $\widehat{=}$ to decide whether it is ENLST$_{\widehat{=}}$-system or not.

Algorithm 3 Checking axioms **A1-A4** for an ST-system $\mathfrak{ts}$ and a co-location relation $\widehat{=}$.

```
 1: function CHECK_AXIOMS($\mathfrak{ts}$, $\widehat{=}$)
 2:     initialise result to false
 3:     if $\mathfrak{ts}$ satisfies axiom A1 then
 4:         input $\mathfrak{ts}$ to Algorithm 1 to compute $\mathfrak{R}_{\mathfrak{ts}}$
 5:         if $\mathfrak{ts}$ satisfies axiom A2 then
 6:             if $\mathfrak{ts}$ satisfies axiom A3 then
 7:                 if $\mathfrak{ts}$ satisfies axiom A4 w.r.t. $\widehat{=}$ then
 8:                     $result = true$
 9:     if result then
10:         input $\mathfrak{R}_{\mathfrak{ts}}$ and $\widehat{=}$ to Algorithm 2 to build $\mathfrak{enl}_{\mathfrak{ts}}^{\widehat{=}}$     ▷ $\mathfrak{ts}$ is an ENLST$_{\widehat{=}}$-system
11:         return $\mathfrak{enl}_{\mathfrak{ts}}^{\widehat{=}}$     ▷ $\mathfrak{enl}_{\mathfrak{ts}}^{\widehat{=}}$ is an ENL$_{\widehat{=}}$-system
12:     else
13:         return "$\mathfrak{ts}$ is not an ENLST$_{\widehat{=}}$-system”
```

Figure 4.2 shows the screenshot from WORKCRAFT with the results of applying Algorithm 3 to a given ST-system, where the axioms **A1-A4** are satisfied, so it is synthesisable.

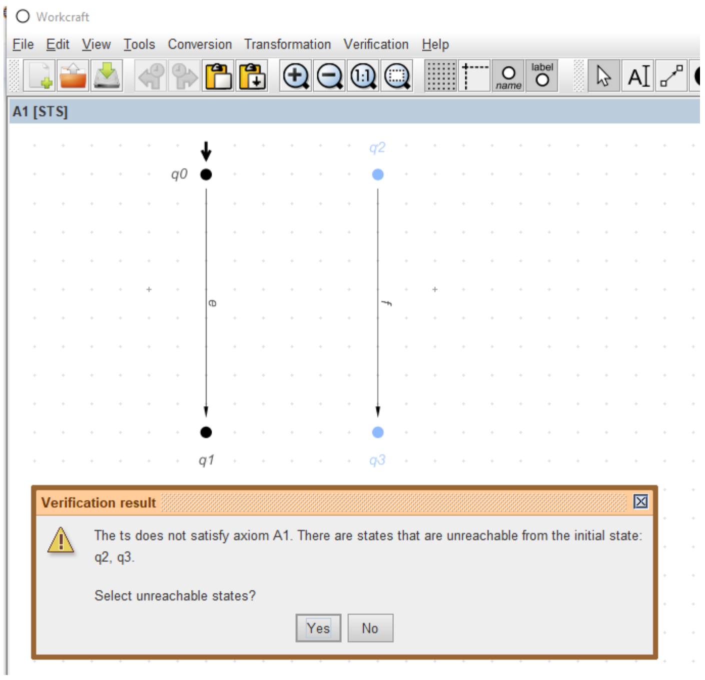

Workcraft
File Edit View Tools Synthesis Transformation Verification Help
ts. [STS]
q0
e
f
q1
q2
Result
OK
Output
Problems
Javascript
Tasks

The given ts satisfies axioms A1-A4 w.r.t. ≏ = {e} × {e} ∪ {f} × {f}. So, it can be synthesised to an ENL_{≏}-system.

```
[INFO] The ts satisfies axiom A1.

[INFO] The ts satisfies axiom A2.

[INFO] The ts satisfies axiom A3.

[INFO] The ts satisfies axiom A4.

-------------------------------
[INFO] The given ts satisfies axioms A1-A4 w.r.t. ≏ =
 {e} × {e} ∪ {f} × {f}. So, it can be synthesised to an ENL_{≏}-system.
```

Fig. 4.2 Shows the screenshot from WORKCRAFT with the positive example of applying Algorithm 3 to a given ST-system, where e and f have different localities.

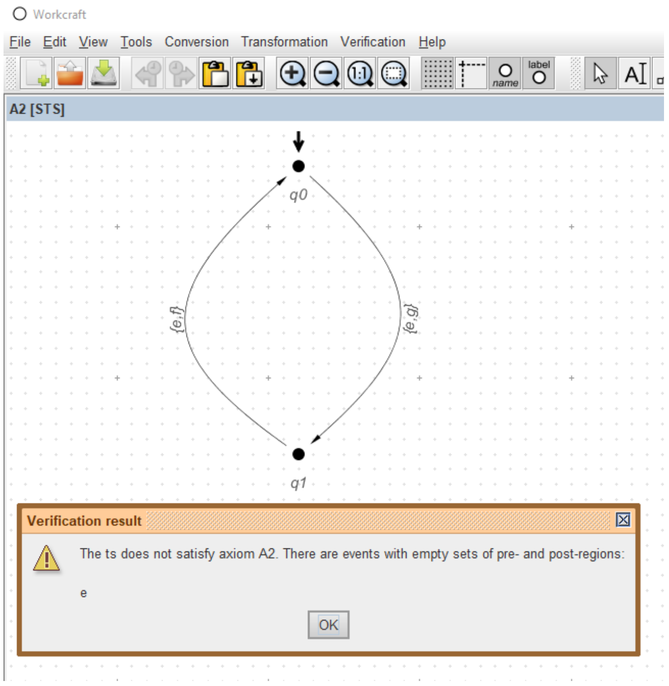
Figures 4.3, 4.4, 4.5 and 4.6 show the screenshots from WORKCRAFT with the results of applying Algorithm 3 to given ST-systems, where the axioms **A1**-**A4** are not satisfied, so they cannot be synthesised to ENLST-systems.


Workcraft
File Edit View Tools Conversion Transformation Verification Help
A1 [STS]
q0
q2
e
f
q1
q3
Verification result
Yes
No

The ts does not satisfy axiom A1. There are states that are unreachable from the initial state: q2, q3.

Select unreachable states?

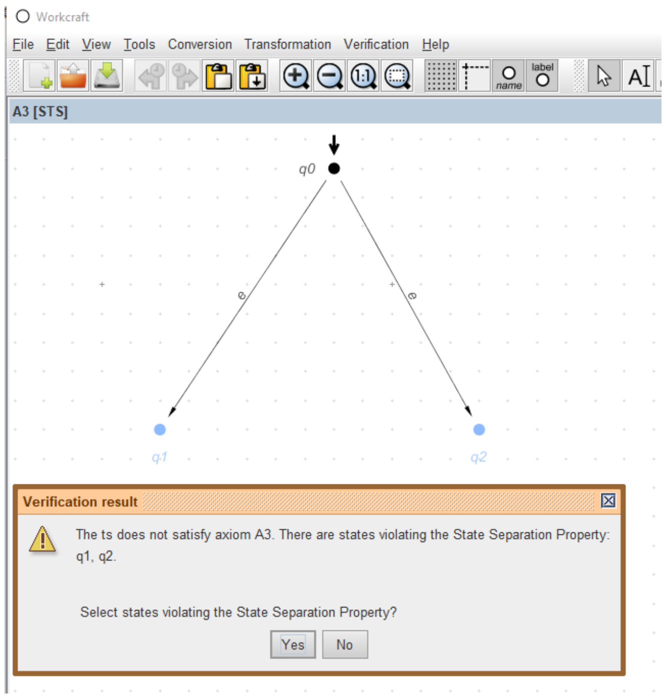
Fig. 4.3 The screenshot from WORKCRAFT showing the violation of axiom **A1** after applying Algorithm 3 to a given ST-system, where e and f have different localities.


Workcraft
File Edit View Tools Conversion Transformation Verification Help
A2 [STS]
q0
{e,f}
{e,g}
q1
Verification result
OK

The ts does not satisfy axiom A2. There are events with empty sets of pre- and post-regions:

e

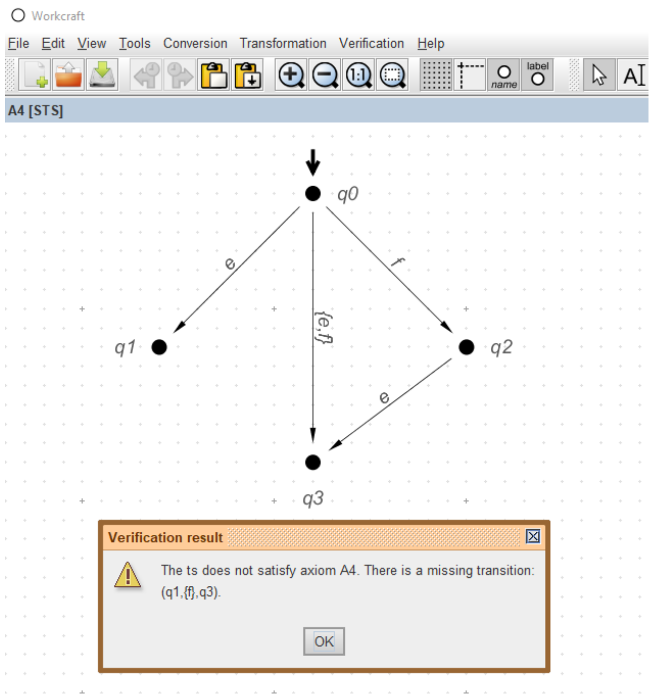
Fig. 4.4 The screenshot from WORKCRAFT showing the violation of axiom **A2** after applying Algorithm 3 to a given ST-system, where e and f and g have the same locality.


Workcraft
File Edit View Tools Conversion Transformation Verification Help
A3 [STS]
q0
e
e
q1
q2
Verification result
Yes
No

The ts does not satisfy axiom A3. There are states violating the State Separation Property: q1, q2.

Select states violating the State Separation Property?

Fig. 4.5 The screenshot from WORKCRAFT showing the violation of axiom **A3** after applying Algorithm 3 to a given ST-system.


Workcraft
File Edit View Tools Conversion Transformation Verification Help
A4 [STS]
q0
e
f
{e,f}
q1
q2
e
q3
Verification result
OK

The ts does not satisfy axiom A4. There is a missing transition: (q1,{f},q3).

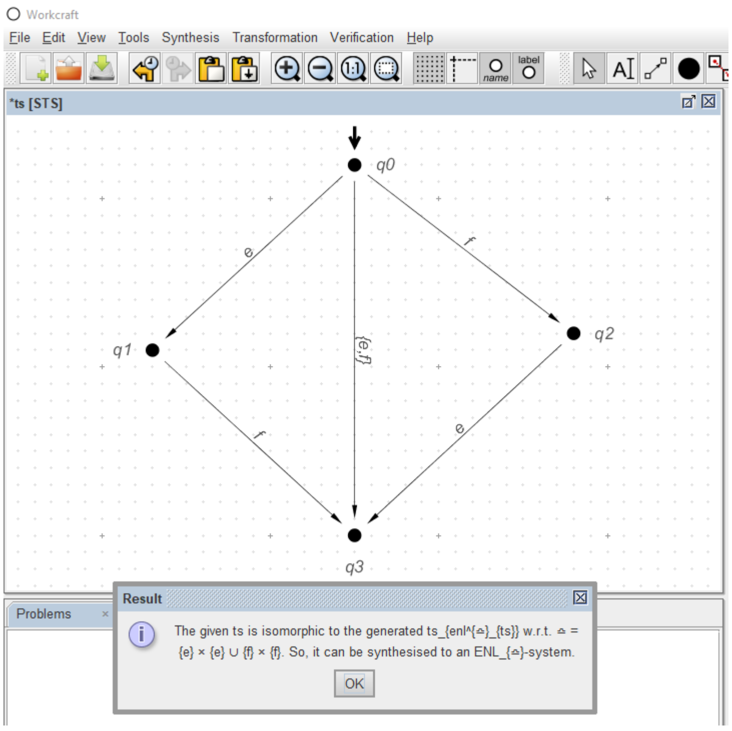
Fig. 4.6 The screenshot from WORKCRAFT showing the violation of axiom **A4** after applying Algorithm 3 to a given ST-system, where e and f have different localities.

Algorithm 4 checks whether an ST-system $\mathfrak{ts}$ is isomorphic to the ST-system generated by $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ (see Theorem 2.1.9). This procedure is straightforward as $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ is deterministic and the only candidate for an isomorphism is a mapping associating $\mathfrak{R}_q$ with each state q of $\mathfrak{ts}$.

Algorithm 4 Checking whether an ST-system $\mathfrak{ts} = (Q, A, q_0)$ is isomorphic to the ST-system generated by $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$: $\mathfrak{ts}' = (Q', A', q_0')$ (isomorphism used: $\psi(q) = \mathfrak{R}_q$, for every state q of $\mathfrak{ts}$, see Theorem 2.1.9).

```
1: function CHECK_ISOMORPHISM($\mathfrak{ts}$,$\mathfrak{ts}'$)
2:     initialise result to true
3:     for every transition $(q,U,q') \in A$ do
4:         if transition $(\psi(q),U,\psi(q')) \notin A'$ then
5:             $result = false$
6:             break
7:     for every transition $(q,U,q') \in A'$ do
8:         if transition $(\psi^{-1}(q),U,\psi^{-1}(q')) \notin A$ then        ▷ $\psi^{-1}$is the inverse of$\psi$
9:             $result = false$
10:            break
11:    if result then
12:        $\mathfrak{ts}$ is isomorphic to $\mathfrak{ts}'$ ($\mathfrak{ts}$ can be synthesised to an ENL$_{\hat{=}}$-system and $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ is the saturated solution to the synthesis problem for $\mathfrak{ts}$ and $\hat{=}$)
13:    else
14:        $\mathfrak{ts}$ is not an ENLST$_{\hat{=}}$-system        ▷ see Corollary 4.1.3
15:    return result
```

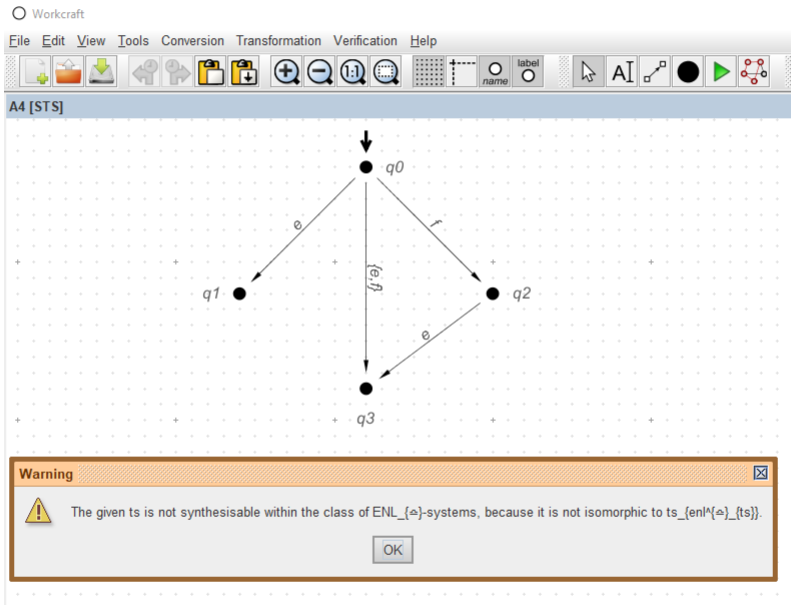
Figure 4.7 shows the screenshot from WORKCRAFT after the Algorithm 4 was applied to a given ST-system, where the ST-system generated by $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ is isomorphic to $\mathfrak{ts}$, so it can be synthesised. On the other hand, Figure 4.8 shows a negative result obtained after the application of Algorithm 4 to an ST-system.


Workcraft
File Edit View Tools Synthesis Transformation Verification Help
*ts [STS]
q0
e
f
{e,f}
q1
q2
f
e
q3
Problems
OK

Result

The given ts is isomorphic to the generated ts_{enl^{≏}_{ts}} w.r.t. ≏ = {e} × {e} ∪ {f} × {f}. So, it can be synthesised to an ENL_{≏}-system.

Fig. 4.7 Shows the screenshot from WORKCRAFT with the result of applying Algorithm 4 to the ST-system in Figure 2.2(a), where e and f have different localities.


Workcraft
File Edit View Tools Conversion Transformation Verification Help
A4 [STS]
q0
e
f
{e,f}
q1
q2
e
q3
OK

Warning

The given ts is not synthesisable within the class of ENL_{≏}-systems, because it is not isomorphic to ts_{enl^{≏}_{ts}}.

Fig. 4.8 Shows the screenshot from WORKCRAFT with the result of applying Algorithm 4 to a given ST-system, where e and f have different localities.

Algorithms 5 and 6 return positives answer if ENL-system is an $\text{ENL}_{\triangleq}$-system. Otherwise, they return negative answers.

Algorithm 5 Checking whether the synthesised net $enl^{\triangleq}_{\mathfrak{ts}}$ is an $\text{ENL}_{\triangleq}$-system.

```
1: function IS_ENL$_{\triangleq}$_SYSTEM($enl^{\triangleq}_{\mathfrak{ts}}$)        ▷ see Section 2.1.3
2:     initialise $result$ to true
3:     for $every\ e \in E$ do
4:         initialise $^{\circ}e$ and $e^{\circ}$ to $\varnothing$
5:         for $every\ \mathfrak{r} = (in, r, out) \in \mathfrak{R}_{\mathfrak{ts}}$ do        ▷ see Section 2.1.2
6:             if $e \in out$ then
7:                 add $\mathfrak{r}$ to $^{\circ}e$
8:             if $e \in in$ then
9:                 add $\mathfrak{r}$ to $e^{\circ}$
10:        if $^{\circ}e = \varnothing \vee e^{\circ} = \varnothing \vee {}^{\circ}e \cap e^{\circ} \neq \varnothing$ then
11:            $result = false$
12:            break
13:    return $result$
```

Algorithm 6 Checking whether a tuple $\mathfrak{enl} = (B, E, F, \triangleq, c_0)$ is an $\text{ENL}_{\triangleq}$-system.

```
1: function IS_ENL$_{\triangleq}$_SYSTEM($\mathfrak{enl}$)
2:     initialise $result$ to true
3:     for $every\ e \in E$ do
4:         initialise $^{\bullet}e$ and $e^{\bullet}$ to $\varnothing$
5:         for $every\ b \in B$ do
6:             if $(b, e) \in F$ then
7:                 add $b$ to $^{\bullet}e$
8:             if $(e, b) \in F$ then
9:                 add $b$ to $e^{\bullet}$
10:        if $^{\bullet}e = \varnothing \vee e^{\bullet} = \varnothing \vee {}^{\bullet}e \cap e^{\bullet} \neq \varnothing$ then
11:            $result = false$
12:            break
13:    return $result$
```

4.2 Results of the experiments

We tested Method I and Method II to compare the efficiency of the two approaches for the synthesis procedure. In our tests, we used in the first instance, the same selected examples that we used for Algorithm 1. The experimental results showed that Method I performed a bit better than Method II on small examples such as $\mathfrak{ts}_1$, $\mathfrak{ts}_2$, $\mathfrak{ts}_3$, and $\mathfrak{ts}_4$. When using bigger examples, like $\mathfrak{ts}_5$, the reverse was true: Method II performed better than Method I (see Figure 4.9). This better performance of Method II on bigger examples was confirmed, when we tested the algorithms on other ST-systems with up to 1296 states. The main reason for this result is the fact that checking the axiom **A4** in Algorithm 3 consumes a considerable time.

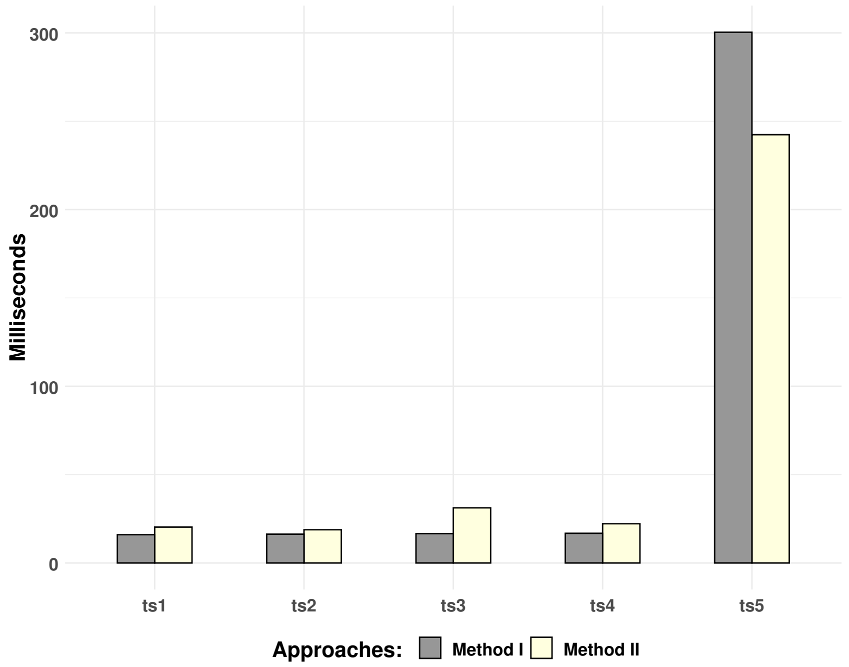


Fig. 4.9 A diagram showing the comparison between the execution times of Method I and Method II, when applied to $\mathfrak{ts}_1$, $\mathfrak{ts}_2$, $\mathfrak{ts}_3$, $\mathfrak{ts}_4$ and $\mathfrak{ts}_5$ ST-systems.

To test the efficiency of the two approaches to the synthesis procedure given by Method I and Method II described in Section 4.1 and to see how the performance of Method I and Method II scale with the increasing sizes of inputs, we used ST-systems generated by nets modelling server-client systems like the net in Figure 4.10 showing the interactions of two servers and two clients. In Figure 4.10, we use the following denotations for events: Snd_{ij} (send), Rcv_{ij} (receive), Srv_{ij} (serve) and Res_{ij} (result), where i is the local number of a server or a client and j is the destination number of a server or a client. Furthermore, the conditions of clients (servers) are denoted by C_{ij} (S_{ij} respectively), where i is the index of a condition within a subsystem and j is the index of the subsystem. The subsystems are connected by buffer conditions: $b_1,\dots,b_8$. The server-client ST-systems are denoted by $\mathfrak{ts}_{i,j}^{ser-cl}$, where the index i denotes the number of servers and j denotes the number of clients. The experimental results are summarised in Table 4.1. Note that, except for the very small examples, algorithms of Method II perform better than those of Method I because checking the axioms **A1**-**A4** of Definition 2.1.7 is computationally expensive (especially the verification of axiom **A4**).

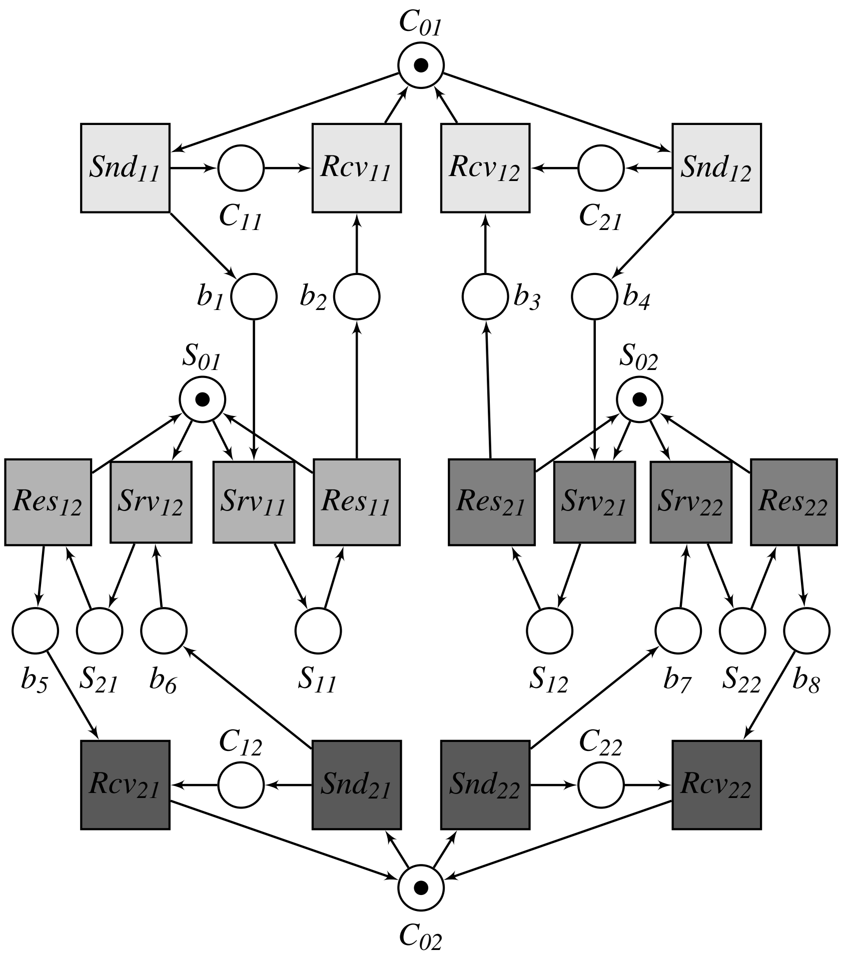


Fig. 4.10 An ENL-system with two clients and two servers that generates the ST-system $\mathfrak{ts}^{ser-cl}_{2,2}$ (later also called $\mathfrak{ts}_9$).

Table 4.1 Comparing execution times of synthesis algorithms of Method I and Method II for selected ST-systems. The results in columns 6 and 7 include the time of executing Algorithm 1 to derive non-trivial regions for each of the tested ST-systems (see column 4).

$\mathfrak{ts}^{ser-cl}$	$\lvert\mathbf{Q}\rvert$	$\lvert\mathbf{E}\rvert$	Algorithm 1	$\lvert\mathfrak{R}_{\mathfrak{ts}}\rvert$	Method I	Method II
$\mathfrak{ts}^{ser-cl}_{1,2}$	15	8	0.0114 s	30	0.0326 s	0.0402 s
$\mathfrak{ts}^{ser-cl}_{1,3}$	54	12	0.2724 s	50	0.4137 s	0.3675 s
$\mathfrak{ts}^{ser-cl}_{2,2}$	47	16	0.9029 s	256	1.8222 s	1.1693 s
$\mathfrak{ts}^{ser-cl}_{2,3}$	176	20	24.0480 s	276	55.0885 s	24.8932 s
$\mathfrak{ts}^{ser-cl}_{3,2}$	97	24	122.4600 s	2050	415.7800 s	127.7800 s
$\mathfrak{ts}^{ser-cl}_{3,3}$	1446	36	68806.3500 s	3094	1188718.1600 s	68972.3000 s
$\mathfrak{ts}^{ser-cl}_{4,2}$	165	32	61182.4100 s	16388	205830.0400 s	61427.6900 s

Chapter 5

ENL-systems with localised conflicts

So far it was assumed that the co-location relation is known in advance. However, in some cases it can be 'deduced' from the structure of an ST-system. In particular, there is a subclass of ENLST-systems for which one can calculate 'canonical' co-location relations, from which any further valid co-location relations can be obtained. Such a subclass corresponds to ENL-systems with localised conflicts.

5.1 ENL/LC-systems

Definition 5.1.1 ([27]). *An* ENL*-system is* ENL*-system with* localised conflicts *(or* ENL/LC*-system) if no conflicting non-co-located events are resource enabled at any reachable case.* $\diamondsuit$

In other words, an ENL$_{\bumpeq}$-system $\mathfrak{enl}$ is an ENL/LC-system if, for all events $e \neq f$ and each reachable case c, $e \leftrightharpoons_{\mathfrak{enl}} f$ and $e \not\simeq f$ together imply $\{e,f\} \not\subseteq \mathit{resenabled}(c)$. As an implication with a *false* antecedent is always *true*, any ENL-system without conflicts is trivially an ENL/LC-system.
It can be argued that ENL-systems which are not ENL/LC-systems are not directly implementable as any mechanism for resolving conflicts between actions of a concurrent system must be local (see [8]). Hence, ENL/LC-systems are highly relevant from the practical point of view.
For some ENL-systems, whose conflicts are not localised, we can re-define their co-location relations so that the conflicts are localised without changing the generated ST-systems. Nevertheless, ENL/LC-systems are a proper subclass of ENL-systems. There are examples of ST-systems, like $\mathfrak{ts}_6$ in Figure 5.1(a) and $\mathfrak{ts}_7$ in Figure 5.2(a), which cannot be synthesised

to any ENL/LC-system. We can only synthesise them to ENL-systems, where conflicts are not localised (see Figure 5.1(b) and Figure 5.2(b)).

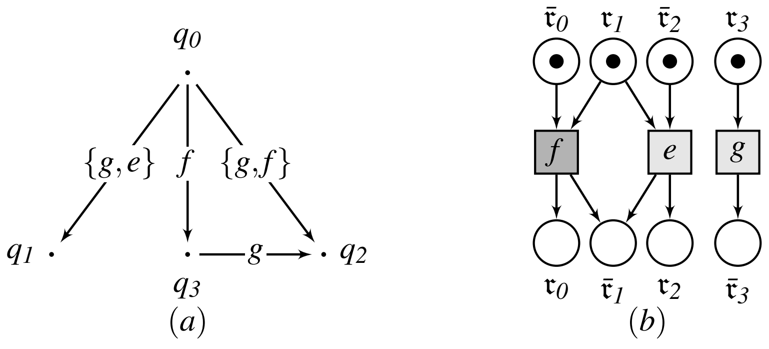

q_0
$\{g,e\}$
f
$\{g,f\}$
q_1
g
q_2
q_3
(a)
$\bar{\mathfrak{r}}_0$ $\mathfrak{r}_1$ $\bar{\mathfrak{r}}_2$ $\mathfrak{r}_3$
f e g
$\mathfrak{r}_0$ $\bar{\mathfrak{r}}_1$ $\mathfrak{r}_2$ $\bar{\mathfrak{r}}_3$
(b)

Fig. 5.1 An ENLST-system $\mathfrak{ts}_6$, where e and g are co-located and f belongs to a different locality (a); and the ENL-system resulting from its synthesis (b).

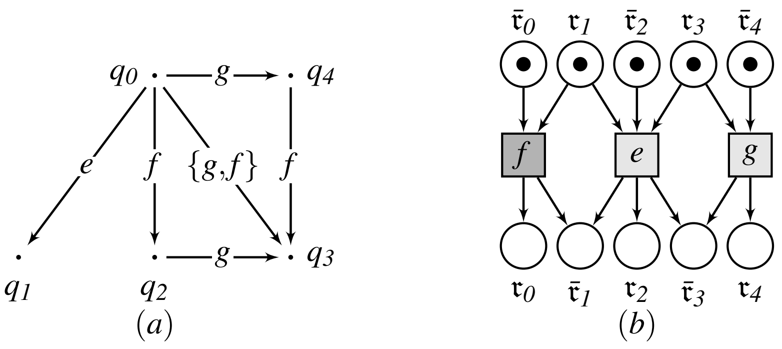

q_0
g
q_4
e
f
$\{g,f\}$
f
g
q_3
q_1
q_2
(a)
$\bar{\mathfrak{r}}_0$ $\mathfrak{r}_1$ $\bar{\mathfrak{r}}_2$ $\mathfrak{r}_3$ $\bar{\mathfrak{r}}_4$
f e g
$\mathfrak{r}_0$ $\bar{\mathfrak{r}}_1$ $\mathfrak{r}_2$ $\bar{\mathfrak{r}}_3$ $\mathfrak{r}_4$
(b)

Fig. 5.2 An ENLST-system $\mathfrak{ts}_7$, where e and g are co-located and event f belongs to a different locality (a); the ENL-system resulting from its synthesis (b).

We will now consider the following synthesis problem, where the discovery of the suitable co-location relation is a part of the synthesis procedure.

Problem 2 ([27]). *Given an* ST*-system* $\mathfrak{ts}$ *find an effective way of checking whether there is an* ENL/LC*-system which is a net realisation of* $\mathfrak{ts}$. *If the answer is positive construct such an* ENL/LC*-system.* ◇

First, we show that conflicts between events in ENLST-systems are present in their regional net realisations.

Proposition 5.1.2. *Let* $\mathfrak{ts}$ *be an* ENLST$_{\hat{=}}$*-system. Then* $e \leftrightharpoons_{\mathfrak{ts}} f$ *implies* $e \leftrightharpoons_{\mathfrak{enl}^{\hat{=}}_{\mathfrak{ts}}} f$, *for all events* e *and* f.

Proof: Let q be a state of $\mathfrak{ts}$ such that $e \leftrightharpoons^{q}_{\mathfrak{ts}} f$. Then, we have $e, f \in E_q$ (and consequently $^{\circ}e \cup {}^{\circ}f \subseteq \mathfrak{R}_q$, $(e^{\circ} \cup f^{\circ}) \cap \mathfrak{R}_q = \varnothing$), and $\{e, f\} \nsubseteq U$, for all $U \in allSteps_q$. In $\mathfrak{enl}^{\frown}_{\mathfrak{ts}}$, where $\mathfrak{R}_q$ is a reachable case, we have $\{e\}, \{f\} \in resenabled(\mathfrak{R}_q)$ and $\{e, f\} \nsubseteq U$, for all $U \in enabled(\mathfrak{R}_q)$. This can only happen if $\{e, f\} \nsubseteq U$, for all $U \in resenabled(\mathfrak{R}_q)$ (see Fact 3). Hence, e and f are in conflict in $\mathfrak{enl}^{\frown}_{\mathfrak{ts}}$ (i.e., $e \leftrightharpoons_{\mathfrak{enl}^{\frown}_{\mathfrak{ts}}} f$). □

The implication in Proposition 5.1.2 cannot be reversed. A pair of events, e and f, can be in conflict in an ENL-system $\mathfrak{enl}$ (sharing a pre-condition or a post-condition), but they might not be in conflict in $\mathfrak{ts}_{\mathfrak{enl}}$, because of not being enabled together at any reachable case of $\mathfrak{enl}$. As an example, consider the ENLST-system $\mathfrak{ts}_3$ in Figure 3.1, on p. 20. Its net realisation in Figure 3.2, on p. 21, contains a common pre-condition $\bar{\mathfrak{r}}_9$ for c_1 and c_4, but there is no state of $\mathfrak{ts}_3$ at which both c_1 and c_4 are included in the labels of the outgoing transitions.

5.2 Solving Problem 2

As shown in [27], co-location relations of ENL/LC-systems can to a significant extent be characterised structurally (and locally). More precisely, if q is a state of the ST-system generated by an ENL/LC-system, then two distinct events in E_q are co-located *iff* either there is no step in $allSteps_q$ to which the two events belong, or there is a step in $minSteps_q$ to which the two events belong. This motivated the introduction of a *canonical co-location relation* $\doteqdot^{\mathfrak{ts}}_{min}$ for an ST-system $\mathfrak{ts} = (Q, A, q_0)$ for which there is an ENL/LC-system realisation. Such a co-location relation is defined as:

$$\doteqdot^{\mathfrak{ts}}_{min} \;=\; \Big(\bigcup_{q \in Q} \doteqdot^{\mathfrak{ts},q}\Big)^{*}, \tag{5.1}$$

where

$$\doteqdot^{\mathfrak{ts},q} \;=\; \bigcup_{U \in minSteps_q} U \times U \;\cup\; \Big((E_q \times E_q) \setminus \bigcup_{U \in allSteps_q} U \times U\Big), \tag{5.2}$$

for every $q \in Q$. Moreover, $\doteqdot^{\mathfrak{ts}}_{min}$ is *consistent* with $\mathfrak{ts}$ if $\doteqdot^{\mathfrak{ts}}_{min} \cap (E_q \times E_q)$ is equal to $\doteqdot^{\mathfrak{ts},q}$, for every state $q \in Q$. The asterisk in Eq.(5.1) denotes the transitive closure operator.
A consistent canonical co-location relation can be employed in the synthesis of ENL/LC-systems. If found, it is the finest (w.r.t. the number of equivalence classes) possible co-location relation for $\mathfrak{ts}$. In other words, $\doteqdot^{\mathfrak{ts}}_{min}$ is included in any other suitable co-location relation $\doteqdot$, each equivalence class of $\doteqdot^{\mathfrak{ts}}_{min}$ is included in an equivalence class of $\doteqdot$, and the number of equivalence classes of $\doteqdot^{\mathfrak{ts}}_{min}$ is greater than the number of equivalence classes of $\doteqdot$. A consequence of this property is stated in the following result.

Proposition 5.2.1. *Let* $\mathfrak{ts}$ *be an* ST*-system such that* $\simeq^{\mathfrak{ts}}_{min}$ *is consistent with* $\mathfrak{ts}$*. If* $\mathfrak{ts}$ *is* <u>*not*</u> *an* ENLST$_{\simeq^{\mathfrak{ts}}_{min}}$*-system, then* $\mathfrak{ts}$ *is* <u>*not*</u> *an* ENLST*-system.*

Proof: Suppose to the contrary that $\mathfrak{ts}$ is not an ENLST$_{\simeq^{\mathfrak{ts}}_{min}}$-system, but $\mathfrak{ts}$ is an ENLST$_{\simeq}$-system, for some co-location relation $\simeq$. Hence, Definition 2.1.7(**A1**–**A4**) is satisfied for $\mathfrak{ts}$ and $\simeq$.
Since Definition 2.1.7(**A1**–**A3**) does not depend on the choice of a co-location relation, it is also satisfied for $\mathfrak{ts}$ and $\simeq^{\mathfrak{ts}}_{min}$. Hence, we have that Definition 2.1.7(**A4**) is not satisfied for $\mathfrak{ts}$ and $\simeq^{\mathfrak{ts}}_{min}$, although it is satisfied for $\mathfrak{ts}$ and $\simeq$.
Let Definition 2.1.7(**A4**) fails to be satisfied for state q and step U such that $U \in allSteps_q$ in $\mathfrak{ts}$, when using the co-location relation $\simeq^{\mathfrak{ts}}_{min}$. As it does not fail to be satisfied when using $\simeq$, there exists $e \notin U$ such that $e \simeq^{\mathfrak{ts}}_{min} U$ and $U \cup \{e\} \in regenabled(q)$ and, at the same time, $e \not\simeq U$ since $\mathfrak{ts}$ is an ENLST$_{\simeq}$-system. Hence, there is an event $f \in U$ such that $e \simeq^{\mathfrak{ts}}_{min} f$ and $e \not\simeq f$. However, the results of [27] showed that a consistent $\simeq^{\mathfrak{ts}}_{min}$, if it exists, is the finest

Algorithm 7 Solution to Problem 2 for an ST-system $\mathfrak{ts} = (Q, A, q_0)$.

```
 1: procedure SOLVING_PROBLEM_2($\mathfrak{ts}$)
 2:     initialise $\simeq^{\mathfrak{ts}}_{min}$ to $\varnothing$
 3:     for every $q \in Q$ do
 4:         calculate $allSteps_q$, $minSteps_q$ and $E_q$
 5:         calculate $\simeq^{\mathfrak{ts},q} = \bigcup_{U \in minSteps_q} U \times U \cup \Big((E_q \times E_q) \setminus \bigcup_{U \in allSteps_q} U \times U\Big)$
 6:     calculate $\simeq^{\mathfrak{ts}}_{min} = \big(\bigcup_{q \in Q} \simeq^{\mathfrak{ts},q}\big)^*$        ▷ see Eq.(5.1)
 7:     initialise result to true
 8:     for every $q \in Q$ do
 9:         if $\simeq^{\mathfrak{ts},q}$ is not equal to $\simeq^{\mathfrak{ts}}_{min} \cap (E_q \times E_q)$ then
10:             $result = false$
11:             break
12:     if result then
13:         calculate $\mathfrak{R}_{\mathfrak{ts}}$
14:         calculate $\mathfrak{enl} = \mathfrak{enl}^{\simeq^{\mathfrak{ts}}_{min}}_{\mathfrak{ts}}$ using $\mathfrak{R}_{\mathfrak{ts}}$ and $\simeq^{\mathfrak{ts}}_{min}$
15:         if $\mathfrak{enl}$ is an ENL$_{\simeq^{\mathfrak{ts}}_{min}}$-system and $\mathfrak{ts}_{\mathfrak{enl}} \cong \mathfrak{ts}$ then
16:             return $\simeq^{\mathfrak{ts}}_{min}$        ▷ $\mathfrak{enl}$ is ENL/LC-system realisation of $\mathfrak{ts}$
17:         else
18:             return "$\mathfrak{ts}$ is not an ENLST-system"        ▷ see Proposition 5.2.1
19:     else
20:         return "Problem 2 is not feasible"        ▷ $\mathfrak{ts}$ may still be ENLST-system
```

possible co-location relation for $\mathfrak{ts}$ (i.e., for all events $e \neq f$, $e \simeq^{\mathfrak{ts}}_{min} f$ implies $e \simeq f$ in this case). Hence, we obtained a contradiction. □

Using Eq.(5.1) and Proposition 5.2.1, an algorithmic solution to Problem 2 is given as Algorithm 7. Note that Algorithm 7 when applied to ST-systems $\mathfrak{ts}_6$ and $\mathfrak{ts}_7$ (in Figures 5.1(a) and 5.2(a), respectively) would exit with the message "Problem 2 is not feasible" (see line 20 of the algorithm; see also Figure 5.3 showing the screenshot from WORKCRAFT after Algorithm 7 was applied to $\mathfrak{ts}_6$).

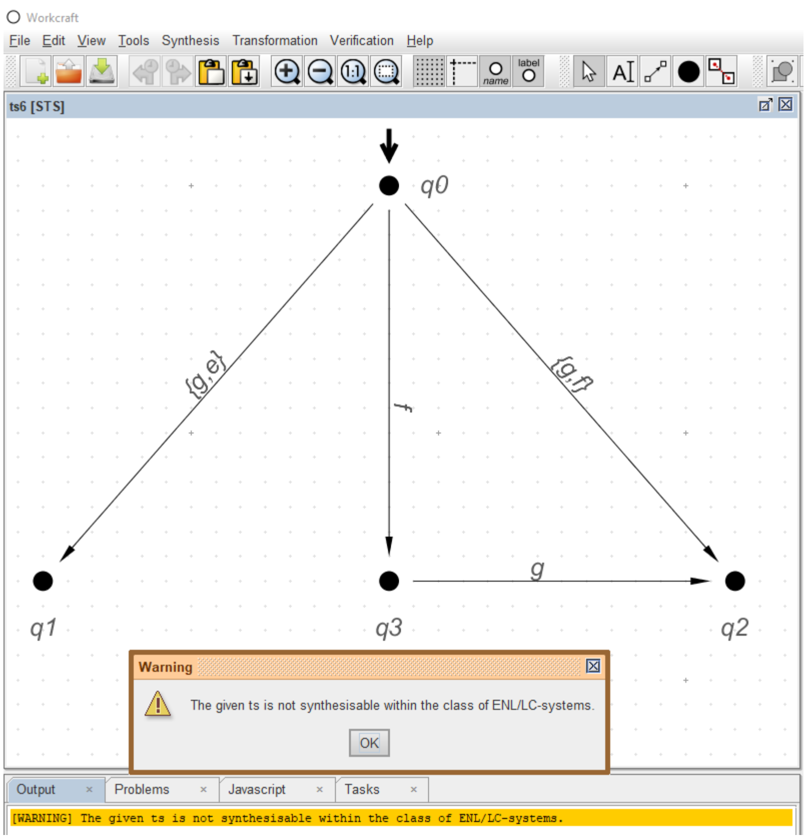
Algorithm 7 solution to Problem 2 takes advantage of the structural (and local) properties of the original ST-system. If it succeeds, we obtain an ENL/LC-system which solves the synthesis problem. Moreover, the approach can be extended to tackle the following related problem.

Problem 3. *Given an* ST*-system* $\mathfrak{ts}$ *and a co-location relation* $\simeq$*, find an effective way of checking whether there is an* ENL$_{\simeq}$*-system with localised conflicts which is a net realisation of* $\mathfrak{ts}$*. If the answer is positive construct such an* ENL$_{\simeq}$*-system.* ♢

To start with, Problem 3 can only have a solution if $\simeq^{\mathfrak{ts}}_{min}$ provides a solution to Problem 2. In such a case, one can use another result established in [27] which characterizes all suitable co-location relations for $\mathfrak{ts}$.
Let $G^{\mathfrak{ts}}$ be an undirected graph whose vertices are the equivalence classes of $\simeq^{\mathfrak{ts}}_{min}$, and there is an edge between vertices V and V' if there is a state q of $\mathfrak{ts}$ and two events, $e \in V$ and $f \in V'$, such that $e, f \in E_q$. Then $\simeq$ provides a solution to Problem 3 if it can be computed as a solution to the vertex colouring problem for $G^{\mathfrak{ts}}$, where we colour two vertices (equivalence classes of $\simeq^{\mathfrak{ts}}_{min}$) using the same colour iff there is an equivalence class of $\simeq$ in which they are both included. Moreover, if the answer is positive, we can simply replace $\simeq^{\mathfrak{ts}}_{min}$ with $\simeq$ in the solution to Problem 2 to address the second part of Problem 3.

5.2.1 Results of the experiments

To test Algorithm 7, we used the examples that we already selected for Algorithm 1. Our goal was to check whether it provides a positive answer to the synthesis problem. As we have described before, the selected examples for our experiments consider ST-systems representing nets with different characteristics, and if it comes to conflicts there are two types of ST-systems: ST-systems generated by nets without conflicts ($\mathfrak{ts}_3, \mathfrak{ts}_4, \mathfrak{ts}_5$) or with conflicts ($\mathfrak{ts}_1, \mathfrak{ts}_2$), where $\mathfrak{ts}_1$ has localised conflicts. According to the experimental results, Algorithm 7 gives positive answers for all chosen ST-systems generated by nets without conflicts ($\mathfrak{ts}_3, \mathfrak{ts}_4, \mathfrak{ts}_5$), which are trivially ENL/LC-systems. Also, $\mathfrak{ts}_1$ and $\mathfrak{ts}_2$ can be synthesised to ENL/LC-systems. The discovered $\simeq^{\mathfrak{ts}_1}_{min}$ is the same as the co-location relation used for $\mathfrak{ts}_1$ in



The given ts is not synthesisable within the class of ENL/LC-systems.

```
[WARNING] The given ts is not synthesisable within the class of ENL/LC-systems.
```

Fig. 5.3 Shows the screenshot from WORKCRAFT after the Algorithm 7 was applied to $\mathfrak{ts}_6$ in Figures 5.1(a).

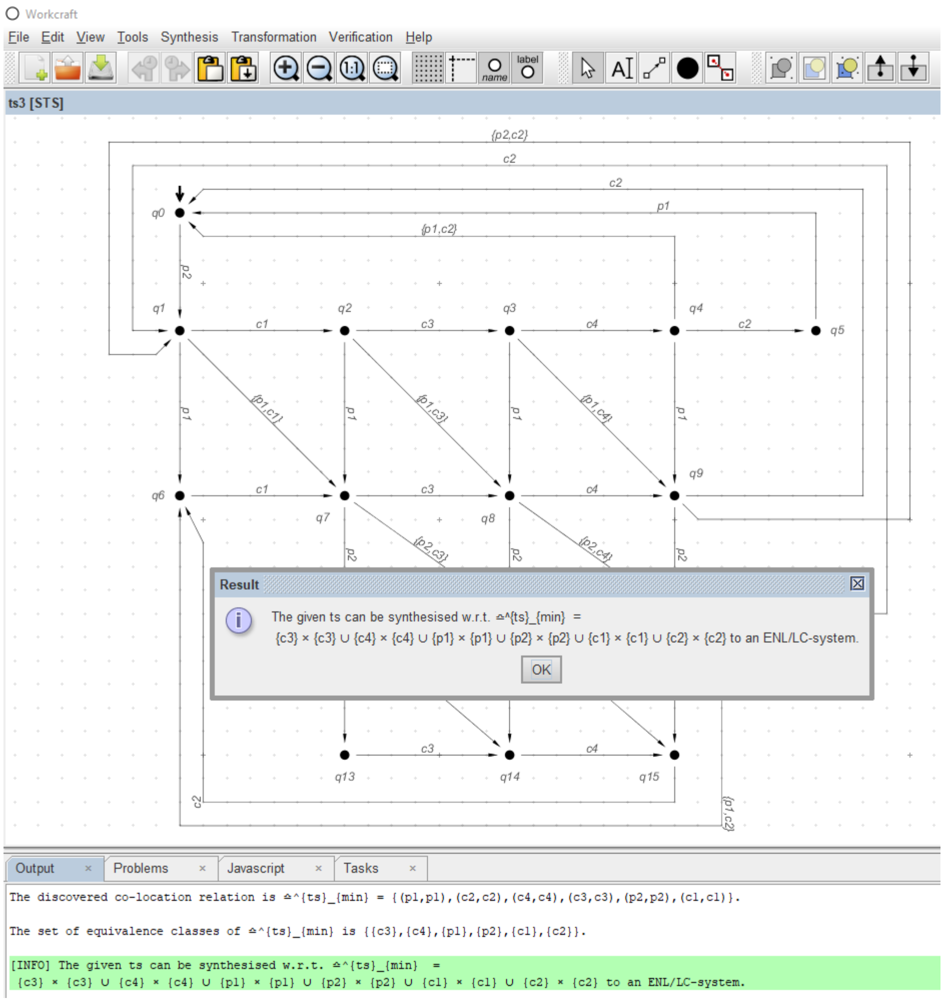
Figure 3.5, on p. 31. However, the discovered $\bumpeq^{\mathfrak{ts}_2}_{min}$ is different than the co-location relation used for $\mathfrak{ts}_2$ in Figure 3.3, on p. 22, but whichever of them we use for the net synthesised from $\mathfrak{ts}_2$ it would generate the same ST-system. The ST-system $\mathfrak{ts}_2$ in Figure 3.3 has conflict between non co-located events g and h, but the discovered co-location relation $\bumpeq^{\mathfrak{ts}_2}_{min}$ localises this conflict without changing the behaviour of the net synthesised from $\mathfrak{ts}_2$.

The results of the experiments for the selected examples are as follows:

- The discovered co-location relation of $\mathfrak{ts}_1$ in Figure 3.5, on p. 31, is $\bumpeq^{\mathfrak{ts}_1}_{min} = \{(g,f),(f,e),(h,g),(e,e),(g,g),(h,h),(f,f),(f,g),(e,f),(g,h),(e,g),(f,h),(e,h),(h,e),(g,e),(h,f)\}$. The set of equivalence classes of $\bumpeq^{\mathfrak{ts}_1}_{min}$ is $\{\{e,f,g,h\}\}$.
- The discovered co-location relation of $\mathfrak{ts}_2$ in Figure 3.3, on p. 22, is $\bumpeq^{\mathfrak{ts}_2}_{min} = \{(f,e),(h,g),(g,g),(e,e),(f,f),(h,h),(e,f),(g,h)\}$. The set of equivalence classes of $\bumpeq^{\mathfrak{ts}_2}_{min}$ is $\{\{e,f\},\{g,h\}\}$.
- The discovered co-location relation of $\mathfrak{ts}_3$ in Figure 3.1, on p. 20, is $\bumpeq^{\mathfrak{ts}_3}_{min} = \{(p_1,p_1),(c_3,c_3),(c_4,c_4),(c_2,c_2),(c_1,c_1),(p_2,p_2)\}$. The set of equivalence classes of $\bumpeq^{\mathfrak{ts}_3}_{min}$ is $\{\{c_3\},\{c_4\}$

$,\{p_1\},\{p_2\},\{c_1\},\{c_2\}\}$. The output from WORKCRAFT for this example is shown in Figure 5.4.

- The discovered co-location relation of $\mathfrak{ts}_4$ in Figure 3.6, on p. 31, is $\bumpeq^{\mathfrak{ts}_4}_{min} = \{(p_1,p_1),(c_3,c_3),(c_2,c_2),(c_4,c_4),(p_2,p_2),(c_1,c_1),(c_3,c_2),(c_2,c_3),(c_1,c_4),(c_4,c_1)\}$. The set of equivalence classes of $\bumpeq^{\mathfrak{ts}_4}_{min}$ is $\{\{p_1\},\{p_2\},\{c_3,c_2\},\{c_4,c_1\}\}$.
- The discovered co-location relation of the ST-system generated by an ENL-system in Figure 3.7, on p. 32, ($\mathfrak{ts}_5$) is $\bumpeq^{\mathfrak{ts}_5}_{min} = \{(a,a),(b,b),(c,c),(d,d),(e,e),(f,f),(g,g),(h,h),(i,i)\}$. The set of equivalence classes of $\bumpeq^{\mathfrak{ts}_5}_{min}$ is $\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\},\{h\},\{i\}\}$.



The given ts can be synthesised w.r.t. ≏^{ts}_{min} =
{c3} × {c3} ∪ {c4} × {c4} ∪ {p1} × {p1} ∪ {p2} × {p2} ∪ {c1} × {c1} ∪ {c2} × {c2} to an ENL/LC-system.

```
The discovered co-location relation is ≏^{ts}_{min} = {(p1,p1),(c2,c2),(c4,c4),(c3,c3),(p2,p2),(c1,c1)}.

The set of equivalence classes of ≏^{ts}_{min} is {{c3},{c4},{p1},{p2},{c1},{c2}}.

[INFO] The given ts can be synthesised w.r.t. ≏^{ts}_{min}  =
 {c3} × {c3} ∪ {c4} × {c4} ∪ {p1} × {p1} ∪ {p2} × {p2} ∪ {c1} × {c1} ∪ {c2} × {c2} to an ENL/LC-system.
```

Fig. 5.4 Shows the screenshot from WORKCRAFT with the discovered co-location relation of $\mathfrak{ts}_3$ in Figures 3.1; and the set of equivalence classes of $\bumpeq^{\mathfrak{ts}_3}_{min}$.

The discovered co-location relations for the ST-systems $\mathfrak{ts}_3$, $\mathfrak{ts}_4$ and $\mathfrak{ts}_5$ are not as shown in Figure 3.2, Figure 1.1 and Figure 3.7 of the nets that generate them (see these figures on pages 21, 4, and 32). They are the finest possible co-location relations that can be discovered for these ST-systems. The co-location relations shown in the mentioned figures are further valid co-location relations that can be computed from the discovered 'canonical' ones.

To test how the performance of Algorithms 7 scales with the increasing sizes of inputs, we used the same ST-systems as listed in Table 3.2. The experimental results are summarised in Table 5.1, where the results from Table 3.2 related to Algorithm 1 are quoted for the sake of comparison. As Algorithm 7 invokes Algorithm 1, it can be seen that the main computational effort of Algorithm 7 comes from executing Algorithm 1.

Table 5.1 Comparing execution times of Algorithm 1 and Algorithm 7.

$\mathfrak{ts}$	|Q|	|E|	**Algorithm 1**	**Algorithm 7**
$\mathfrak{ts}_{3,3}$	64	9	0.1954 s	0.2499 s
$\mathfrak{ts}_{3,4}$	125	12	2.0029 s	2.3355 s
$\mathfrak{ts}_{3,5}$	216	15	13.9049 s	15.0469 s
$\mathfrak{ts}_{4,3}$	256	12	10.2716 s	12.5550 s
$\mathfrak{ts}_{4,4}$	625	16	172.0079 s	181.2883 s
$\mathfrak{ts}_{4,5}$	1296	20	1720.4560 s	1755.8690 s

5.3 Computing further co-location relations from $\bumpeq^{\mathfrak{ts}}_{min}$

In [27], a general approach for computing any further valid co-location relations from $\bumpeq^{\mathfrak{ts}}_{min}$ was briefly sketched. It involves looking for the solutions of the vertex colouring problem for a graph defined as follows.

Let $G^{\mathfrak{ts}}$ be an undirected graph whose vertices are the equivalence classes of the co-location relation $\bumpeq^{\mathfrak{ts}}_{min}$, and there is an edge between vertices V and V' if there is a state q of $\mathfrak{ts}$ and two events, $e \in V$ and $f \in V'$, such that $e, f \in E_q$.

An algorithm for constructing the graph $G^{\mathfrak{ts}}$ for a given ST-system $\mathfrak{ts} = (Q, A, q_0)$ is given below as Algorithm 8.

It follows from the results presented in [27], that all other co-location relations, which also provide solutions to Problem 2, are given through the solutions of the vertex colouring problem for the graph $G^{\mathfrak{ts}}$. More precisely, for each valid colouring (i.e., one which uses different colours for vertices joined by an edge), we can join into clusters of co-located events all equivalence classes of $\bumpeq^{\mathfrak{ts}}_{min}$ labelled with the same colour.

Algorithm 8 Constructing the graph $G^{\mathfrak{ts}}$ for a given ST-system $\mathfrak{ts} = (Q, A, q_0)$

```
1: function CONSTRUCT_$G^{\mathfrak{ts}}$( $\mathfrak{ts}$)
2:     initialise $V_{G^{\mathfrak{ts}}}$, $E_{G^{\mathfrak{ts}}}$ and $E_Q$ to $\varnothing$   ▷ $V_{G^{\mathfrak{ts}}}$ and $E_{G^{\mathfrak{ts}}}$ denote vertices and edges of the graph $G^{\mathfrak{ts}}$, respectively
3:     calculate $\bumpeq^{\mathfrak{ts}}_{min}$
4:     set $V_{G^{\mathfrak{ts}}}$ to $\bumpeq^{\mathfrak{ts}}_{min}$
5:     for every $q \in Q$ do
6:         initialise $E_q$ to $\varnothing$
7:         calculate $E_q$                  ▷ $E_q$ is the union of all the steps in $allSteps_q$ (see 2.1)
8:         add $E_q$ to $E_Q$
9:     for every $V' \in V_{G^{\mathfrak{ts}}}$ do
10:        for every $V'' \in V_{G^{\mathfrak{ts}}}$ do                    ▷ where $V' \neq V''$
11:            for every $E_q \in E_Q$ do
12:                if $e \in E_q$ and $f \in E_q$ then          ▷ where $e \in V'$ and $f \in V''$
13:                    initialise $V_E$ to empty list
14:                    add $V'$ and $V''$ to $V_E$        ▷ no need to add $V''$ and $V'$ to $V_E$
15:                    add $V_E$ to $E_{G^{\mathfrak{ts}}}$
16:    initialise $G^{\mathfrak{ts}}$ to graph
17:    $G^{\mathfrak{ts}}$ = build an undirected graph by using $V_{G^{\mathfrak{ts}}}$ and $E_{G^{\mathfrak{ts}}}$
18:    return $G^{\mathfrak{ts}}$
```

An algorithm for finding all possible valid co-location relations for a given $\mathfrak{ts}$ (out of $\bumpeq^{\mathfrak{ts}}_{min}$) would be both computationally expensive and of no practical value. Therefore, we focused on developing algorithms for finding further valid co-location relations, which satisfy some particular additional criteria.

An algorithm for computing valid co-location relations from $\bumpeq^{\mathfrak{ts}}_{min}$ with the smallest number of equivalence classes is given below as Algorithm 9. It takes the graph $G^{\mathfrak{ts}}$ as its parameter. The number of vertices of $G^{\mathfrak{ts}}$, let's call it n, is the number of equivalence classes of $\bumpeq^{\mathfrak{ts}}_{min}$, and it is the greatest number of equivalence classes we can have for any valid co-location relation for $\mathfrak{ts}$, as $\bumpeq^{\mathfrak{ts}}_{min}$ is the finest co-location relation for $\mathfrak{ts}$. Instead of checking the possibility of having solutions with $n-1$ colours, $n-2$ colours, etc., in a linear way, we adopted the idea used in Binary Search algorithm (at every iteration we examine either the candidate numbers from the upper or lower half of the previously considered interval), reducing the number of iterations in Algorithm 9 from n to $log_2 n$.

The results given by Algorithm 9 for our selected five examples are given below. The valid co-location relations with the smallest number of equivalent classes computed for them from $\bumpeq^{\mathfrak{ts}}_{min}$ are given as sets of their equivalence classes.

- For $\mathfrak{ts}_1$ in Figure 3.5, on p. 31, the result is $\{\{e, f, g, h\}\}$, which is the same as $\bumpeq^{\mathfrak{ts}_1}_{min}$.

Algorithm 9 Computing co-location relations from $\simeq^{\mathfrak{ts}}_{min}$ with the minimum number of equivalence classes/colours (m) and returning them as a set CR_m.

```
1: function COMPUTE_CO-LOCATION_RELATIONS_FROM_$\simeq^{\mathfrak{ts}}_{min}(G^{\mathfrak{ts}})$
2:     initialise $CR_m$ to $\varnothing$          ▷ m denotes the minimum number of colours
3:     initialise m to n          ▷ n is the number of vertices of $G^{\mathfrak{ts}}$
4:     initialise $upper$ to n-1
5:     initialise $lower$ to 0
6:     initialise $mid$ to 0
7:     while $lower <= upper$ do
8:         $mid = (lower + upper)/2$
9:         if there is a valid solution with mid number of colours then
10:            $m = mid$          ▷ $mid$ becomes the new minimum
11:            $upper = mid - 1$          ▷ can we improve the result further?
12:        else
13:            $lower = mid + 1$   ▷ we need to consider only the numbers greater than $mid$
14:    $CR_m$= compute all valid co-location relations from $\simeq^{\mathfrak{ts}}_{min}$ with m colours
15:    return $CR_m$
```

- For $\mathfrak{ts}_2$ in Figure 3.3, on p. 22, the result is $\{\{e, f, g, h\}\}$.
- For $\mathfrak{ts}_3$ in Figure 3.1, on p. 20, the result is $\{\{p_1, p_2\}, \{c_1, c_2, c_3, c_4\}\}$.
- For $\mathfrak{ts}_4$ in Figure 3.6, on p. 31, the result is $\{\{p_1, p_2\}, \{c_1, c_2, c_3, c_4\}\}$.
- For $\mathfrak{ts}_5$ generated by an ENL-system in Figure 3.7, on p. 32, the result is $\{\{a, b, c\}, \{d, e, f\}, \{g, h, i\}\}$.

We observe that while the co-location relation $\simeq^{\mathfrak{ts}_1}_{min}$ has only one equivalence class, the canonical co-location relations for $\mathfrak{ts}_2, \mathfrak{ts}_3$, $\mathfrak{ts}_4$ and $\mathfrak{ts}_5$ have many equivalence classes. However, for all the latter ST-systems some of the computed equivalence classes can be combined producing valid co-location relations with smaller number of equivalence classes as the results of applying Algorithm 9 to them show. This might not always be possible as shown in the screenshot from WORKCRAFT in Figure 5.6 giving the results of applying Algorithm 9 to the ST-system $\mathfrak{ts}_8$ generated by an ENL/LC-system in Figure 5.5.

Notice also that for all our running examples ($\mathfrak{ts}_1$–$\mathfrak{ts}_5$) we obtained only one co-location relation with the smallest number of equivalence classes. However, in general, Algorithm 9 can return more than one co-location relation satisfying this criterion for a given $\mathfrak{ts}$. The computed result might not be unique as shown in Figure 5.7 that presents the screenshot from WORKCRAFT with the output produced by Algorithm 9 for the ST-system $\mathfrak{ts}_9$ generated by an ENL/LC-system in Figure 4.10, on p. 46.

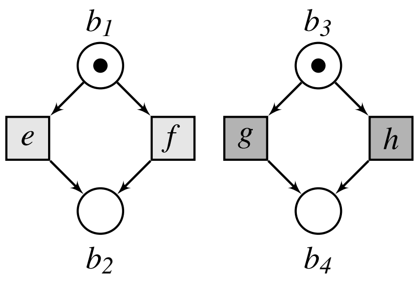
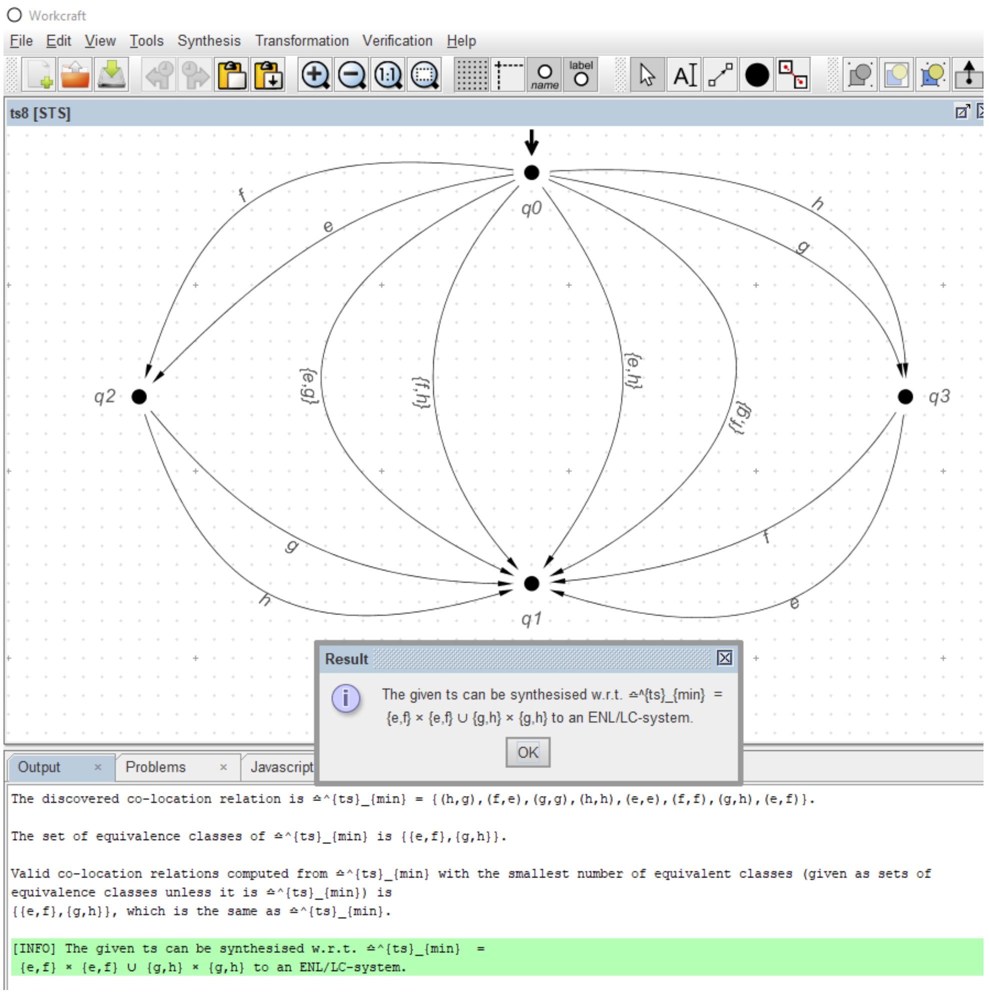


Fig. 5.5 An ENL/LC-system, where e and f are co-located events, and g and h are co-located events; and it has localised conflicts.

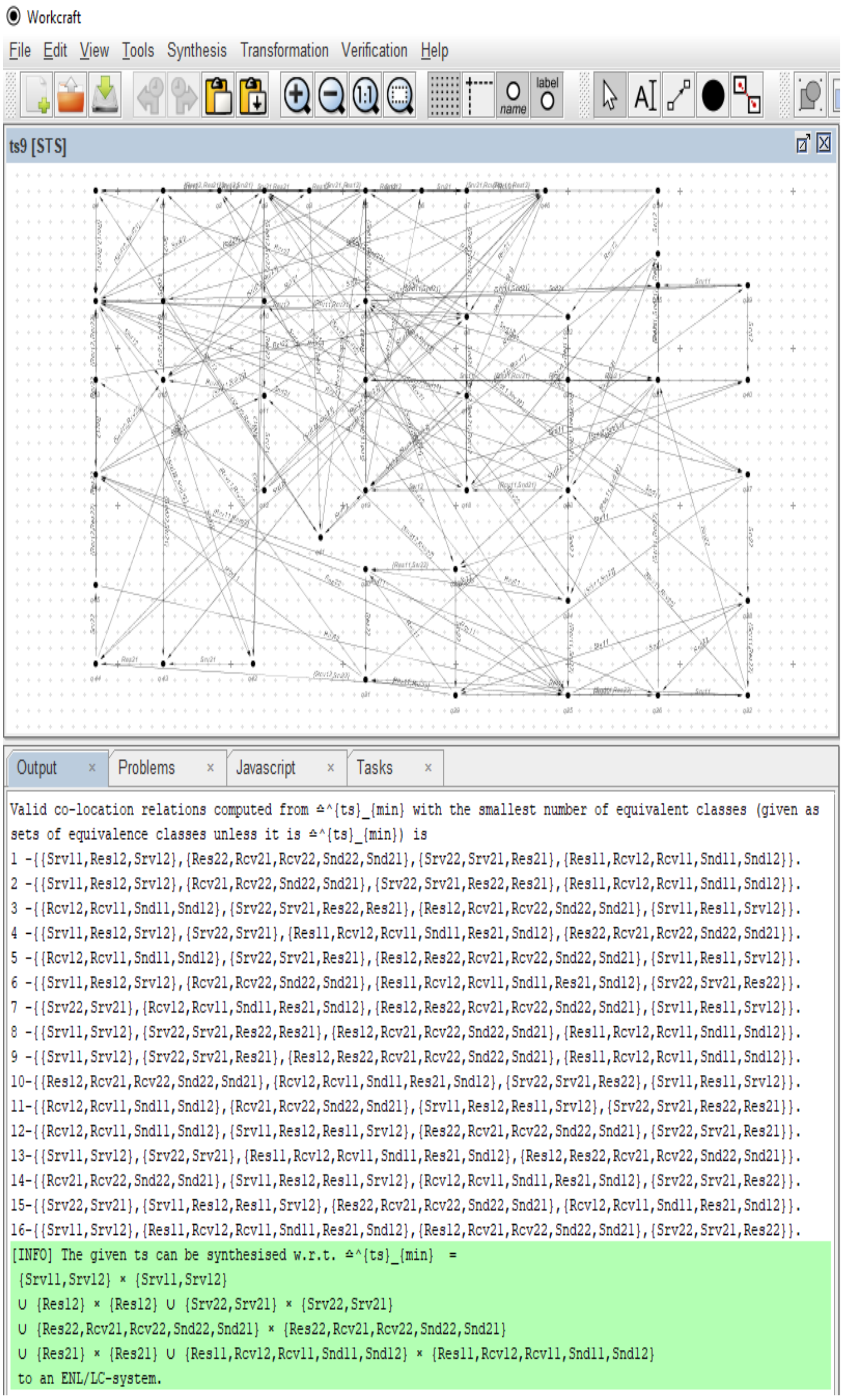


The given ts can be synthesised w.r.t. ≙^{ts}_{min} =
{e,f} × {e,f} ∪ {g,h} × {g,h} to an ENL/LC-system.

```
The discovered co-location relation is ≙^{ts}_{min} = {(h,g),(f,e),(g,g),(h,h),(e,e),(f,f),(g,h),(e,f)}.

The set of equivalence classes of ≙^{ts}_{min} is {{e,f},{g,h}}.

Valid co-location relations computed from ≙^{ts}_{min} with the smallest number of equivalent classes (given as sets of
equivalence classes unless it is ≙^{ts}_{min}) is
{{e,f},{g,h}}, which is the same as ≙^{ts}_{min}.

[INFO] The given ts can be synthesised w.r.t. ≙^{ts}_{min}  =
 {e,f} × {e,f} ∪ {g,h} × {g,h} to an ENL/LC-system.
```

Fig. 5.6 Shows the screenshot from WORKCRAFT giving the results of applying Algorithm 9 to the ST-system $\mathfrak{ts}_8$ generated by an ENL/LC-system in Figure 5.5.

Figure 5.8 shows the execution time taken to compute valid co-location relations with the smallest number of equivalence classes (from $\simeq^{\mathfrak{ts}}_{min}$) for the ST-systems: $\mathfrak{ts}_1$, $\mathfrak{ts}_2$, $\mathfrak{ts}_3$, $\mathfrak{ts}_4$ and $\mathfrak{ts}_5$, when using Algorithm 9.


Workcraft
File Edit View Tools Synthesis Transformation Verification Help
ts9 [STS]
Output
Problems
Javascript
Tasks

```
Valid co-location relations computed from ≙^{ts}_{min} with the smallest number of equivalent classes (given as
sets of equivalence classes unless it is ≙^{ts}_{min}) is
1 -{{Srv11,Res12,Srv12},{Res22,Rcv21,Rcv22,Snd22,Snd21},{Srv22,Srv21,Res21},{Res11,Rcv12,Rcv11,Snd11,Snd12}}.
2 -{{Srv11,Res12,Srv12},{Rcv21,Rcv22,Snd22,Snd21},{Srv22,Srv21,Res22,Res21},{Res11,Rcv12,Rcv11,Snd11,Snd12}}.
3 -{{Rcv12,Rcv11,Snd11,Snd12},{Srv22,Srv21,Res22,Res21},{Res12,Rcv21,Rcv22,Snd22,Snd21},{Srv11,Res11,Srv12}}.
4 -{{Srv11,Res12,Srv12},{Srv22,Srv21},{Res11,Rcv12,Rcv11,Snd11,Res21,Snd12},{Res22,Rcv21,Rcv22,Snd22,Snd21}}.
5 -{{Rcv12,Rcv11,Snd11,Snd12},{Srv22,Srv21,Res21},{Res12,Res22,Rcv21,Rcv22,Snd22,Snd21},{Srv11,Res11,Srv12}}.
6 -{{Srv11,Res12,Srv12},{Rcv21,Rcv22,Snd22,Snd21},{Res11,Rcv12,Rcv11,Snd11,Res21,Snd12},{Srv22,Srv21,Res22}}.
7 -{{Srv22,Srv21},{Rcv12,Rcv11,Snd11,Res21,Snd12},{Res12,Res22,Rcv21,Rcv22,Snd22,Snd21},{Srv11,Res11,Srv12}}.
8 -{{Srv11,Srv12},{Srv22,Srv21,Res22,Res21},{Res12,Rcv21,Rcv22,Snd22,Snd21},{Res11,Rcv12,Rcv11,Snd11,Snd12}}.
9 -{{Srv11,Srv12},{Srv22,Srv21,Res21},{Res12,Res22,Rcv21,Rcv22,Snd22,Snd21},{Res11,Rcv12,Rcv11,Snd11,Snd12}}.
10-{{Res12,Rcv21,Rcv22,Snd22,Snd21},{Rcv12,Rcv11,Snd11,Res21,Snd12},{Srv22,Srv21,Res22},{Srv11,Res11,Srv12}}.
11-{{Rcv12,Rcv11,Snd11,Snd12},{Rcv21,Rcv22,Snd22,Snd21},{Srv11,Res12,Res11,Srv12},{Srv22,Srv21,Res22,Res21}}.
12-{{Rcv12,Rcv11,Snd11,Snd12},{Srv11,Res12,Res11,Srv12},{Res22,Rcv21,Rcv22,Snd22,Snd21},{Srv22,Srv21,Res21}}.
13-{{Srv11,Srv12},{Srv22,Srv21},{Res11,Rcv12,Rcv11,Snd11,Res21,Snd12},{Res12,Res22,Rcv21,Rcv22,Snd22,Snd21}}.
14-{{Rcv21,Rcv22,Snd22,Snd21},{Srv11,Res12,Res11,Srv12},{Rcv12,Rcv11,Snd11,Res21,Snd12},{Srv22,Srv21,Res22}}.
15-{{Srv22,Srv21},{Srv11,Res12,Res11,Srv12},{Res22,Rcv21,Rcv22,Snd22,Snd21},{Rcv12,Rcv11,Snd11,Res21,Snd12}}.
16-{{Srv11,Srv12},{Res11,Rcv12,Rcv11,Snd11,Res21,Snd12},{Res12,Rcv21,Rcv22,Snd22,Snd21},{Srv22,Srv21,Res22}}.
[INFO] The given ts can be synthesised w.r.t. ≙^{ts}_{min}  =
 {Srv11,Srv12} × {Srv11,Srv12}
 ∪ {Res12} × {Res12} ∪ {Srv22,Srv21} × {Srv22,Srv21}
 ∪ {Res22,Rcv21,Rcv22,Snd22,Snd21} × {Res22,Rcv21,Rcv22,Snd22,Snd21}
 ∪ {Res21} × {Res21} ∪ {Res11,Rcv12,Rcv11,Snd11,Snd12} × {Res11,Rcv12,Rcv11,Snd11,Snd12}
 to an ENL/LC-system.
```

Fig. 5.7 Shows the screenshot from WORKCRAFT giving the results of applying Algorithm 9 to the ST-system $\mathfrak{ts}_9$ generated by an ENL/LC-system in Figure 4.10.

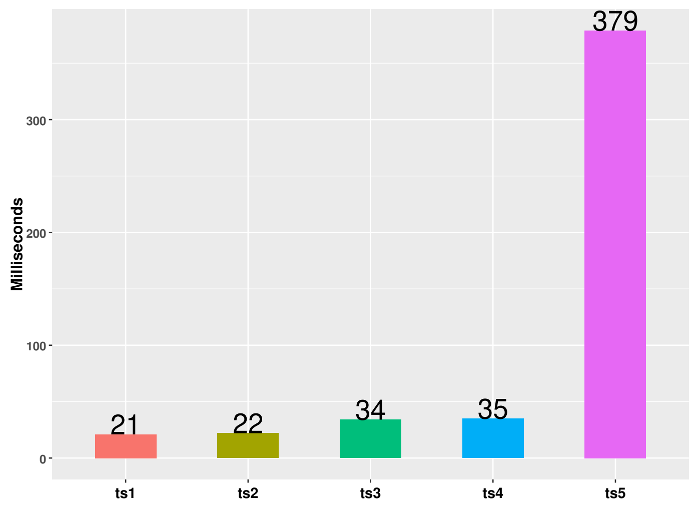


Fig. 5.8 A diagram showing the execution time taken to compute valid co-location relations with the smallest number of equivalence classes (from $\simeq^{\mathfrak{ts}}_{min}$) for the ST-systems: $\mathfrak{ts}_1$, $\mathfrak{ts}_2$, $\mathfrak{ts}_3$, $\mathfrak{ts}_4$ and $\mathfrak{ts}_5$, when using Algorithm 9.

Moreover, when we applied Algorithm 9 to $\mathfrak{ts}_9$ about 2.4839s needed to compute valid co-location relations with the smallest number of equivalence classes (from $\simeq^{\mathfrak{ts}}_{min}$).

The next algorithm, Algorithm 10, computes valid co-location relations from $\simeq^{\mathfrak{ts}}_{min}$ with the smallest number of equivalence classes and the most balanced distribution of events in their equivalence classes (the criteria are considered in this order). We understand that a co-location relation, in a set of co-location relations, has the most balanced distribution of events between equivalence classes, when the difference between the sizes of its biggest and smallest equivalence classes (called the *balanced indicator* of a co-location relation) is the smallest in the considered set of co-location relations.

The input for this algorithm is the set of co-location relations returned by Algorithm 9, CR_m, containing the co-location relations for $\mathfrak{ts}$ (computed from $\simeq^{\mathfrak{ts}}_{min}$) with the smallest number of equivalence classes (m denotes this number).

The results given by Algorithm 10 for our running examples are given below. The results for these examples are predictable as the sets obtained from Algorithm 9 for them, and used as inputs for Algorithm 10, are one-element sets. So, this one element in a set satisfies trivially the second criterion. The valid co-location relations with the smallest number of equivalent classes and the most balanced distribution of events in their equivalence classes computed for them from $\simeq^{\mathfrak{ts}}_{min}$ are given as sets of their equivalence classes.

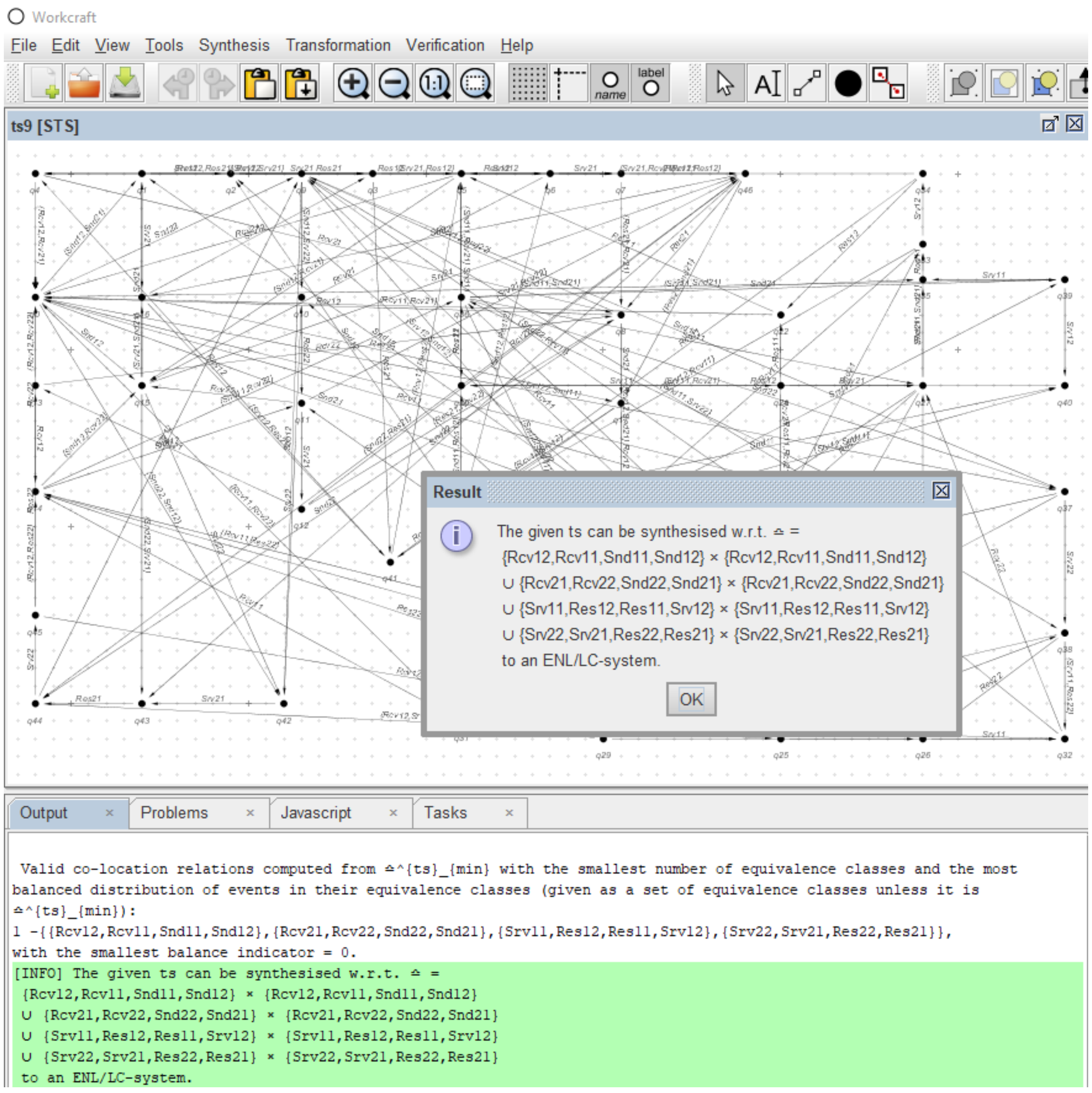
- For $\mathfrak{ts}_1$ in Figure 3.5, on p. 31, the result is $\{\{e,f,g,h\}\}$, with balanced indicator = 0.

Algorithm 10 Select co-location relations from CR_m with the smallest balance indicators.

```
1: function SELECT_BALANCED_RELATIONS($CR_m$)
2:     initialise $CR_m^{bI}$ to empty map    ▷ mapping co-location relations to their balance indicators
3:     for every co-location relation $cr_m \in CR_m$ do
4:         initialise bSize, sSize and bI to 0
5:         bSize = get the size of the biggest equivalence class of $cr_m$
6:         sSize = get the size of the smallest equivalence class of $cr_m$
7:         $bI = bSize - sSize$
8:         add the pair $(cr_m, bI)$ to the $CR_m^{bI}$ map
9:     initialise $CR_m^b$ to $\varnothing$ ▷ set of co-location relations with the smallest balance indicators
10:    $CR_m^b$ = get co-location relations with the smallest bI from $CR_m^{bI}$
11:    return $CR_m^b$
```

- For $\mathfrak{ts}_2$ in Figure 3.3, on p. 22, the result is $\{\{e,f,g,h\}\}$, with balanced indicator = 0.
- For $\mathfrak{ts}_3$ in Figure 3.1, on p. 20, the result is $\{\{p_1,p_2\},\{c_1,c_2,c_3,c_4\}\}$, with balanced indicator = 2.
- For $\mathfrak{ts}_4$ in Figure 3.6, on p. 31, the result is $\{\{p_1,p_2\},\{c_1,c_2,c_3,c_4\}\}$, with balanced indicator = 2.
- For $\mathfrak{ts}_5$ generated by an ENL-system in Figure 3.7, on p. 32, the result is $\{\{a,b,c\},\{d,e,f\},\{g,h,i\}\}$, with balanced indicator = 0.

The results of running Algorithm 10 for the ST-system generated by the ENL-system in Figure 4.10, on p. 46, are given in Figure 5.9 as they are presented in WORKCRAFT. Although there are 16 co-location relations for this example with 4 equivalence classes (see Figure 5.7), only one of them has the balanced indicator equal to zero, giving the split for event localities as in Figure 4.10.


Workcraft
File Edit View Tools Synthesis Transformation Verification Help
ts9 [STS]
Result
OK
Output
Problems
Javascript
Tasks

The given ts can be synthesised w.r.t. ≏ =
{Rcv12,Rcv11,Snd11,Snd12} × {Rcv12,Rcv11,Snd11,Snd12}
∪ {Rcv21,Rcv22,Snd22,Snd21} × {Rcv21,Rcv22,Snd22,Snd21}
∪ {Srv11,Res12,Res11,Srv12} × {Srv11,Res12,Res11,Srv12}
∪ {Srv22,Srv21,Res22,Res21} × {Srv22,Srv21,Res22,Res21}
to an ENL/LC-system.

```
 Valid co-location relations computed from ≏^{ts}_{min} with the smallest number of equivalence classes and the most
balanced distribution of events in their equivalence classes (given as a set of equivalence classes unless it is
≏^{ts}_{min}):
1 -{{Rcv12,Rcv11,Snd11,Snd12},{Rcv21,Rcv22,Snd22,Snd21},{Srv11,Res12,Res11,Srv12},{Srv22,Srv21,Res22,Res21}},
with the smallest balance indicator = 0.
[INFO] The given ts can be synthesised w.r.t. ≏ =
 {Rcv12,Rcv11,Snd11,Snd12} × {Rcv12,Rcv11,Snd11,Snd12}
 ∪ {Rcv21,Rcv22,Snd22,Snd21} × {Rcv21,Rcv22,Snd22,Snd21}
 ∪ {Srv11,Res12,Res11,Srv12} × {Srv11,Res12,Res11,Srv12}
 ∪ {Srv22,Srv21,Res22,Res21} × {Srv22,Srv21,Res22,Res21}
 to an ENL/LC-system.
```

Fig. 5.9 Shows the screenshot from WORKCRAFT with the results of applying Algorithm 10 to the ST-system $\mathfrak{ts}_9$ generated by an ENL/LC-system in Figure 4.10.

Chapter 6

Minimisation of the synthesised nets

Suppose now that we have applied the synthesis procedure on an ENLST-system as described in Chapter 4. The resulting $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ is a saturated net because it uses all the non-trivial regions as conditions. Many of these conditions might be unnecessarily and redundant. Removing such conditions would not change the net behaviour [13, 20, 28, 36]. So, in this chapter, we will address the minimisation of solutions of the synthesised nets.

6.1 Optimising solutions to the synthesis problem

In this section we recall some notions and results from [28].

The ENL-system $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ obtained from the synthesis of the ENLST-system $\mathfrak{ts}$ may 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 that generates the ST-system, which is still isomorphic to $\mathfrak{ts}$.

Suppose that we have reduced $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ in this way obtaining a sub-ENL-system $\mathfrak{enl}$. We would like to reduce $\mathfrak{enl}$ further by deleting a condition/region $\mathfrak{r}$ (and its adjacent arcs) without, as before, violating the property of it being an ENL-system [3] and making sure that the resultant net still generates the ST-system isomorphic to $\mathfrak{ts}$. We will denote the ENL-system after such one step reduction: $\mathfrak{enl}_{\mathfrak{r}}$.

In [28], it was proved that complement regions are very often redundant:

Reduction Rule 1. *If $\mathfrak{r} = (in, r, out)$ and $\bar{\mathfrak{r}} = (out, Q \setminus r, in)$ are two conditions in $\mathfrak{enl}$ and deleting $\bar{\mathfrak{r}}$ leads to an* ENL*-system, then the* ST*-systems generated by $\mathfrak{enl}$ and $\mathfrak{enl}_{\bar{\mathfrak{r}}}$ are isomorphic and $\bar{\mathfrak{r}}$ is redundant.* $\diamondsuit$

[3] Every ENL-system $\mathfrak{enl} = (B, E, F, \hat{=}, c_0)$ should satisfy the following conditions: $\forall e \in E\ ({}^{\bullet}e \neq \varnothing \wedge e^{\bullet} \neq \varnothing \wedge {}^{\bullet}e \cap e^{\bullet} = \varnothing)$.

Another source of redundancy among conditions/regions in the synthesised net are "big" regions that are compositions of smaller regions. To define a composition operator, [28] introduces the concept of *compatible* regions.

Definition 6.1.1. *A region (in, r, out) is compatible with another region (in', r', out') iff the following three conditions hold:*

1. *$r \cap r' = \varnothing$.*
2. *For every $e \in out$ exactly one of the following holds:*
   - *For all the transitions (q, U, q') such that $e \in U$ we have $q' \in r'$.*
   - *For all the transitions (q, U, q') such that $e \in U$ we have $q' \notin r'$.*
3. *For every $e \in in$ exactly one of the following holds:*
   - *For all the transitions (q, U, q') such that $e \in U$ we have $q \in r'$.*
   - *For all the transitions (q, U, q') such that $e \in U$ we have $q \notin r'$.*

If region $\mathfrak{r}$ is compatible with region $\mathfrak{r}'$ and region $\mathfrak{r}'$ is compatible with $\mathfrak{r}$ we say that the two regions are compatible. ♢

In [28], it was proved that the *composition of two compatible regions* (defined below) is also a region. If $\mathfrak{r} = (in, r, out)$ and $\mathfrak{r}' = (in', r', out')$ are two non-trivial compatible regions of an ENLST-system $\mathfrak{ts}$, then the following is a (possibly trivial) region of $\mathfrak{ts}$:

$$\mathfrak{r} \oplus \mathfrak{r}' \overset{\mathrm{df}}{=} (in \cup in' \setminus H, r \cup r', out \cup out' \setminus H),$$

where H is a set of events that belong <u>only</u> to steps labelling transitions hidden/buried in $r \cup r'$ (with its source in r and its target in r' or the other way round). The region $\mathfrak{r} \oplus \mathfrak{r}'$ is called the composition of $\mathfrak{r}$ and $\mathfrak{r}'$. In [28], the following reduction rule was proved:

Reduction Rule 2. *If $\mathfrak{r} = (in, r, out)$, $\mathfrak{r}' = (in', r', out')$ and $\mathfrak{r} \oplus \mathfrak{r}'$ are three conditions/regions in $\mathfrak{enl}$, then the* ST*-systems generated by $\mathfrak{enl}$ and $\mathfrak{enl}_{\mathfrak{r} \oplus \mathfrak{r}'}$ are isomorphic and $\mathfrak{r} \oplus \mathfrak{r}'$ is redundant.* ♢

The third reduction rule considers regions of $\mathfrak{ts} = (Q, A, q_0)$ based on the same set of states. We will call such regions *companion* regions. For a given set of states r, they will belong to the set denoted by $\mathfrak{R}^r_{\mathfrak{ts}}$.

In [28], it was proved that if the events contained in the set *in* (*out*) of a region $\mathfrak{r} = (in, r, out)$ can be found in the *in* (*out*) sets of other companion regions then $\mathfrak{r}$ is redundant and can be deleted. Formally:

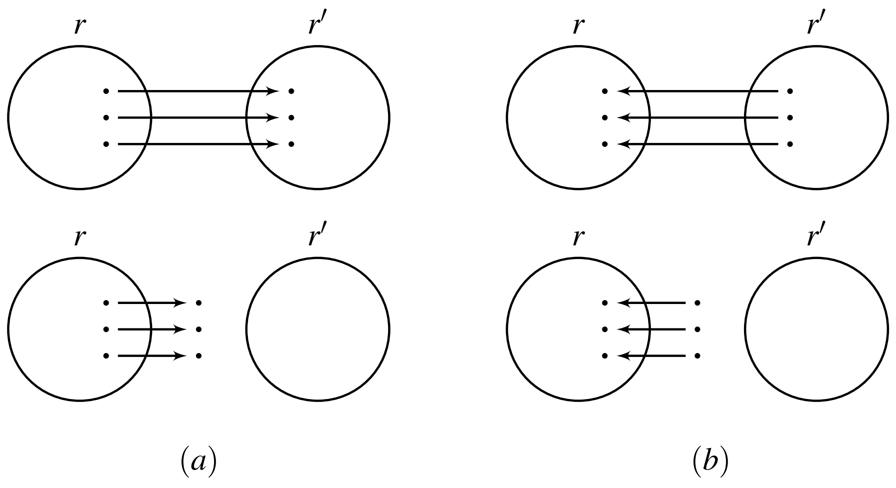


Fig. 6.1 An illustration for the definition of region $\mathfrak{r} = (in, r, out)$ being compatible with region $\mathfrak{r}' = (in', r', out')$, showing the two possibilities for point 2 of the definition (a); and the two possibilities for point 3 of the definition (b).

Reduction Rule 3. *Let* $\mathfrak{r} = (in, r, out)$ *be a condition/region of* $\mathfrak{enl}$ *such that:*

$$in \subseteq \bigcup\{in' \mid (in', r, out') \textit{ is condition in } \mathfrak{enl} \textit{ different from } \mathfrak{r}\} \tag{6.1}$$

$$out \subseteq \bigcup\{out' \mid (in', r, out') \textit{ is condition in } \mathfrak{enl} \textit{ different from } \mathfrak{r}\} \tag{6.2}$$

Then the ST*-systems generated by* $\mathfrak{enl}$ *and* $\mathfrak{enl}_{\mathfrak{r}}$ *are isomorphic and* $\mathfrak{r}$ *is redundant.*

◊

6.2 Minimal regions

For many classes of Petri nets, for which the synthesis problem was investigated, a region was defined as a subset of states of a transition system. For such classes of nets and their transition systems a minimal region was defined *w.r.t.* the set inclusion $\subset$ [13, 12, 17, 36]. Also, composition of regions (as sets) was defined by using the set union operator ($\cup$), which is both commutative and associative.

The regions of ENLST-systems are triples of the form: $\mathfrak{r} = (in, r, out)$. The minimal regions in this class of (step) transition systems are defined *w.r.t.* the strict pre-order $\prec$ on the set of regions, that utilises the idea of regions' composition by means of $\oplus$:

$$\mathfrak{r} \prec \mathfrak{r}' \text{ iff there is a non-trivial region } \mathfrak{r}'' \text{ such that } \mathfrak{r} \oplus \mathfrak{r}'' = \mathfrak{r}' \text{ [28].}$$

Formally, we have the following definition of a *minimal region*:

Definition 6.2.1. *A region* $\mathfrak{r} \in \mathfrak{R}_{\mathfrak{ts}}$ *is* minimal *iff* $\forall \widehat{\mathfrak{r}} \in \mathfrak{R}_{\mathfrak{ts}} : \widehat{\mathfrak{r}} \not\prec \mathfrak{r}$. ◇

The set of minimal regions of $\mathfrak{ts}$ w.r.t. $\prec$ will be denoted by $\mathfrak{R}^{min}_{\mathfrak{ts}}$.

We observe that if a non-trivial region is *non-minimal* then it can be represented as a composition of two other non-trivial regions. This follows from the definition of the relation $\prec$ and the fact that the composition operator $\oplus$ is commutative, which, in turn, follows immediately from the definition of $\oplus$.

As an example, consider the ENLST-system in Figure 6.2(a). Its non-trivial regions are:

$$\begin{array}{lll} \mathfrak{r}_1 & = & (\varnothing, \{q_0\}, \{e\}) \\ \mathfrak{r}_2 & = & (\varnothing, \{q_0\}, \{e_1, e_2\}) \\ \mathfrak{r}_5 & = & (\{e_1\}, \{q_1\}, \varnothing) \\ \mathfrak{r}_6 & = & (\{e_2\}, \{q_2\}, \varnothing) \end{array} \qquad \begin{array}{lll} \mathfrak{r}_3 & = & \bar{\mathfrak{r}}_1 = (\{e\}, \{q_1, q_2\}, \varnothing) \\ \mathfrak{r}_4 & = & \bar{\mathfrak{r}}_2 = (\{e_1, e_2\}, \{q_1, q_2\}, \varnothing) \\ \mathfrak{r}_7 & = & \bar{\mathfrak{r}}_5 = (\varnothing, \{q_0, q_2\}, \{e_1\}) \\ \mathfrak{r}_8 & = & \bar{\mathfrak{r}}_6 = (\varnothing, \{q_0, q_1\}, \{e_2\}) \end{array}$$

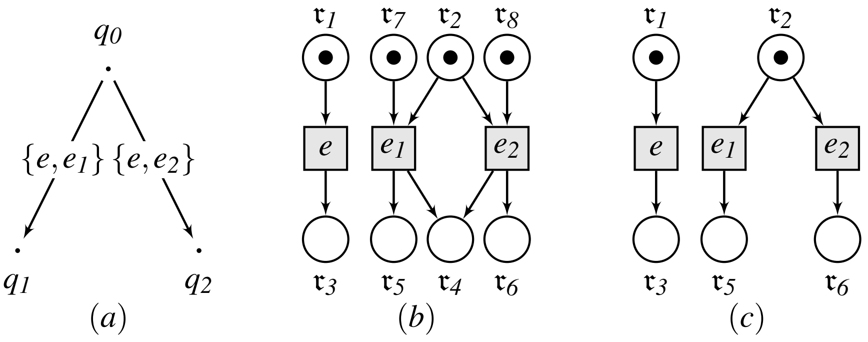


Fig. 6.2 An ENLST-system with three co-located events e, e_1 and e_2 (a); the ENL-system resulting from its synthesis (b); and the reduced ENL-system solution for (a) that uses only regions minimal *w.r.t.* $\prec$ (c).

The minimal regions of the ENLST-system in Figure 6.2(a) are: $\mathfrak{r}_1$, $\mathfrak{r}_2$, $\mathfrak{r}_3$, $\mathfrak{r}_5$ and $\mathfrak{r}_6$. The remaining regions are non-minimal (their set is denoted by $\mathfrak{R}^{\oplus}_{\mathfrak{ts}}$):

$$\begin{array}{lllr} \mathfrak{r}_4 & = & \mathfrak{r}_5 \oplus \mathfrak{r}_6 & (H = \varnothing) \\ \mathfrak{r}_7 & = & \mathfrak{r}_2 \oplus \mathfrak{r}_6 & (H = \{e_2\}) \\ \mathfrak{r}_8 & = & \mathfrak{r}_2 \oplus \mathfrak{r}_5 & (H = \{e_1\}) \end{array}$$

The reduced ENL-system solution for the ENLST-system in Figure 6.2(a) that uses only regions minimal *w.r.t.* $\prec$ is shown in Figure 6.2(c).

Note that the operator $\oplus$ is not associative as can be shown by using, again, the example of the ENLST-system in Figure 6.2(a). We can observe that: $(\mathfrak{r}_5 \oplus \mathfrak{r}_6) \oplus \mathfrak{r}_1 \neq \mathfrak{r}_5 \oplus (\mathfrak{r}_6 \oplus \mathfrak{r}_1)$.

While $\mathfrak{r}_5$ and $\mathfrak{r}_6$ are compatible and their composition produces $\mathfrak{r}_4$ ($\mathfrak{r}_4 = \mathfrak{r}_5 \oplus \mathfrak{r}_6$), regions $\mathfrak{r}_6$ and $\mathfrak{r}_1$ are not compatible, because $\mathfrak{r}_1$ is not compatible with $\mathfrak{r}_6$, and they cannot be composed.

Also notice that there might be two companion regions (regions based on the same set of states) such that one of them is a minimal region and the second one is a non-minimal region. See, for example, regions $\mathfrak{r}_3 = (\{e\},\{q_1,q_2\},\varnothing)$ and $\mathfrak{r}_4 = (\{e_1,e_2\},\{q_1,q_2\},\varnothing)$ of the ENLST-system in Figure 6.2(a), where $\mathfrak{r}_3$ is minimal and $\mathfrak{r}_4$ is non-minimal. So, the minimality of a region cannot be decided by looking at its set of states only.

The next result, about the representation of non-trivial regions, is similar to the results proved for other classes of nets (and their transition systems) that can be found in the literature: Elementary Net Systems [13], pure and bounded Place/Transition Nets [12], Safe Nets [17] or Elementary Net Systems with Inhibitor Arcs (ENI-systems) [36].

Theorem 6.2.2. *Every* $\mathfrak{r} = (in, r, out) \in \mathfrak{R}_{\mathfrak{ts}}$ *can be represented as a composition of minimal regions, where for each pair of different minimal regions in this representation,* $\widehat{\mathfrak{r}} = (\widehat{in}, \widehat{r}, \widehat{out})$ *and* $\widetilde{\mathfrak{r}} = (\widetilde{in}, \widetilde{r}, \widetilde{out})$*,* $\widehat{r} \cap \widetilde{r} = \varnothing$.

Proof. If $\mathfrak{r}$ is a minimal region then the result holds. If $\mathfrak{r}$ is non-minimal then there exists a minimal region $\mathfrak{r}' = (in', r', out') \prec \mathfrak{r}$. From the definition of the strict pre-order $\prec$ we have that there exists $\mathfrak{r}'' = (in'', r'', out'')$ such that $\mathfrak{r}' \oplus \mathfrak{r}'' = \mathfrak{r}$. If $\mathfrak{r}''$ is minimal we have $\mathfrak{r} = \mathfrak{r}' \oplus \mathfrak{r}''$. If $\mathfrak{r}''$ is non-minimal we continue in the same way with $\mathfrak{r}''$ as we have done before with $\mathfrak{r}$. In this way we will build a sequence of minimal regions, whose sets of states are mutually disjoint and their sum would define the set of states for $\mathfrak{r}$. As Q is finite, the number of minimal regions in the representation of $\mathfrak{r}$ will be finite:

$$\mathfrak{r} = \mathfrak{r}_1 \oplus (\mathfrak{r}_2 \oplus \ldots (\mathfrak{r}_{n-2} \oplus (\mathfrak{r}_{n-1} \oplus \mathfrak{r}_n)) \ldots),$$

where $\mathfrak{r}_i = (in_i, r_i, out_i)$ (for $i = 1, \ldots, n$) are minimal regions.

Observe that by the definition of compatibility of regions and that of $\oplus$ operator, we have $r_{n-1} \cap r_n = \varnothing$ and $r_{n-2} \cap (r_{n-1} \cup r_n) = \varnothing$. We can proceed in this way from right to left of the above representation, ending with $r_1 \cap r'' = \varnothing$, where r_1 is the original r' and $r'' = r_2 \cup r_3 \cup \ldots \cup r_{n-1} \cup r_n$. So, all the sets of states of the minimal regions in the representation are pairwise disjoint. $\square$

Theorem 6.2.2 and Reduction Rule 2 imply that one can construct a solution to the synthesis problem based on minimal regions *w.r.t.* the strict pre-order $\prec$. The consequence of the fact that the operator $\oplus$ is not associative is that we cannot drop the brackets, if there are any, when we represent a non-trivial region of an ENLST-system as a composition of minimal regions (see the proof of Theorem 6.2.2).

6.3 Properties of regions

In this section we gather facts regarding relationships of complementary, compatible, companion and minimal regions of an ENLST-system $\mathfrak{ts} = (Q, A, q_0)$.

Fact 4. *Any pair of complementary regions of* $\mathfrak{ts}$, $\mathfrak{r}$ *and* $\bar{\mathfrak{r}}$, *form a pair of compatible regions and* $\mathfrak{r} \oplus \bar{\mathfrak{r}} = (\varnothing, Q, \varnothing)$. $\diamondsuit$

From Fact 4 it follows that if $\mathfrak{ts}$ has only minimal regions among non-trivial regions, then only the pairs of complementary regions can be composed resulting in the trivial region $(\varnothing, Q, \varnothing)$.

Fact 5. *Let* $\mathfrak{r}_1 = (in_1, r_1, out_1)$ *and* $\mathfrak{r}_2 = (in_2, r_2, out_2)$ *be two non-trivial compatible regions of an* ENLST*-system* $\mathfrak{ts}$. *Then* $in_1 \cap in_2 = \varnothing$ *and* $out_1 \cap out_2 = \varnothing$.

Proof. Since $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are compatible regions, we have $r_1 \cap r_2 = \varnothing$. Suppose $in_1 \cap in_2 \neq \varnothing$. Then there exists $e \in in_1 \cap in_2$ and every transition labelled with a step containing e enters both r_1 and r_2 (see Definition 2.1.6(**R4**)), but that is impossible as $r_1 \cap r_2 = \varnothing$ - a contradiction. So, $in_1 \cap in_2 = \varnothing$. Similarly, we can show that $out_1 \cap out_2 = \varnothing$. $\square$

Now, we introduce a notion of *strong compatibility* of regions.

Definition 6.3.1. *A region* (in, r, out) *is strongly compatible with another region* (in', r', out') *iff the following three conditions hold:*

1. $r \cap r' = \varnothing$.
2. *For every* $e \in out$ *exactly one of the following holds:*
    - $e \in in'$.
    - *For all the transitions* (q, U, q') *such that* $e \in U$ *we have* $q' \notin r'$.
3. *For every* $e \in in$ *exactly one of the following holds:*
    - $e \in out'$.
    - *For all the transitions* (q, U, q') *such that* $e \in U$ *we have* $q \notin r'$.

If region $\mathfrak{r}$ *is strongly compatible with region* $\mathfrak{r}'$ *and region* $\mathfrak{r}'$ *is strongly compatible with* $\mathfrak{r}$ *we say that the two regions are strongly compatible.* $\diamondsuit$

Fact 6. *Two regions,* (in, r, out) *and* (in', r', out'), *that are strongly compatible are compatible.*

Proof. Follows from the facts that:

- $e \in in'$ implies: For all the transitions (q,U,q') such that $e \in U$ we have $q' \in r'$ (see Definition 2.1.6(**R4**)).
- $e \in out'$ implies: For all the transitions (q,U,q') such that $e \in U$ we have $q \in r'$ (see Definition 2.1.6(**R3**)).

□

The composition operator defined for the strongly compatible regions (rather than compatible regions) will be denoted by $\oplus_s$. The strict pre-order relation for the set of regions that utilises operator $\oplus_s$ instead of $\oplus$ will be denoted by $\prec_s$. The set of minimal regions of $\mathfrak{ts}$ w.r.t. $\prec_s$ will be denoted by $\mathfrak{R}^{min,s}_{\mathfrak{ts}}$.

The implication of Fact 6 is that Reduction Rule 2 works with strongly compatible regions (we can replace operator $\oplus$ by $\oplus_s$ in that rule). Furthermore, Definition 6.3.1 means that we can strengthen Fact 4 to:

Fact 7. *Any pair of complementary regions of* $\mathfrak{ts}$, $\mathfrak{r}$ *and* $\bar{\mathfrak{r}}$, *form a pair of strongly compatible regions and* $\mathfrak{r} \oplus_s \bar{\mathfrak{r}} = (\varnothing, Q, \varnothing)$. ◇

We will give examples of compatible and strongly compatible pairs of regions using the ENLST-system in Figure 6.2(a). We present the pairs in Table 6.1, where we use the following denotations:

- Symbol '–' means that a pair of regions $(\mathfrak{r}_i, \mathfrak{r}_j)$, where i is a row index and j is a column index, has sets of states, r_i and r_j, such that $r_i \cap r_j \neq \varnothing$, and therefore they are not compatible.
- **C** means that $(\mathfrak{r}_i, \mathfrak{r}_j)$ is a pair of compatible regions.
- **SC** means that $(\mathfrak{r}_i, \mathfrak{r}_j)$ is a pair of strongly compatible regions.
- **NC** means that a pair of regions $(\mathfrak{r}_i, \mathfrak{r}_j)$ is not compatible, but not because their sets of states have non-empty intersection.

Note the asymmetry in the table for pairs involving regions $\mathfrak{r}_1$ and $\mathfrak{r}_5$, where $\mathfrak{r}_5$ is compatible with $\mathfrak{r}_1$, but $\mathfrak{r}_1$ is not compatible with $\mathfrak{r}_5$. So, there is **C** assigned for the pair $(\mathfrak{r}_5, \mathfrak{r}_1)$ in the table, but there is **NC** assigned for the pair $(\mathfrak{r}_1, \mathfrak{r}_5)$. Similarly, we have asymmetry for the pairs involving $\mathfrak{r}_1$ and $\mathfrak{r}_6$.

The next proposition shows that unlike $\oplus$ operator, $\oplus_s$ is associative.

Proposition 6.3.2. *The operator* $\oplus_s$ *is associative.*

Table 6.1 Shows compatibility status for the pairs of regions of the ENLST-system in Figure 6.2(a).

	$\mathfrak{r}_1$	$\mathfrak{r}_2$	$\mathfrak{r}_3$	$\mathfrak{r}_4$	$\mathfrak{r}_5$	$\mathfrak{r}_6$	$\mathfrak{r}_7$	$\mathfrak{r}_8$
$\mathfrak{r}_1$	–	–	SC	C	NC	NC	–	–
$\mathfrak{r}_2$	–	–	C	SC	SC	SC	–	–
$\mathfrak{r}_3$	SC	C	–	–	–	–	–	–
$\mathfrak{r}_4$	C	SC	–	–	–	–	–	–
$\mathfrak{r}_5$	C	SC	–	–	–	SC	SC	–
$\mathfrak{r}_6$	C	SC	–	–	SC	–	–	SC
$\mathfrak{r}_7$	–	–	–	–	SC	–	–	–
$\mathfrak{r}_8$	–	–	–	–	–	SC	–	–

Proof. Let $\mathfrak{r}_1 = (in_1, r_1, out_1)$, $\mathfrak{r}_2 = (in_2, r_2, out_2)$ and $\mathfrak{r}_3 = (in_3, r_3, out_3)$ be non-trivial regions of $\mathfrak{ts} = (Q, A, q_0)$. Furthermore, we assume that $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are strongly compatible and their composition $\mathfrak{r} = \mathfrak{r}_1 \oplus_s \mathfrak{r}_2$ is a non-trivial region such that $\mathfrak{r}$ and $\mathfrak{r}_3$ are strongly compatible.

We show that $\mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible, their composition $\mathfrak{r}' = \mathfrak{r}_2 \oplus_s \mathfrak{r}_3$ is a non-trivial region such that $\mathfrak{r}'$ and $\mathfrak{r}_1$ are strongly compatible. Moreover, the following holds:

$$(\mathfrak{r}_1 \oplus_s \mathfrak{r}_2) \oplus_s \mathfrak{r}_3 = \mathfrak{r}_1 \oplus_s (\mathfrak{r}_2 \oplus_s \mathfrak{r}_3).$$

First we show that $\mathfrak{r}_2$ is strongly compatible with $\mathfrak{r}_3$. As $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are strongly compatible regions and $\mathfrak{r} = \mathfrak{r}_1 \oplus_s \mathfrak{r}_2$, we have:

$$\mathfrak{r} = (in, r, out) = (in_1 \cup in_2 \setminus H, r_1 \cup r_2, out_1 \cup out_2 \setminus H),$$

where H is a set of events that belong <u>only</u> to steps labelling transitions buried in $r_1 \cup r_2$. From assumptions we also have: $r_1 \cap r_2 = \varnothing$, $(r_1 \cup r_2) \cap r_3 = \varnothing$. So, $r_1 \cap r_3 = \varnothing$ and $r_2 \cap r_3 = \varnothing$.

Now, we show that:

$$\forall e \in out_2 : \ e \in in_3 \ \vee \ (\forall (q, U, q') \in A : \ e \in U \implies q' \notin r_3) \qquad (\dagger)$$

$$\forall e \in in_2 : \ e \in out_3 \ \vee \ (\forall (q, U, q') \in A : \ e \in U \implies q \notin r_3) \qquad (\dagger\dagger)$$

Suppose, to the contrary, that ($\dagger$) is not true. Then $\exists \widehat{e} \in out_2 : \ \widehat{e} \notin in_3 \ \wedge \ (\exists (\widehat{q}, \widehat{U}, \widehat{q}') \in A : \widehat{e} \in \widehat{U} \ \wedge \ \widehat{q}' \in r_3)$. So, there exists $(\widehat{q}, \widehat{U}, \widehat{q}') \in A$ such that $\widehat{e} \in \widehat{U} \cap out_2$ and $\widehat{q}' \in r_3$. This and $r_3 \cap r = \varnothing$ implies that $(\widehat{q}, \widehat{U}, \widehat{q}')$ is a transition outgoing from $r = r_1 \cup r_2$, so we have $\widehat{e} \notin H$

and, consequently, $\widehat{e} \in out$. From assumptions we have that $\mathfrak{r}$ and $\mathfrak{r}_3$ are strongly compatible, so the existence of $(\widehat{q},\widehat{U},\widehat{q}')$ with $\widehat{q}' \in r_3$ means $\widehat{e} \in in_3$, a contradiction with $\widehat{e} \notin in_3$. So, (†) holds.

Suppose, to the contrary, that (††) is not true. Then $\exists \widehat{e} \in in_2 : \ \widehat{e} \notin out_3 \ \wedge \ (\exists (\widehat{q},\widehat{U},\widehat{q}') \in A : \ \widehat{e} \in \widehat{U} \ \wedge \ \widehat{q} \in r_3)$. So, there exists $(\widehat{q},\widehat{U},\widehat{q}') \in A$ such that $\widehat{e} \in \widehat{U} \cap in_2$ and $\widehat{q} \in r_3$. This and $r_3 \cap r = \varnothing$ implies that $(\widehat{q},\widehat{U},\widehat{q}')$ is a transition incoming into $r = r_1 \cup r_2$, so we have $\widehat{e} \notin H$ and, consequently, $\widehat{e} \in in$. From assumptions we have that $\mathfrak{r}$ and $\mathfrak{r}_3$ are strongly compatible, so the existence of $(\widehat{q},\widehat{U},\widehat{q}')$ with $\widehat{q} \in r_3$ means $\widehat{e} \in out_3$, a contradiction with $\widehat{e} \notin out_3$. So, (††) holds.

Consequently, $\mathfrak{r}_2$ is strongly compatible with $\mathfrak{r}_3$.

To prove that $\mathfrak{r}_3$ is strongly compatible with $\mathfrak{r}_2$ we still need to show:

$$\forall e \in out_3 : \ e \in in_2 \ \vee \ (\forall (q,U,q') \in A : \ e \in U \implies q' \notin r_2) \qquad (‡)$$

$$\forall e \in in_3 : \ e \in out_2 \ \vee \ (\forall (q,U,q') \in A : \ e \in U \implies q \notin r_2) \qquad (‡‡)$$

Suppose, to the contrary, that (‡) is not true. Then $\exists \widehat{e} \in out_3 : \ \widehat{e} \notin in_2 \ \wedge \ (\exists (\widehat{q},\widehat{U},\widehat{q}') \in A : \ \widehat{e} \in \widehat{U} \ \wedge \ \widehat{q}' \in r_2)$. From the assumption that $\mathfrak{r}$ and $\mathfrak{r}_3$ are strongly compatible and the existence of $(\widehat{q},\widehat{U},\widehat{q}')$ with $\widehat{q}' \in r_2 \subset r$ and $\widehat{e} \in out_3 \cap \widehat{U}$, we have $\widehat{e} \in in = in_1 \cup in_2 \setminus H$ and consequently $\widehat{e} \in in_2$ (see Facts 5 and 6), a contradiction with $\widehat{e} \notin in_2$. So, (‡) holds.

Suppose, to the contrary, that (‡‡) is not true. Then $\exists \widehat{e} \in in_3 : \ \widehat{e} \notin out_2 \ \wedge \ (\exists (\widehat{q},\widehat{U},\widehat{q}') \in A : \ \widehat{e} \in \widehat{U} \ \wedge \ \widehat{q} \in r_2)$. From the assumption that $\mathfrak{r}$ and $\mathfrak{r}_3$ are strongly compatible and the existence of $(\widehat{q},\widehat{U},\widehat{q}')$ with $\widehat{q} \in r_2 \subset r$ and $\widehat{e} \in in_3 \cap \widehat{U}$, we have $\widehat{e} \in out = out_1 \cup out_2 \setminus H$ and consequently $\widehat{e} \in out_2$ (see Facts 5 and 6), a contradiction with $\widehat{e} \notin out_2$. So, (‡‡) holds.

Hence, $\mathfrak{r}_3$ is strongly compatible with $\mathfrak{r}_2$ and so $\mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible regions. So, we can compose them obtaining:

$$\mathfrak{r}' = (in', r', out') = \mathfrak{r}_2 \oplus_s \mathfrak{r}_3 = (in_2 \cup in_3 \setminus H', r_2 \cup r_3, out_2 \cup out_3 \setminus H'),$$

where H' is a set of events that belong <u>only</u> to steps labelling transitions buried in $r_2 \cup r_3$.

We need to show now that $\mathfrak{r}_1$ and $\mathfrak{r}'$ are strongly compatible. We already showed that $r_1 \cap r_2 = \varnothing$ and $r_1 \cap r_3 = \varnothing$. So, $r_1 \cap (r_2 \cup r_3) = \varnothing$.

Next we show that $\mathfrak{r}_1$ is strongly compatible with $\mathfrak{r}'$. To do so, we still need to show that:

$$\forall e \in out_1 : \ e \in in' \ \vee \ (\forall (q,U,q') \in A : \ e \in U \implies q' \notin r') \qquad (\triangle)$$

$$\forall e \in in_1 : \ e \in out' \ \vee \ (\forall (q,U,q') \in A : \ e \in U \implies q \notin r') \qquad (\triangle\triangle)$$

Suppose, to the contrary, that ($\triangle$) is not true. Then $\exists \widehat{e} \in out_1 : \ \widehat{e} \notin in' \ \wedge \ (\exists(\widehat{q},\widehat{U},\widehat{q}') \in A : \ \widehat{e} \in \widehat{U} \ \wedge \ \widehat{q}' \in r')$. We now consider two cases:

1. $\widehat{q}' \in r_2$.
   As $(\widehat{q},\widehat{U},\widehat{q}')$ is a transition incoming into r_2 and $\widehat{q} \in r_1$ ($\widehat{e} \in out_1$) and $r_1 \cap (r_2 \cup r_3) = \varnothing$ and $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are strongly compatible, we have $\widehat{e} \notin H'$ and $\widehat{e} \in in_2$. So, $\widehat{e} \in in' = in_2 \cup in_3 \setminus H'$, a contradiction with $\widehat{e} \notin in'$.

2. $\widehat{q}' \in r_3$.
   From the assumption that $\mathfrak{r} = \mathfrak{r}_1 \oplus_s \mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible we have $r \cap r_3 = \varnothing$. This and the existence of the transition $(\widehat{q},\widehat{U},\widehat{q}')$, where $\widehat{e} \in \widehat{U} \cap out_1$ and $\widehat{q}' \in r_3$, means that $\widehat{e} \notin H$ and consequently $\widehat{e} \in out = out_1 \cup out_2 \setminus H$. Furthermore, as $\mathfrak{r}$ and $\mathfrak{r}_3$ are strongly compatible, we have $\widehat{e} \in in_3$. Also, $\widehat{e} \notin H'$, because the transition $(\widehat{q},\widehat{U},\widehat{q}')$ has the source $\widehat{q} \in r_1$ ($r_1 \cap r' = \varnothing$). So, $\widehat{e} \in in' = in_2 \cup in_3 \setminus H'$, a contradiction with $\widehat{e} \notin in'$.

We obtained a contradiction in both considered cases, so ($\triangle$) holds.

Now we prove ($\triangle\triangle$). Suppose, to the contrary, that ($\triangle\triangle$) is not true. Then $\exists \widehat{e} \in in_1 : \widehat{e} \notin out' \ \wedge \ (\exists(\widehat{q},\widehat{U},\widehat{q}') \in A : \ \widehat{e} \in \widehat{U} \ \wedge \ \widehat{q} \in r')$. We now consider two cases:

1. $\widehat{q} \in r_2$.
   As $(\widehat{q},\widehat{U},\widehat{q}')$ is a transition outgoing from r_2 and $\widehat{q}' \in r_1$ ($\widehat{e} \in in_1$) and $r_1 \cap (r_2 \cup r_3) = \varnothing$ and $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are strongly compatible, we have $\widehat{e} \notin H'$ and $\widehat{e} \in out_2$. So, $\widehat{e} \in out' = out_2 \cup out_3 \setminus H'$, a contradiction with $\widehat{e} \notin out'$.

2. $\widehat{q} \in r_3$.
   From the assumption that $\mathfrak{r} = \mathfrak{r}_1 \oplus_s \mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible we have $r \cap r_3 = \varnothing$. This and the existence of the transition $(\widehat{q},\widehat{U},\widehat{q}')$, where $\widehat{e} \in \widehat{U} \cap in_1$ and $\widehat{q} \in r_3$, means $\widehat{e} \notin H$ and consequently $\widehat{e} \in in = in_1 \cup in_2 \setminus H$. Furthermore, as $\mathfrak{r}$ and $\mathfrak{r}_3$ are strongly compatible, we have $\widehat{e} \in out_3$. Also, $\widehat{e} \notin H'$, because the transition $(\widehat{q},\widehat{U},\widehat{q}')$ has the target $\widehat{q}' \in r_1$ ($r_1 \cap r' = \varnothing$). So, $\widehat{e} \in out' = out_2 \cup out_3 \setminus H'$, a contradiction with $\widehat{e} \notin out'$.

We obtained a contradiction in both considered cases, so ($\triangle\triangle$) holds. Therefore, region $\mathfrak{r}_1$ is strongly compatible with region $\mathfrak{r}' = \mathfrak{r}_2 \oplus_s \mathfrak{r}_3$.

Now we prove that $\mathfrak{r}' = \mathfrak{r}_2 \oplus_s \mathfrak{r}_3$ is strongly compatible with $\mathfrak{r}_1$. To do so, we still need to show:

$$\forall e \in out' : \ e \in in_1 \ \vee \ (\forall(q,U,q') \in A : \ e \in U \implies q' \notin r_1) \qquad (\sharp)$$

$$\forall e \in in' : \ e \in out_1 \ \vee \ (\forall(q,U,q') \in A : \ e \in U \implies q \notin r_1) \qquad (\sharp\sharp)$$

Suppose, to the contrary, that (♯) is not true. Then $\exists \widehat{e} \in out' : \widehat{e} \notin in_1 \wedge (\exists (\widehat{q}, \widehat{U}, \widehat{q}') \in A : \widehat{e} \in \widehat{U} \wedge \widehat{q}' \in r_1)$. We now consider two cases:

1. $\widehat{q} \in r_2$.
   As $(\widehat{q}, \widehat{U}, \widehat{q}')$ is a transition outgoing from r_2 and $\widehat{q}' \in r_1$, $r_1 \cap r' = \varnothing$ and $\widehat{e} \in out'$, we have $\widehat{e} \notin H'$ and $\widehat{e} \in out_2$ (as $out' = out_2 \cup out_3 \setminus H'$, and from Facts 5 and 6 we have $out_2 \cap out_3 = \varnothing$ as $\mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible). Hence, from the fact that $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are strongly compatible we have that $\widehat{e} \in in_1$, a contradiction with $\widehat{e} \notin in_1$.

2. $\widehat{q} \in r_3$.
   As $(\widehat{q}, \widehat{U}, \widehat{q}')$ is a transition outgoing from r_3 and $\widehat{q}' \in r_1$, $r_1 \cap r' = \varnothing$ and $\widehat{e} \in out'$, we have $\widehat{e} \notin H'$ and $\widehat{e} \in out_3$ (as $\widehat{e} \in out' = out_2 \cup out_3 \setminus H'$, and from Facts 5 and 6 we have $out_2 \cap out_3 = \varnothing$ as $\mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible). From the fact that $\mathfrak{r} = \mathfrak{r}_1 \oplus_s \mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible we have that $\widehat{e} \in in$ ($\widehat{q}' \in r_1 \subset r$). From the existence of the transition $(\widehat{q}, \widehat{U}, \widehat{q}')$, where $\widehat{e} \in \widehat{U}$, $\widehat{q} \in r_3$, $\widehat{q}' \in r_1 \subset r$ and $r \cap r_3 = \varnothing$, we have that $\widehat{e} \notin H$. As $\widehat{e} \in in = in_1 \cup in_2 \setminus H$ and $\widehat{q}' \in r_1$, we have $\widehat{e} \in in_1$ (from Facts 5 and 6 we have $in_1 \cap in_2 = \varnothing$ as $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are strongly compatible). So, we obtained a contradiction with $\widehat{e} \notin in_1$.

We obtained a contradiction in both considered cases, so (♯) holds.

We now prove (♯♯). Suppose, to the contrary, that (♯♯) is not true. Then $\exists \widehat{e} \in in' : \widehat{e} \notin out_1 \wedge (\exists (\widehat{q}, \widehat{U}, \widehat{q}') \in A : \widehat{e} \in \widehat{U} \wedge \widehat{q} \in r_1)$. We now consider two cases:

1. $\widehat{q}' \in r_2$.
   As $(\widehat{q}, \widehat{U}, \widehat{q}')$ is a transition incoming into r_2 and $\widehat{q} \in r_1$, $r_1 \cap r' = \varnothing$ and $\widehat{e} \in in'$, we have $\widehat{e} \notin H'$ and $\widehat{e} \in in_2$ (as $in' = in_2 \cup in_3 \setminus H'$, and from Facts 5 and 6 we have $in_2 \cap in_3 = \varnothing$ as $\mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible). Hence, from the fact that $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are strongly compatible we have that $\widehat{e} \in out_1$, a contradiction with $\widehat{e} \notin out_1$.

2. $\widehat{q}' \in r_3$.
   As $(\widehat{q}, \widehat{U}, \widehat{q}')$ is a transition incoming into r_3 and $\widehat{q} \in r_1$, $r_1 \cap r' = \varnothing$ and $\widehat{e} \in in'$, we have $\widehat{e} \notin H'$ and $\widehat{e} \in in_3$ (as $\widehat{e} \in in' = in_2 \cup in_3 \setminus H'$, and from Facts 5 and 6 we have $in_2 \cap in_3 = \varnothing$ as $\mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible). From the fact that $\mathfrak{r} = \mathfrak{r}_1 \oplus_s \mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible we have that $\widehat{e} \in out$ ($\widehat{q} \in r_1 \subset r$). From the existence of the transition $(\widehat{q}, \widehat{U}, \widehat{q}')$, where $\widehat{e} \in \widehat{U}$, $\widehat{q} \in r_1 \subset r$, $\widehat{q}' \in r_3$ and $r \cap r_3 = \varnothing$, we have that $\widehat{e} \notin H$. As $\widehat{e} \in out = out_1 \cup out_2 \setminus H$ and $\widehat{q} \in r_1$, we have $\widehat{e} \in out_1$ (from Facts 5 and 6 we have $out_1 \cap out_2 = \varnothing$ as $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are strongly compatible). So, we obtained a contradiction with $\widehat{e} \notin out_1$.

We obtained a contradiction in both considered cases, so ($\sharp\sharp$) holds. Therefore, $\mathfrak{r}_2 \oplus_s \mathfrak{r}_3$ is strongly compatible with $\mathfrak{r}_1$, and consequently regions $\mathfrak{r}_2 \oplus_s \mathfrak{r}_3$ and $\mathfrak{r}_1$ are strongly compatible regions.

Finally, we need to show that:

$$(\mathfrak{r}_1 \oplus_s \mathfrak{r}_2) \oplus_s \mathfrak{r}_3 = \mathfrak{r}_1 \oplus_s (\mathfrak{r}_2 \oplus_s \mathfrak{r}_3),$$

where

$$\begin{array}{rclcl} \mathfrak{r} & = & \mathfrak{r}_1 \oplus_s \mathfrak{r}_2 & = & (in_1 \cup in_2 \setminus H, r_1 \cup r_2, out_1 \cup out_2 \setminus H) \text{ and} \\ \mathfrak{r}' & = & \mathfrak{r}_2 \oplus_s \mathfrak{r}_3 & = & (in_2 \cup in_3 \setminus H', r_2 \cup r_3, out_2 \cup out_3 \setminus H'). \end{array}$$

Let H'' be a set of events that belong only to steps labelling transitions buried in $r_1 \cup r_3$ and $\widehat{H} = H \cup H' \cup H''$.

We need to show: $L = R$, where

$$\begin{array}{rcl} L & = & \big((in_1 \cup in_2 \setminus H) \cup in_3 \setminus \widehat{H}, r_1 \cup r_2 \cup r_3, (out_1 \cup out_2 \setminus H) \cup out_3 \setminus \widehat{H}\big) \text{ and} \\ R & = & \big(in_1 \cup (in_2 \cup in_3 \setminus H') \setminus \widehat{H}, r_1 \cup r_2 \cup r_3, out_1 \cup (out_2 \cup out_3 \setminus H') \setminus \widehat{H}\big). \end{array}$$

Observe first that $H \cap in_3 = \varnothing$ and $H \cap out_3 = \varnothing$, so in the equation for L above we can use $\widehat{H}$ instead of $H' \cup H''$. Similarly, as $H' \cap in_1 = \varnothing$ and $H' \cap out_1 = \varnothing$, we can use $\widehat{H}$ instead of $H \cup H''$ in the equation for R.

Now we show that $L_{in} = (in_1 \cup in_2 \setminus H) \cup in_3 \setminus \widehat{H} = in_1 \cup (in_2 \cup in_3 \setminus H') \setminus \widehat{H} = R_{in}$. From the definitions of H, H' and H'' we have $H \cap H' = \varnothing$, $H \cap H'' = \varnothing$ and $H' \cap H'' = \varnothing$. From Facts 5 and 6 and the fact that $\mathfrak{r}_1$ and $\mathfrak{r}_2$ and $\mathfrak{r}_2$ and $\mathfrak{r}_3$ are strongly compatible we have $in_1 \cap in_2 = \varnothing$ and $out_1 \cap out_2 = \varnothing$ as well as $in_2 \cap in_3 = \varnothing$ and $out_2 \cap out_3 = \varnothing$. Furthermore, $in_1 \cap in_3 = \varnothing$ as every transition labelled with a step containing $e \in in_1$ must enter r_1 and every transition labelled with a step containing $e \in in_3$ must enter r_3, but $r_1 \cap r_3 = \varnothing$, so $in_1 \cap in_3 \neq \varnothing$ is impossible. Hence, the difference between the sets of events of L_{in} and R_{in}, before taking away events from $\widehat{H}$, results from the events that are contained in the intersections between the various sets of 'buried' events (H, H', H'') and *in* sets (in_1, in_2, in_3). However, since $H \subseteq \widehat{H}$, $H' \subseteq \widehat{H}$ and $H'' \subseteq \widehat{H}$, the events that cause the difference between the two sides of the equation will be removed when the set $\widehat{H}$ is subtracted from both sides of the equation, leaving $L_{in} = R_{in}$.

Similarly we can show that:

$$L_{out} = (out_1 \cup out_2 \setminus H) \cup out_3 \setminus \widehat{H} = out_1 \cup (out_2 \cup out_3 \setminus H') \setminus \widehat{H} = R_{out}.$$

□

Proposition 6.3.3. *Let $\mathfrak{r}_1 = (in_1, r_1, out_1)$ and $\mathfrak{r}_2 = (in_2, r_2, out_2)$ be compatible regions of an* ST*-system $\mathfrak{ts} = (Q, A, q_0)$, which do not satisfy the conditions to be strongly compatible regions of $\mathfrak{ts}$. Then there exists a companion region of $\mathfrak{r}_1$, $\mathfrak{r}'_1 \in \mathfrak{R}^{r_1}_{\mathfrak{ts}}$, and a companion region of $\mathfrak{r}_2$, $\mathfrak{r}'_2 \in \mathfrak{R}^{r_2}_{\mathfrak{ts}}$, such that $\mathfrak{r}_1$ and $\mathfrak{r}'_2$ are strongly compatible and $\mathfrak{r}'_1$ and $\mathfrak{r}_2$ are strongly compatible. Furthermore, $\mathfrak{r}_1 \oplus_s \mathfrak{r}'_2 = \mathfrak{r}'_1 \oplus_s \mathfrak{r}_2 = \mathfrak{r}_1 \oplus \mathfrak{r}_2$.*

Proof. First, we show the existence of regions $\mathfrak{r}'_1$ and $\mathfrak{r}'_2$.

Since $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are compatible we have $r_1 \cap r_2 = \varnothing$. Bearing in mind that $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are compatible regions we now consider two cases:

1. There are no 'buried' transitions in $r_1 \cup r_2$.
   For this case the definitions of compatibility and strong compatibility (Definitions 6.1.1 and 6.3.1) coincide. Hence, $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are strongly compatible and $\mathfrak{r}'_1 = \mathfrak{r}_1$ and $\mathfrak{r}'_2 = \mathfrak{r}_2$.

2. There are 'buried' transitions in $r_1 \cup r_2$.
   Then, without loss of generality, we can assume that there exists a transition $(q, U, q') \in A$ such that $q \in r_1$ and $q' \in r_2$. Hence, from Definition 2.1.6(**R1**) for $\mathfrak{r}_1$ and (q, U, q'), and $r_1 \cap r_2 = \varnothing$, we have $|U \cap out_1| = 1$. Let $e_1 \in U \cap out_1$. As $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are <u>not</u> strongly compatible we can assume, again without loss of generality, that Definition 6.3.1 fails for $e_1 \in out_1$ resulting in $e_1 \notin in_2$. Furthermore, from Definition 2.1.6(**R2**) for $\mathfrak{r}_2$ and (q, U, q'), and $r_1 \cap r_2 = \varnothing$, we have $|U \cap in_2| = 1$. Since $e_1 \notin in_2$, there exists $e_2 \neq e_1$ such that $e_2 \in U \cap in_2$. Therefore, $|U| \geq 2$. From Definition 2.1.6(**R1**) for $\mathfrak{r}_1$ and (q, U, q'), we have that $e_2 \notin out_1$.

   The manifestation of $e_1 \notin in_2$ (recall $e_1 \in out_1$) might be the existence of a transition (q_1, U_1, q'_1) such that $q_1 \in r_1$, $q'_1 \notin r_1 \cup r_2$ and $e_1 \in U_1$. The manifestation of $e_2 \notin out_1$ (recall $e_2 \in in_2$) might be the existence of a transition (q_2, U_2, q'_2) such that $q_2 \notin r_1 \cup r_2$, $q'_2 \in r_2$ and $e_2 \in U_2$. However, there is not possible for the two such transitions to exist considering the existence of the transition (q, U, q'). This is because $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are compatible regions, and therefore they can be composed producing the region: $\mathfrak{r}_{sum} = \mathfrak{r}_1 \oplus \mathfrak{r}_2 = (in_{sum}, r_{sum}, out_{sum}) = (in_1 \cup in_2 \setminus H, r_1 \cup r_2, out_1 \cup out_2 \setminus H)$ and then $e_1 \notin H$ (because of (q_1, U_1, q'_1)) and $e_2 \notin H$ (because of (q_2, U_2, q'_2)). Hence, as $e_1 \in out_1$ and $e_2 \in in_2$, we have $e_1 \in out_{sum}$ and $e_2 \in in_{sum}$. Neither of these is possible, because of the existence of the transition (q, U, q'), as $e_1, e_2 \in U$ and this transition is 'buried' in $r_1 \cup r_2$. So, all transitions labelled by steps containing e_1 or e_2 are 'buried' in $r_1 \cup r_2$ and the transition (q, U, q') has a label containing both of them.

   For the thick transitions that are 'buried' in $r_1 \cup r_2$ (with source in r_1 and target in r_2) we can have a mismatch between the event that leads out from r_1 and the one that leads into r_2. We will now show how to alleviate such mismatches like the mismatch between $e_1 \in out_1$ and $e_2 \in in_2$ for the transition (q,U,q'). Let $U^{r_1,r_2} = \{U \mid \exists (q,U,q') \in A : q \in r_1,\ q' \in r_2 \text{ and } |U| \geq 2\}$. Also, let $pairs^{r_1,r_2} \colon U^{r_1,r_2} \to out_1 \times in_2$ be a function, which assigns, for labels of thick transitions going from r_1 to r_2, their pairs of events: leading out of r_1 and leading into r_2. Furthermore, let N be a set of events from E, which are not contained in labels of transitions 'buried' in $r_1 \cup r_2$. We now define [4]:

$$\begin{array}{rcll} out'_1 & = & out_1 \cap N & \cup \ \ second(pairs^{r_1,r_2}(U^{r_1,r_2})) \text{ and} \\ in'_2 & = & in_2 \cap N & \cup \ \ first(pairs^{r_1,r_2}(U^{r_1,r_2})). \end{array}$$

We can then define tuples: (in_1, r_1, out'_1) and (in'_2, r_2, out_2). It is clear that the first tuple is a region as $\mathfrak{r}_1$ is a region and Definition 2.1.6(**R1,R3**) is satisfied for out'_1. Also, it is clear that the second tuple is a region as $\mathfrak{r}_2$ is a region and Definition 2.1.6(**R2,R4**) is satisfied for in'_2. Furthermore, assuming that we had only mismatches between events contained in the labels of transitions going from r_1 to r_2, it can be seen that $\mathfrak{r}'_1 = (in_1, r_1, out'_1)$ and $\mathfrak{r}_2$ are strongly compatible regions (as $out'_1 \setminus out_1 \cap N \subseteq in_2$), and $\mathfrak{r}_1$ and $\mathfrak{r}'_2 = (in'_2, r_2, out_2)$ are strongly compatible regions (as $in'_2 \setminus in_2 \cap N \subseteq out_1$). The mismatches between events contained in the labels of transitions going from r_2 to r_1 can be alleviated in a similar way.

Finally, we observe that $\mathfrak{r}_1 \oplus_s \mathfrak{r}'_2 = \mathfrak{r}'_1 \oplus_s \mathfrak{r}_2 = \mathfrak{r}_1 \oplus \mathfrak{r}_2$ is true as: (1) the sets of states for all three pairs of regions is $r_1 \cup r_2$; and (2) the only difference between the in/out sets of regions involved in the three pairs are due to the events that belong only to steps labelling transitions 'buried' in $r_1 \cup r_2$, but they will be removed from the appropriate sums of the in/out sets when the compositions of regions are formed. □

To illustrate the result of Proposition 6.3.3, we can use again the ENLST-system in Figure 6.2(a). A pair of its regions, $\mathfrak{r}_4$ and $\mathfrak{r}_1$, are compatible, but not strongly compatible regions (see Table 6.1). However, there are regions $\mathfrak{r}_3 \in \mathfrak{R}^{r_4}_{\mathfrak{ts}}$ and $\mathfrak{r}_2 \in \mathfrak{R}^{r_1}_{\mathfrak{ts}}$, such that $\mathfrak{r}_4$ and $\mathfrak{r}_2$ are strongly compatible and $\mathfrak{r}_3$ and $\mathfrak{r}_1$ are strongly compatible. Also, we have $\mathfrak{r}_4 \oplus_s \mathfrak{r}_2 = \mathfrak{r}_3 \oplus_s \mathfrak{r}_1 = \mathfrak{r}_4 \oplus \mathfrak{r}_1 = (\varnothing, Q, \varnothing)$. Furthermore, using the same example, observe that when we express a composition of two <u>minimal</u> compatible regions by a composition of two strongly compatible regions, one of the two strongly compatible regions might be not a minimal region. For example, $\mathfrak{r}_2 \oplus \mathfrak{r}_3 = \mathfrak{r}_2 \oplus_s \mathfrak{r}_4$, where $\mathfrak{r}_2$ and $\mathfrak{r}_3$ are minimal regions, but $\mathfrak{r}_4$ is not. Proposition 6.3.3 does not take the minimality of regions into consideration.

Corollary 6.3.4. *Let* $\mathfrak{ts} = (Q, A, q_0)$ *be an* ENLST-*system. Then*

[4]For any sets X and Y, $first \colon X \times Y \to X$ and $second \colon X \times Y \to Y$ are mappings defined as follows: $first(x,y) = x$ and $second(x,y) = y$, where $x \in X$ and $y \in Y$.

1. $\mathfrak{R}^{min}_{\mathfrak{ts}} = \mathfrak{R}^{min,s}_{\mathfrak{ts}}$.

2. Every $\mathfrak{r} \in \mathfrak{R}_{\mathfrak{ts}}$ *can be represented as a composition of minimal regions, where each pair of different minimal regions in this representation is a pair of strongly compatible regions.*

3. Let $\mathfrak{R} = \{\mathfrak{r}_1, \mathfrak{r}_2, \ldots, \mathfrak{r}_n\}$ *be a set of pairwise strongly compatible non-trivial regions of* $\mathfrak{ts}$*, where* $\mathfrak{r}_i = (in_i, r_i, out_i)$*,* $i \in \{1, \ldots, n\}$*. Then there exists a region* $\mathfrak{r} = (in, r, out) = \mathfrak{r}_1 \oplus_s \ldots \oplus_s \mathfrak{r}_n$*, where* $r = r_1 \cup \ldots \cup r_n$*.*

Proof. 1. The minimal regions of $\mathfrak{ts}$ can be obtained from $\mathfrak{R}_{\mathfrak{ts}}$ by elimination of non-minimal regions. From Proposition 6.3.3 it follows that any non-minimal region that can be represented as a composition of two compatible regions can also be represented as a composition of two strongly compatible regions. So, the same set of regions will be eliminated from $\mathfrak{R}_{\mathfrak{ts}}$ regardless of which strict pre-order is used ($\prec$ or $\prec_s$) when defining non-minimal (minimal) regions. Hence, $\mathfrak{R}^{min}_{\mathfrak{ts}} = \mathfrak{R}^{min,s}_{\mathfrak{ts}}$.

2. The proof is similar to the proof of Theorem 6.2.2. However, as $\oplus_s$ is an associative operator (see Proposition 6.3.2), we can drop the brackets, if there are any, in the final representation of $\mathfrak{r}$.

3. Assume first that $n = 3$. We show that each of the three regions ($\mathfrak{r}_1$, $\mathfrak{r}_2$ and $\mathfrak{r}_3$) and the region obtained by composing the two remaining regions by means of $\oplus_s$ form a pair of strongly compatible regions, which after being composed produce a region $\mathfrak{r}_1 \oplus_s \mathfrak{r}_2 \oplus_s \mathfrak{r}_3$. As an example, we can prove that $(\mathfrak{r}_1 \oplus_s \mathfrak{r}_2)$ and $\mathfrak{r}_3$ form a pair of strongly compatible regions. This part of the proof is omitted as it uses the same techniques as were employed in Proposition 6.3.2.

   Assuming now that $n > 3$, we can prove in a similar way that $(\mathfrak{r}_1 \oplus_s \mathfrak{r}_2)$ and $\mathfrak{r}_i$, $3 < i \leq n$, are pairs of strongly compatible regions. Since $\{\mathfrak{r}_3, \ldots, \mathfrak{r}_n\}$ is a set of pairwise strongly compatible regions by assumption, we have that the set $\{(\mathfrak{r}_1 \oplus_s \mathfrak{r}_2), \mathfrak{r}_3, \ldots, \mathfrak{r}_n\}$ is a set of $n-1$ regions that are non-trivial and pairwise strongly compatible. We can continue this process, decrementing by 1 in each step the number of regions in this set of regions, till we have just one region in this set: $\mathfrak{r} = (in, r, out) = \mathfrak{r}_1 \oplus_s \mathfrak{r}_2 \oplus_s \ldots \oplus_s \mathfrak{r}_n$, where $r = r_1 \cup \ldots \cup r_n$.

□

The next result, about special families of non-trivial regions of $\mathfrak{ts}$, is inspired by a result proved for the class of Elementary Net Systems in [13]. We have adapted this result here for the context of ENL-systems by changing one of the original conditions that a family of regions should satisfy, but the implied result is the same: a family of regions that satisfy the conditions of Theorem 6.3.5, treated as a set of conditions of the synthesised net, would generate a state machine component of this net. Points 2 and 3 of the consequent of Theorem 6.3.5 guarantee the satisfaction of Definition 2.1.3(1) and the point 1 of the consequent of Theorem 6.3.5 guarantees the satisfaction of Definition 2.1.3(2).

Theorem 6.3.5. *Let $\mathfrak{R} = \{\mathfrak{r}_1, \mathfrak{r}_2, \ldots, \mathfrak{r}_n\}$ be a family of non-trivial regions of $\mathfrak{ts} = (Q, A, q_0)$, where $\mathfrak{r}_i = (in_i, r_i, out_i)$, $i \in \{1, \ldots, n\}$, satisfy the following conditions:*

1. *Every two different regions $\mathfrak{r}_i, \mathfrak{r}_j \in \mathfrak{R}$ are strongly compatible regions.*
2. $\forall \widehat{\mathfrak{r}} = (\widehat{in}, \widehat{r}, \widehat{out}) \in \mathfrak{R}_{\mathfrak{ts}} : \widehat{\mathfrak{r}} \notin \mathfrak{R} \implies (\exists \mathfrak{r}_i \in \mathfrak{R} : \widehat{r} \cap r_i \neq \varnothing)$.

Then:

1. $\bigcup r_i = Q$;
2. $\forall e \in E : |{}^{\circ}e \cap \mathfrak{R}| \leq 1$ *and* $|e^{\circ} \cap \mathfrak{R}| \leq 1$;
3. $\forall e \in E : e \in {}^{\circ}\mathfrak{r}_i \iff \exists j : e \in \mathfrak{r}_j{}^{\circ}$.

Proof.

1. Suppose $q \in Q \setminus \bigcup r_i$. As all regions of $\mathfrak{R}$ are pairwise strongly compatible then there exists a region $\mathfrak{r} = (in, r, out) = \mathfrak{r}_1 \oplus_s \ldots \oplus_s \mathfrak{r}_n$, where $r = r_1 \cup \ldots \cup r_n$ and $r_i \cap r_j = \varnothing$ for different $i, j \in \{1, \ldots, n\}$ (see Corollary 6.3.4(3)). Also, $\bar{\mathfrak{r}}$ is a region with $\bar{r} = Q \setminus \{r_1 \cup \ldots \cup r_n\}$ (disjoint from all r_i) and $\bar{\mathfrak{r}}$ is non-trivial ($q \in \bar{r}$), contradicting the second assumption that $\mathfrak{R}$ must satisfy.

2. Suppose to the contrary that there are $\mathfrak{r}_i$ and $\mathfrak{r}_j$ in $\mathfrak{R}$ such that $\mathfrak{r}_i$, $\mathfrak{r}_j \in {}^{\circ}e$. Hence, $e \in out_i \cap out_j$. So, for all transitions $(q, U, q') \in A$, where $e \in U$, we have $q \in r_i$ and $q \in r_j$ (see Definition 2.1.6(**R3**)), but $r_i \cap r_j = \varnothing$ - a contradiction. The second part of this point can be proved in a similar way.

3. Suppose $e \in {}^{\circ}\mathfrak{r}_i$. Then $\mathfrak{r}_i \in e^{\circ}$ and consequently $e \in in_i$. As, by definition, every event $e \in E$ occurs in at least one of the steps labelling the transitions of $\mathfrak{ts}$, we have a transition $(q, U, q') \in A$ such that $e \in U$. Therefore, from Definition 2.1.6(**R4**) we have $q \notin r_i$ and $q' \in r_i$. From Theorem 6.3.5(1), we have that $r_i = Q \setminus \bigcup_{k \in \{1, \ldots, n\} \setminus \{i\}} r_k$. So, there is $j \neq i$ such that $q \in r_j$ and $q' \notin r_j$ (the sets of states of $\mathfrak{r}_i$ and $\mathfrak{r}_j$ are disjoint as these regions are compatible). Hence, from Definition 2.1.6(**R1**), we have that $|U \cap out_j| = 1$, meaning that there is $\widehat{e} \in U$ such that $\widehat{e} \in out_j$. Since $\mathfrak{r}_i$ and $\mathfrak{r}_j$ are strongly compatible regions, we have $e = \widehat{e}$. So, $e \in out_j$, and therefore $\mathfrak{r}_j \in {}^{\circ}e$ and consequently $e \in \mathfrak{r}_j{}^{\circ}$. The converse implication can be proved in a similar way.

□

The saturated ENL-system that is a solution to Problem 1 for a given ENLST-system $\mathfrak{ts}$, $\mathfrak{enl} = \mathfrak{enl}_{\mathfrak{ts}}^{\widehat{\ }}$, and is based on all non-trivial regions, is state machine decomposable (see Definition 2.1.5), as due to Fact 7 every pair of complementary regions satisfies the conditions of Theorem 6.3.5 and would form a state machine component of $\mathfrak{enl}$. Furthermore, from Corollary 6.3.4 it follows that, similarly as for the class of Elementary Net Systems (see [13]), the ENL-system obtained from $\mathfrak{enl}$ by deleting all non-minimal regions following Reduction Rule 2 is also state machine decomposable as every region can be represented as a composition of minimal regions (w.r.t. $\prec_s$) and selected subsets of $\mathfrak{R}_{\mathfrak{ts}}^{min} = \mathfrak{R}_{\mathfrak{ts}}^{min,s}$ would satisfy the conditions of Theorem 6.3.5.

As an example we can take the saturated ENL-system synthesised from the ENLST-system in Figure 6.2(a), shown in Figure 6.2(b), and its minimised version shown in Figure 6.2(c). The state machine components of the former ENL-system are generated by the following subsets of conditions/regions:

$$\begin{aligned}
\mathfrak{r}_1 \oplus_s \mathfrak{r}_3 &= (\varnothing, Q, \varnothing)\\
\mathfrak{r}_2 \oplus_s \mathfrak{r}_5 \oplus_s \mathfrak{r}_6 &= (\varnothing, Q, \varnothing)\\
\mathfrak{r}_2 \oplus_s \mathfrak{r}_4 &= (\varnothing, Q, \varnothing)\\
\mathfrak{r}_7 \oplus_s \mathfrak{r}_5 &= (\varnothing, Q, \varnothing)\\
\mathfrak{r}_8 \oplus_s \mathfrak{r}_6 &= (\varnothing, Q, \varnothing)
\end{aligned}$$

The minimised ENL-system in Figure 6.2(c) has the first two state machine components from the list above.

6.4 A strategy to eliminate redundant regions

The three reduction rules give conditions for deleting one of the redundant regions at a time. Therefore, we need a *strategy* to delete as many redundant regions as possible to obtain a net, where all (or almost all) remaining regions are needed (essential). The regions are redundant or essential only in the context of other regions. Different strategies would lead to different sets of essential (or nearly essential) regions. Such sets of regions were called in [20] *admissible* sets of regions.

As an example we can take the ENLST-system in Figure 6.3(a). Its non-trivial regions are:

$$\begin{aligned}
\mathfrak{r}_1 &= (\varnothing, \{q_0\}, \{e,f\}) & \mathfrak{r}_4 &= \bar{\mathfrak{r}}_2 = (\varnothing, \{q_0,q_2\}, \{e\})\\
\mathfrak{r}_2 &= (\{e\}, \{q_1\}, \varnothing) & \mathfrak{r}_5 &= \bar{\mathfrak{r}}_3 = (\varnothing, \{q_0,q_1\}, \{f\})\\
\mathfrak{r}_3 &= (\{f\}, \{q_2\}, \varnothing) & \mathfrak{r}_6 &= \bar{\mathfrak{r}}_1 = (\{e,f\}, \{q_1,q_2\}, \varnothing)
\end{aligned}$$

We observe that the region $\mathfrak{r}_6 = \bar{\mathfrak{r}}_1$ can be deleted according to Reduction Rule 1 (as the complement of region $\mathfrak{r}_1$) or according to Reduction Rule 2 (as a non-minimal region: $\mathfrak{r}_6 = \mathfrak{r}_2 \oplus \mathfrak{r}_3$).

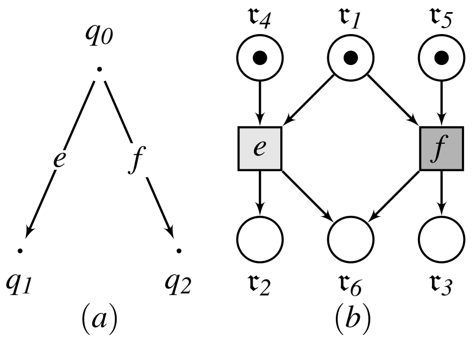


Fig. 6.3 An ENLST-system with two non-collocated events e and f (a); and the ENL-system resulting from its synthesis (b).

When looking for a strategy for deleting redundant regions, we will take into consideration the following criteria:

- Limiting as much as possible the non-determinism in the process of computing admissible regions.
- Effectiveness of the strategy gauged in terms of the number of the removed regions.
- Efficiency of the strategy gauged in terms of time needed to compute a set of admissible regions.

Our first attempt at formulating a strategy will be based on the first criterion listed above. Reduction Rule 2 showed that all non-minimal regions are redundant [5], so we can eliminate first the non-minimal regions. After this step, for a given ENLST-system, we obtain from the unique set of regions, $\mathfrak{R}_{\mathfrak{ts}}$, the unique set of minimal regions: $\mathfrak{R}_{\mathfrak{ts}}^{min}$. The application of Reduction Rule 1 and Reduction Rule 3 might not lead to a unique resultant set of regions. We might decide to keep certain companion regions and delete other companion regions in case of Reduction Rule 3. Similarly, we can keep both or one (random one) out of two complementary regions. As, in general, the Reduction Rule 3, leads to fewer possible choices of regions to delete, and might be even irrelevant in the case of thin step transition systems, where there are no companion regions, we might decide that this rule should be applied

[5]Reduction Rule 2 uses operator $\oplus$ and it was proved in [28] for this operator, but from Corollary 6.3.4(1) we have $\mathfrak{R}_{\mathfrak{ts}}^{min} = \mathfrak{R}_{\mathfrak{ts}}^{min,s}$, so it does not matter whether we use $\prec$ and $\oplus$, or $\prec_s$ and $\oplus_s$, to define the set of minimal regions.

before the Reduction Rule 1, which can lead to many possible combinations of regions to keep/delete. This strategy, called *Strategy (2,3,1)*, can be defined as follows:

1. Use Reduction Rule 2 to delete all non-minimal regions.
2. Use Reduction Rule 3 to delete any redundant companion regions that might be present among the minimal regions.
3. Use Reduction Rule 1 to delete any redundant complementary regions that might be present after the first two steps of the strategy.

To check how good this strategy is from the second criterion point of view we consider a set of ENLST-systems generated by nets composed of several sequential subsystems, where all the events are co-located. Such systems have a lot of companion regions. We will call them $\mathfrak{ts}_{i,j}^{co-loc}$, where the index i denotes the number of sequential subsystems, and the index j denotes the number of events in each of the line-like sequential subsystem. As an example of such a step transition system we can see an ENLST-system $\mathfrak{ts}_{2,2}^{co-loc}$ in Figure 6.4.

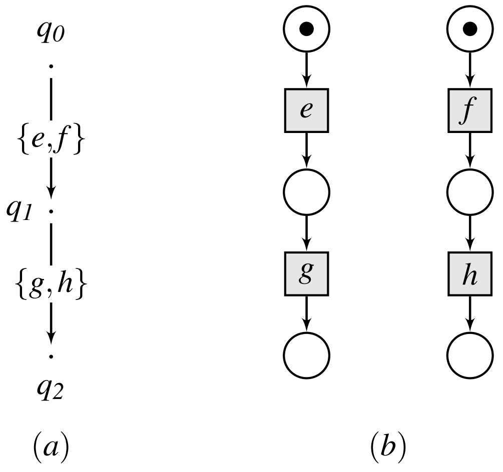


Fig. 6.4 An ENLST-system $\mathfrak{ts}_{2,2}^{co-loc}$ with co-located events e, f, g and h (a), and one of the possible ENL-systems generating it (b).

Using the set of step transition systems $\mathfrak{ts}_{i,j}^{co-loc}$ $(i = 2,\ldots,4;\ j = 2,\ldots,5)$, we compare the effectiveness of region removal of Strategy (2,3,1) and Strategy (3,2,1) (Strategy (2,3,1) with the first two steps reversed). The result of this comparison is presented in Table 6.2.

The results in Table 6.2 are not so surprising. The removal of non-minimal regions makes only sense in the context of all non-trivial regions (as the first step of the strategy). When removing companion regions, the algorithm processes groups of companion regions (each

Table 6.2 Shows comparison between the effectiveness of Strategy (3,2,1) and Strategy (2,3,1), where denotation x - y - z in the last two columns reports the number of remaining regions after the first (x), the second (y) and the third (z) stage of the strategies.

$\mathfrak{ts}$	$\|\mathbf{Q}\|$	$\|\mathbf{E}\|$	$\|\mathfrak{R}_{\mathfrak{ts}}\|$	Strategy (3,2,1)	Strategy (2,3,1)
$\mathfrak{ts}_{2,2}^{co-loc}$	3	4	16	12 - 6 - 6	8 - 6 - 6
$\mathfrak{ts}_{2,3}^{co-loc}$	4	6	52	28 - 10 - 10	12 - 8 - 8
$\mathfrak{ts}_{2,4}^{co-loc}$	5	8	160	60 - 12 - 12	16 - 10 - 10
$\mathfrak{ts}_{2,5}^{co-loc}$	6	10	484	124 - 26 - 26	20 - 12 - 12
$\mathfrak{ts}_{3,2}^{co-loc}$	3	6	30	22 - 11 - 11	15 - 11 - 11
$\mathfrak{ts}_{3,3}^{co-loc}$	4	9	126	66 - 20 - 20	24 - 16 - 16
$\mathfrak{ts}_{3,4}^{co-loc}$	5	12	510	190 - 52 - 51	33 - 21 - 21
$\mathfrak{ts}_{3,5}^{co-loc}$	6	15	2046	546 - 147 - 143	42 - 26 - 26
$\mathfrak{ts}_{4,2}^{co-loc}$	3	8	48	36 - 18 - 18	24 - 18 - 18
$\mathfrak{ts}_{4,3}^{co-loc}$	4	12	248	140 - 48 - 47	40 - 28 - 28
$\mathfrak{ts}_{4,4}^{co-loc}$	5	16	1248	540 - 165 - 159	56 - 38 - 38
$\mathfrak{ts}_{4,5}^{co-loc}$	6	20	6248	2108 - 532 - 508	72 - 48 - 48

based on a shared set of states) separately from each other. From each group some subset of regions may be removed, at random, according to Reduction Rule 3. Once some of the minimal companion regions are removed (if we remove companion regions first), some of the regions that were previously non-minimal would become minimal as it won't be possible to represent them as compositions of minimal regions using the remaining minimal regions. Therefore, they won't be deleted by the Reduction Rule 2, if it is applied after Reduction Rule 3. The results in Table 6.2 show how great would be the loss of effectiveness if we used Strategy (3,2,1) for ENLST-systems with a big number of companion regions.

While the Reduction Rule 2, applied first in our strategy, can be considered as a method for eliminating non-minimal regions, non-minimal from the 'state information' point of view, the Reduction Rule 3 can be understood as a method for eliminating regions that are redundant from the 'event information' point of view. However, some subsets of companion regions will remain after the application of Reduction Rule 3, because they are essential as shown in the following results:

Fact 8. *Let $\mathfrak{r} = (in, r, out)$ be a non-trivial region of $\mathfrak{ts} = (Q, A, q_0)$ and let $\mathfrak{R}^r \subseteq \mathfrak{R}^r_{\mathfrak{ts}}$ be a set of its companion regions (including $\mathfrak{r}$) that remained after the application of the Reduction Rule 3 and let $|\mathfrak{R}^r| \geq 2$. Then all the regions of $\mathfrak{R}^r$ do not satisfy the same conditions of the Reduction Rule 3.*

Proof. Suppose, $\widehat{\mathfrak{r}} \in \mathfrak{R}^r$ and $\widehat{\mathfrak{r}} \neq \mathfrak{r}$. Furthermore, suppose that $\mathfrak{r}$ does not satisfy condition (6.1) of the Reduction Rule 3, and therefore we have:

$$\exists\, e \in in\, \forall\, \mathfrak{r}' = (in', r, out') \in \mathfrak{R}^r : \mathfrak{r}' \neq \mathfrak{r} \implies e \notin in' \tag{6.3}$$

Since $e \in in$ we have that there is a transition $(q, U, q') \in A$ such that $e \in U$ and $q \notin r$ and $q' \in r$. As $\widehat{\mathfrak{r}} = (\widehat{in}, r, \widehat{out}) \in \mathfrak{R}^r$ and $\widehat{\mathfrak{r}} \neq \mathfrak{r}$, we have from (6.3) that $e \notin \widehat{in}$, and therefore there must be $\widehat{e} \neq e$ such that $\widehat{e} \in U \cap \widehat{in}$. From Definition 2.1.6(R2) for $\mathfrak{r}$ we have that $\widehat{e} \notin in$. As $\mathfrak{r} \in \mathfrak{R}^r$ is an example of a companion region of $\widehat{\mathfrak{r}}$ (different than $\widehat{\mathfrak{r}}$), we can see that $\widehat{\mathfrak{r}}$ satisfies (6.3) with $\widehat{\mathfrak{r}}$ taking the role of $\mathfrak{r}$ and $\mathfrak{r}$ taking the role of $\mathfrak{r}'$ in (6.3). Hence, $\widehat{\mathfrak{r}}$ does not satisfy condition (6.1) of the Reduction Rule 3. Similar arguments can be used if $\mathfrak{r}$ does not satisfy condition (6.2) of the Reduction Rule 3. □

Corollary 6.4.1. *Let $\mathfrak{R}^r = \{\mathfrak{r}_1, \mathfrak{r}_2, \ldots, \mathfrak{r}_n\} \subseteq \mathfrak{R}^r_{\mathfrak{ts}}$ be a set of companion regions based on r that remained after the application of the Reduction Rule 3. Then, for every region $\mathfrak{r}_i = (in_i, r, out_i)$ of $\mathfrak{R}^r$ that does not satisfy condition (6.1) (respectively (6.2)) of Reduction Rule 3 there exists a unique $E_i \subseteq in_i$ (respectively $E'_i \subseteq out_i$) with events that are not present in the in (respectively out) sets of other regions from $\mathfrak{R}^r$. So, companion regions of $\mathfrak{R}^r$ are "indexed" by the unique subsets of events of their in (respectively out) sets. We will call these subsets of events* in-indices *(respectively* out-indices*) of r for the regions of $\mathfrak{R}^r$.* ◇

Notice that sets $\mathfrak{R}^r$ in Corollary 6.4.1 (and in Fact 8) might be equal to $\mathfrak{R}^r_{\mathfrak{ts}}$. For example, for $\mathfrak{ts}$ in Figure 6.2(a), we have $\mathfrak{R}^{\{q_0\}} = \mathfrak{R}^{\{q_0\}}_{\mathfrak{ts}} = \{\mathfrak{r}_1, \mathfrak{r}_2\}$. Also, the indexing sets of events do not need to be singleton sets (as, for example, set $\{e_1, e_2\}$ for $\mathfrak{r}_2$ of $\mathfrak{ts}$ in Figure 6.2(a)).

We will further illustrate the above results using the ENLST-system in Figure 6.4(a). Its non-trivial regions are listed below:

$$\begin{array}{lcllcl}
\mathfrak{r}_1 & = & (\varnothing, \{q_0\}, \{e\}) & \bar{\mathfrak{r}}_1 & = & (\{e\}, \{q_1, q_2\}, \varnothing) \\
\mathfrak{r}_2 & = & (\{e\}, \{q_1\}, \{g\}) & \bar{\mathfrak{r}}_2 & = & (\{g\}, \{q_0, q_2\}, \{e\}) \\
\mathfrak{r}_3 & = & (\{g\}, \{q_2\}, \varnothing) & \bar{\mathfrak{r}}_3 & = & (\varnothing, \{q_0, q_1\}, \{g\}) \\
\mathfrak{r}_4 & = & (\{e\}, \{q_1\}, \{h\}) & \bar{\mathfrak{r}}_4 & = & (\{h\}, \{q_0, q_2\}, \{e\}) \\
\mathfrak{r}_5 & = & (\{h\}, \{q_2\}, \varnothing) & \bar{\mathfrak{r}}_5 & = & (\varnothing, \{q_0, q_1\}, \{h\}) \\
\mathfrak{r}_6 & = & (\varnothing, \{q_0\}, \{f\}) & \bar{\mathfrak{r}}_6 & = & (\{f\}, \{q_1, q_2\}, \varnothing) \\
\mathfrak{r}_7 & = & (\{f\}, \{q_1\}, \{g\}) & \bar{\mathfrak{r}}_7 & = & (\{g\}, \{q_0, q_2\}, \{f\}) \\
\mathfrak{r}_8 & = & (\{f\}, \{q_1\}, \{h\}) & \bar{\mathfrak{r}}_8 & = & (\{h\}, \{q_0, q_2\}, \{f\})
\end{array}$$

The implemented tool, after applying Reduction Rule 2, will delete 8 out of 16 regions, leaving the minimal regions ($\mathfrak{r}_1$ - $\mathfrak{r}_8$). The set of minimal regions will be the same whether they are defined w.r.t. $\prec$ or $\prec_s$ strict pre-order (see Corollary 6.3.4(1)) as every non-minimal region that can be expressed as a composition of compatible regions can be also expressed as a composition of strongly compatible regions (see Proposition 6.3.3). For example, $\bar{\mathfrak{r}}_3 = \mathfrak{r}_1 \oplus_s \mathfrak{r}_2 = \mathfrak{r}_1 \oplus \mathfrak{r}_7$, where the first two regions are strongly compatible, but the second two regions are only compatible, but not strongly compatible. The algorithm that implements Reduction Rule 3, when applied to this example, would delete two out of four regions based on the set of states $\{q_1\}$ leaving either $\mathfrak{r}_2$ and $\mathfrak{r}_8$ or $\mathfrak{r}_4$ and $\mathfrak{r}_7$. The remaining pairs of companion regions, based on sets of states $\{q_0\}$,$\{q_1\}$ and $\{q_2\}$, will remain as they are essential (having different in-indices or/and out-indices for the shared sets of states; see Corollary 6.4.1). The Reduction Rule 1, the last to be used in Strategy (2,3,1), is not applicable to this example as all the complementary regions of $\mathfrak{r}_1$ - $\mathfrak{r}_8$ were already deleted as non-minimal regions (see the results for $\mathfrak{ts}_{2,2}^{co-loc}$ in Table 6.2).

We will now show the workings of Strategy (2,3,1) on further examples. For the first three examples we will list their sets of non-trivial regions and present the pictures of the step transition systems involved with their saturated nets obtained in the synthesis procedure. Also, we will show the screenshots from WORKCRAFT with the minimised nets for every saturated net. Note that in WORKCRAFT screenshots, the regions' (conditions') names are changed as follows: $\mathfrak{r}_i$ is written as *ri* and $\bar{\mathfrak{r}}_i$ is written as *r_i*.

The first example to test Strategy (2,3,1) on is the ENLST-system in Figure 3.6, on p. 31 (see also the saturated net resulting from its synthesis in Figure 6.5).

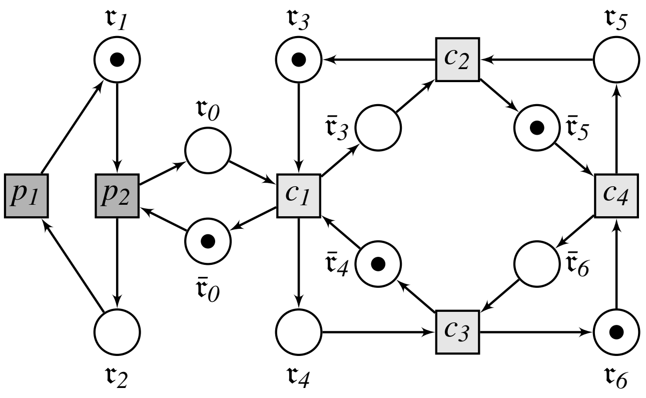


Fig. 6.5 An ENL-system resulting from the synthesis of the ENLST-system in Figure 3.6.

Its non-trivial regions are listed below:

$\mathfrak{r}_0 = (\{p_2\}, \{q_1, q_2, q_4, q_5, q_8, q_9\}), \{c_1\}),$
$\mathfrak{r}_1 = (\{p_1\}, \{q_0, q_2, q_3, q_5, q_7, q_9\}, \{p_2\}),$
$\mathfrak{r}_2 = (\{p_2\}, \{q_1, q_4, q_6, q_8, q_{10}, q_{11}\}, \{p_1\}),$
$\mathfrak{r}_3 = (\{c_2\}, \{q_0, q_1, q_2, q_3, q_4, q_5, q_{10}, q_{11}\}, \{c_1\}),$

$\mathfrak{r}_4 = (\{c_1\},\{q_6,q_7,q_8,q_9\},\{c_3\})$,
$\mathfrak{r}_5 = (\{c_4\},\{q_3,q_4,q_5,q_6,q_7,q_8,q_9,q_{11}\},\{c_2\})$,
$\mathfrak{r}_6 = (\{c_3\},\{q_0,q_1,q_2,q_{10}\},\{c_4\})$,
$\bar{\mathfrak{r}}_0 = (\{c_1\},\{q_0,q_3,q_6,q_7,q_{10},q_{11}\},\{p_2\})$,
$\bar{\mathfrak{r}}_3 = (\{c_1\},\{q_6,q_7,q_8,q_9\},\{c_2\})$,
$\bar{\mathfrak{r}}_4 = (\{c_3\},\{q_0,q_1,q_2,q_3,q_4,q_5,q_{10},q_{11}\},\{c_1\})$,
$\bar{\mathfrak{r}}_5 = (\{c_2\},\{q_0,q_1,q_2,q_{10}\},\{c_4\})$,
$\bar{\mathfrak{r}}_6 = (\{c_4\},\{q_3,q_4,q_5,q_6,q_7,q_8,q_9,q_{11}\},\{c_3\})$.

All these 12 regions are minimal and all its companion regions, associated with various sets of states, are essential (they won't be removed by the application of Reduction Rule 3) as they are 'indexed' by events c_2 or c_3. So, after first two steps of Strategy (2,3,1) our tool reports that there are still 12 regions. The only minimisation of the synthesised net is therefore due to the application of Reduction Rule 1, which can (non-deterministically) remove some of the non-essential complementary regions. Figure 6.6 shows a screenshot from WORKCRAFT of one of many possible outcomes of the last step of Strategy (2,3,1) when applied to this example (note there are 8 conditions there). The ENL-system in Figure 1.1 on p. 4, which generates the ENLST-system in Figure 3.6, on p. 31, and has 7 conditions, can also be returned by the tool for this example. Right now our tool does not contain any algorithms that would allow us to target specific regions to remove/keep, when Reduction Rule 1 is applied.

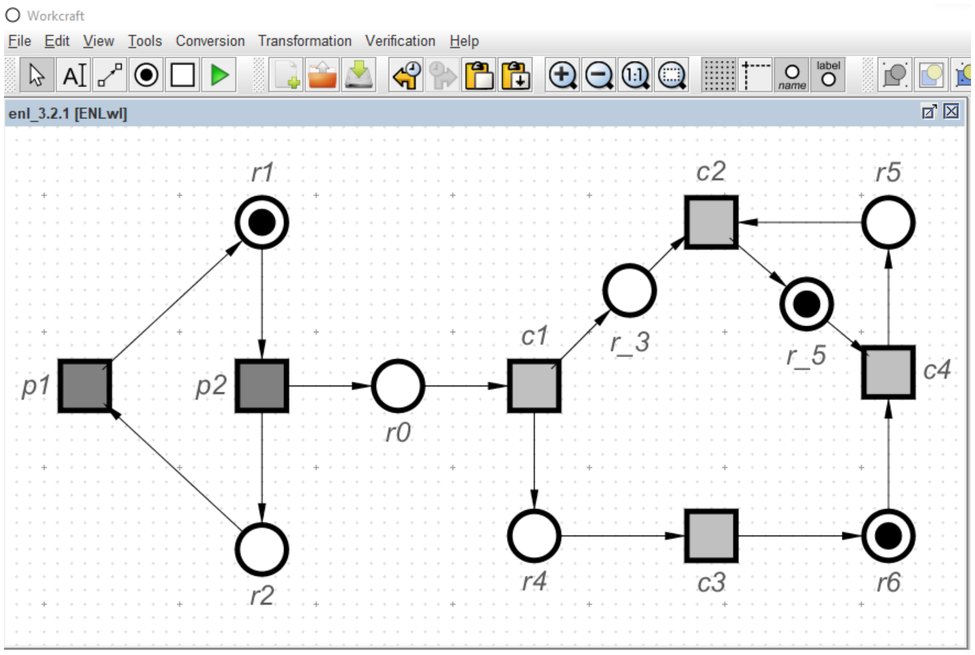

Workcraft
File Edit View Tools Conversion Transformation Verification Help
enl_3.2.1 [ENLwl]
r1
c2
r5
r_3
r_5
c1
p1
p2
c4
r0
r2
r4
c3
r6

Fig. 6.6 Shows a screenshot from WORKCRAFT with a minimised ENL-system of Figure 6.5.

Another interesting example to test Strategy (2,3,1) on is the ENLST-system in Figure 6.7 (see also the saturated net resulting from its synthesis in Figure 6.8). In this example there are no non-minimal regions and no companion regions (all regions are based on different sets of states as this ENLST-system is thin). Therefore after the first two steps of Strategy (2,3,1) we still have all 12 initial non-trivial regions listed below:

$$\begin{aligned}
\mathfrak{r}_0 &= (\{e\},\{q_{14},q_{28},q_{30},q_{15},q_{17},q_{18},q_{23},q_{11},q_1,q_{26},q_{32},q_7,q_{19},q_{25},q_{27},q_{16},q_4,\\
&\quad q_{10},q_{22},q_{29}\}),\{o\}),\\
\mathfrak{r}_1 &= (\{o\},\{q_{28},q_{21},q_8,q_2,q_{31},q_{18},q_{23},q_{11},q_{26},q_{13},q_{32},q_5,q_{20},q_{19},q_{25},q_{27},q_{16},\\
&\quad q_9,q_{24},q_{10},q_{29}\},\{g\}),\\
\mathfrak{r}_2 &= (\{e\},\{q_8,q_2,q_{17},q_{18},q_{11},q_1,q_{32},q_5,q_7,q_{19},q_{25},q_4,q_{10},q_{24}\},\{g,h\}),\\
\mathfrak{r}_3 &= (\{f\},\{q_{21},q_8,q_2,q_0,q_{23},q_{11},q_1,q_{26},q_5,q_7,q_{25},q_{27},q_4,q_{10},q_{22}\},\{g,h\}),\\
\mathfrak{r}_4 &= (\{f\},\{q_{14},q_{21},q_{30},q_8,q_{18},q_{31},q_{17},q_{23},q_{13},q_{32},q_7,q_{12},q_{25},q_{27},q_4,q_{10},q_{24},\\
&\quad q_{22},q_{29}\},\{i\}),\\
\mathfrak{r}_5 &= (\{i\},\{q_{28},q_{30},q_{15},q_{21},q_8,q_{31},q_{17},q_{11},q_{26},q_{32},q_5,q_7,q_{20},q_{12},q_{19},q_{25},q_{27},\\
&\quad q_6,q_{24},q_{22}\},\{h\}),\\
\bar{\mathfrak{r}}_0 &= (\{o\},\{q_3,q_{21},q_8,q_0,q_2,q_{31},q_{13},q_5,q_{20},q_{12},q_9,q_6,q_{24}\},\{e\}),\\
\bar{\mathfrak{r}}_1 &= (\{g\},\{q_3,q_{14},q_{12},q_{30},q_{15},q_{17},q_0,q_1,q_4,q_6,q_{22},q_7\},\{o\}),\\
\bar{\mathfrak{r}}_2 &= (\{g,h\},\{q_3,q_{14},q_{28},q_{30},q_{15},q_{21},q_0,q_{31},q_{23},q_{26},q_{13},q_{20},q_{12},q_{27},q_{16},q_9,\\
&\quad q_6,q_{22},q_{29}\},\{e\}),\\
\bar{\mathfrak{r}}_3 &= (\{g,h\},\{q_3,q_{14},q_{28},q_{30},q_{15},q_{31},q_{17},q_{18},q_{13},q_{32},q_{20},q_{12},q_{19},q_{16},q_9,q_6,\\
&\quad q_{24},q_{29}\},\{f\}),\\
\bar{\mathfrak{r}}_4 &= (\{i\},\{q_3,q_{28},q_{15},q_0,q_2,q_{11},q_1,q_{26},q_5,q_{20},q_{19},q_{16},q_9,q_6\},\{f\}),\\
\bar{\mathfrak{r}}_5 &= (\{h\},\{q_3,q_{14},q_{18},q_2,q_0,q_{23},q_1,q_{13},q_{16},q_4,q_9,q_{10},q_{29}\},\{i\}).
\end{aligned}$$

The only minimalisation of the synthesised net in Figure 6.8 is therefore due to the application of Reduction Rule 1, which can (non-deterministically) remove some of the non-essential complementary regions. Figure 6.9 shows a screenshot from WORKCRAFT of one of many possible outcomes of the last step of Strategy (2,3,1) when applied to this example.

The third example we will consider is an ENLST-system depicted in Figure 3.3(a), on p. 22. The saturated net solution resulting from its synthesis is shown in Figure 3.3(b). Its

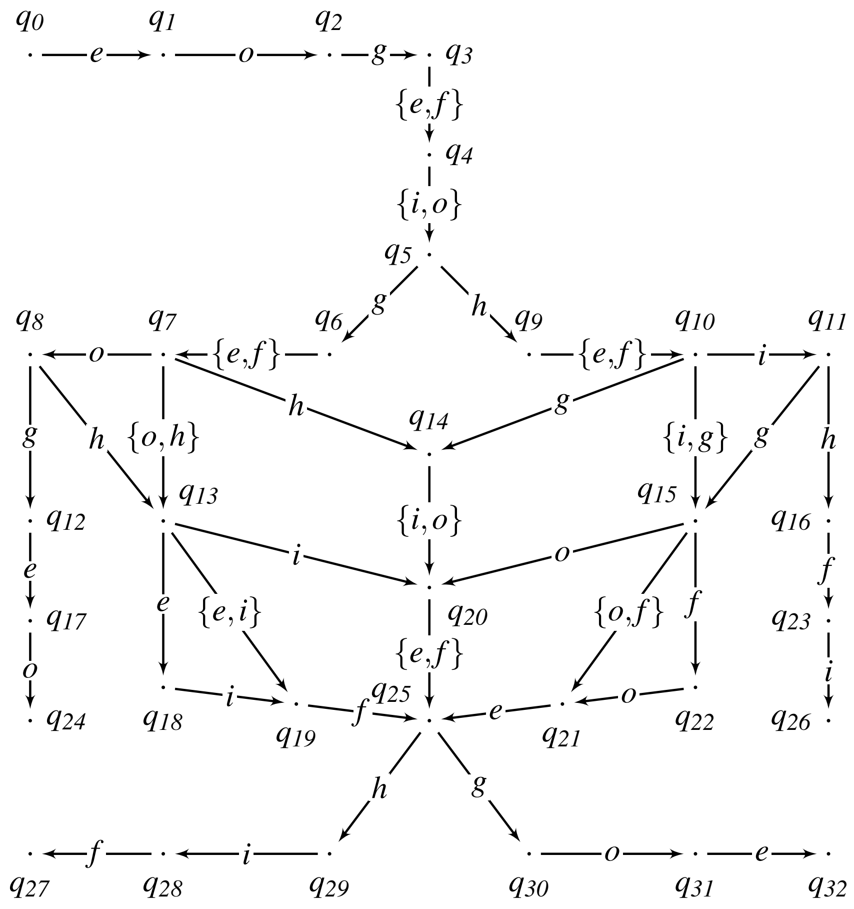


Fig. 6.7 An ENLST-system, where there are three pairs of co-located events with different localities: e and f; o and i, and g and h.

non-trivial regions are:

$$
\begin{aligned}
\mathfrak{r}_0 &= (\{g,h\},\{q_0,q_2,q_4,q_6,q_7\},\{e\}), & \bar{\mathfrak{r}}_0 &= (\{e\},\{q_1,q_3,q_5\},\{g,h\}),\\
\mathfrak{r}_1 &= (\{e\},\{q_1,q_3,q_5,q_6,q_7\},\{g\}), & \bar{\mathfrak{r}}_1 &= (\{g\},\{q_0,q_2,q_4\},\{e\}),\\
\mathfrak{r}_2 &= (\{g,h\},\{q_2,q_4,q_5,q_6\},\{f\}), & \bar{\mathfrak{r}}_2 &= (\{f\},\{q_0,q_1,q_3,q_7\},\{g,h\}),\\
\mathfrak{r}_3 &= (\{f\},\{q_3,q_4,q_5,q_7\},\{h\}), & \bar{\mathfrak{r}}_3 &= (\{h\},\{q_0,q_1,q_2,q_6\},\{f\}),\\
\mathfrak{r}_4 &= (\varnothing,\{q_0,q_1,q_2,q_3,q_4,q_5\},\{h\}), & \bar{\mathfrak{r}}_4 &= (\{h\},\{q_6,q_7\},\varnothing).
\end{aligned}
$$

The Reduction Rule 2 applied to this example would eliminate three non-minimal regions ($\mathfrak{r}_0$, $\mathfrak{r}_1$ and $\mathfrak{r}_4$). There are no companion regions in this ENLST-system (all regions are based on different sets of states as this ENLST-system is thin). Therefore, after the first two steps of Strategy (2,3,1) we will have seven regions. Figure 6.10 shows a screenshot from WORKCRAFT of one of many possible outcomes of the Strategy (2,3,1), when applied to this

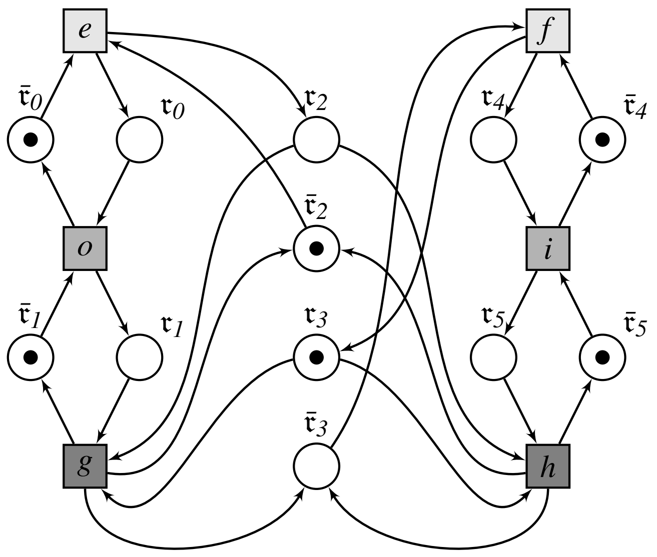


Fig. 6.8 The ENL-system resulting from the synthesis of the ENLST-system in Figure 6.7.

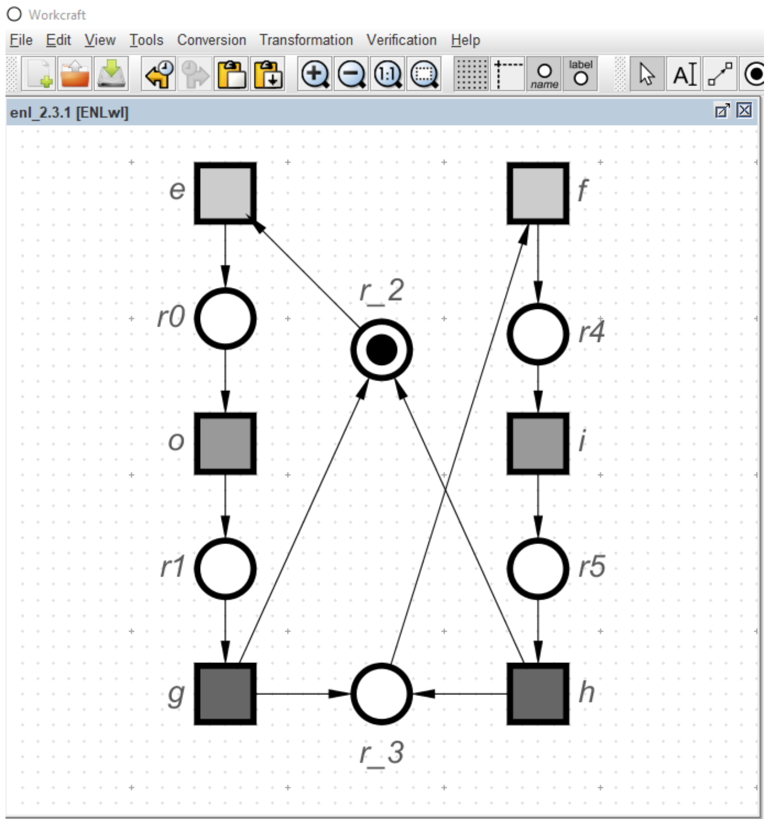

Workcraft
File Edit View Tools Conversion Transformation Verification Help
enl_2.3.1 [ENLwl]
e
f
r0
r_2
r4
o
i
r1
r5
g
h
r_3

Fig. 6.9 Shows a screenshot from WORKCRAFT with a minimised ENL-system of Figure 6.8.

example. We can see that Reduction Rule 1 deleted further two regions, so the minimised net has five regions/conditions. It can be easily verified that region $\bar{\mathfrak{r}}_4 = (\{h\}, \{q_6, q_7\}, \varnothing)$ is still redundant. This is an example that Strategy (2,3,1) might not delete all the redundant regions/conditions.

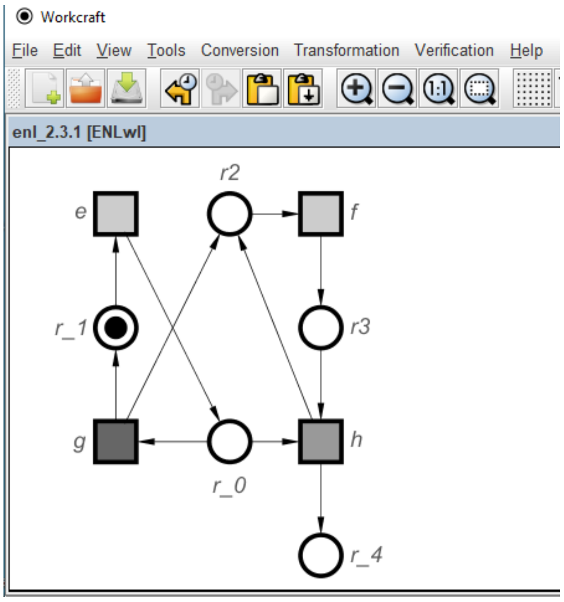

Workcraft
File Edit View Tools Conversion Transformation Verification Help
enl_2.3.1 [ENLwl]
r2
e
f
r_1
r3
g
h
r_0
r_4

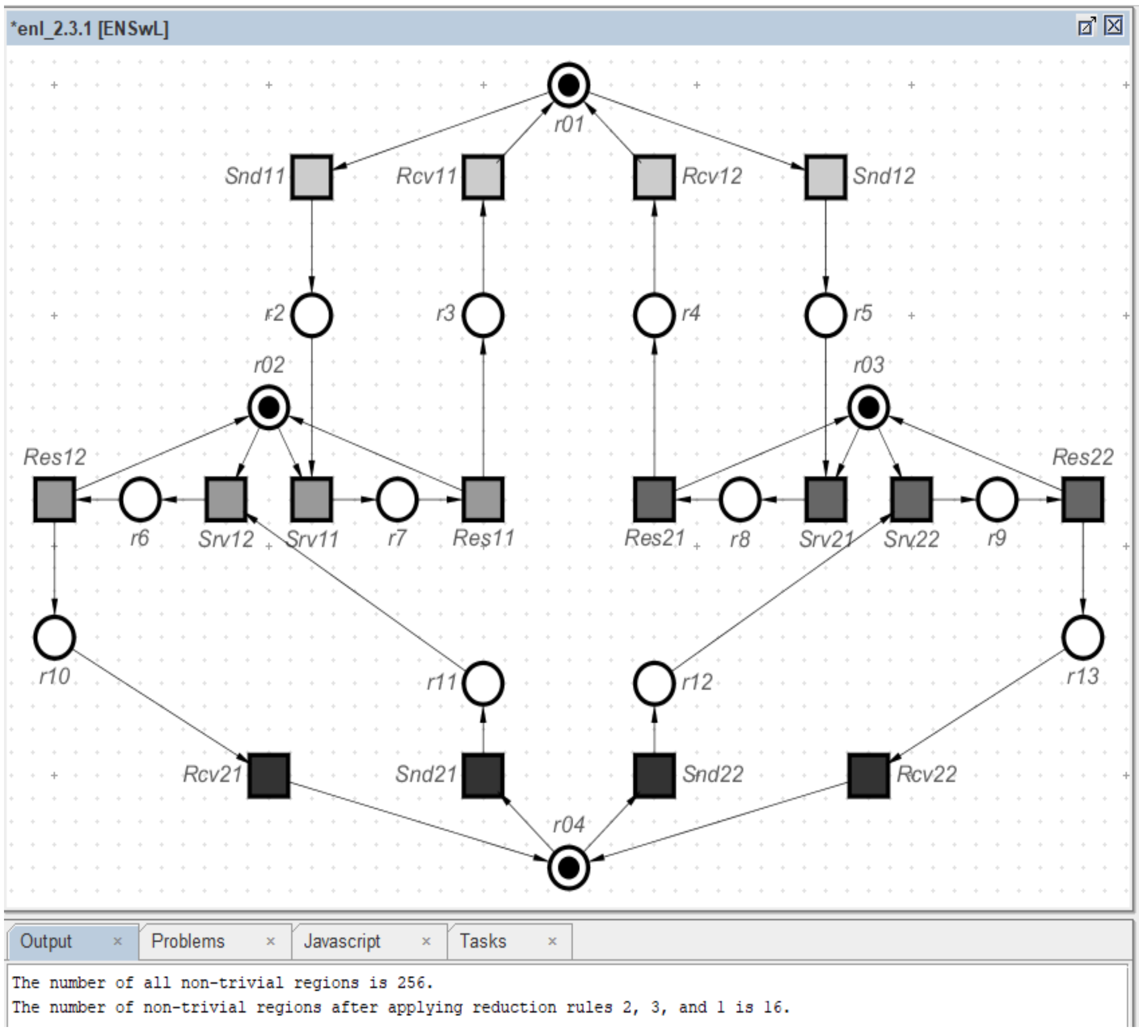
Fig. 6.10 Shows a screenshot from WORKCRAFT with a minimised ENL-system of Figure 3.3(b).

The last example of this section uses a very large ENLST-system ($\mathfrak{ts}_{2,2}^{ser-cl}$) and therefore we only show the initial ENL-system we used to generate it, as well as the minimised net after Strategy (2,3,1) was applied to the saturated ENL-system that resulted from the synthesis procedure. The initial ENL-system, depicted in Figure 4.10, on p. 46, models interactions between two servers and two clients with each client and each sever residing in a different locality. The minimising algorithms of our tool reduced the number of the non-trivial regions from 256, in the saturated ENL-system, to 16, in its minimised version (see Figure 6.11).

6.5 Algorithms for the minimisation

The first algorithm considered, Algorithm 11, is the algorithm that implements Reduction Rule 2. From Definition 6.2.1 we have that a non-trivial region $\mathfrak{r} = (in, r, out)$ is <u>not</u> a minimal region iff there exists a region $\widehat{\mathfrak{r}} = (\widehat{in}, \widehat{r}, \widehat{out}) \in \mathfrak{R}_{\mathfrak{ts}}$ such that $\widehat{\mathfrak{r}} \prec \mathfrak{r}$. Hence we have the following implication:

$$\text{A region } \mathfrak{r} \text{ is } \underline{\text{not}} \text{ a minimal region} \Longrightarrow \exists \widehat{\mathfrak{r}} \in \mathfrak{R}_{\mathfrak{ts}} : \widehat{r} \subset r.$$



```
The number of all non-trivial regions is 256.
The number of non-trivial regions after applying reduction rules 2, 3, and 1 is 16.
```

Fig. 6.11 Shows a screenshot from WORKCRAFT with a minimised ENL-system of Figure 4.10.

Writing the above in an alternative form we have:

$$\forall \widehat{\mathfrak{r}} \in \mathfrak{R}_{\mathfrak{ts}} : \widehat{r} \not\subset r \implies \mathfrak{r} \text{ is a minimal region.}$$

This test will identify <u>some</u> of the minimal regions. However, there might be some minimal regions that do not satisfy the antecedent of this implication. For example, region $\mathfrak{r}_3 = (\{e\}, \{q_1, q_2\}, \varnothing)$ of the ENLST-system in Figure 6.2(a) is minimal as it is not a composition of any other two non-trivial regions, but its set of states $\{q_1, q_2\}$ is a superset of the sets of states of the two regions: $\mathfrak{r}_5$ and $\mathfrak{r}_6$. So, this test is not offering an easy to implement condition for identifying minimal regions. Therefore, instead of calculating minimal regions directly, Algorithm 11 calculates the set of non-minimal regions ($\mathfrak{R}_{\mathfrak{ts}}^{\oplus} = \mathfrak{R}_{\mathfrak{ts}}^{\oplus_s}$) first and then computes the difference $\mathfrak{R}_{\mathfrak{ts}} \setminus \mathfrak{R}_{\mathfrak{ts}}^{\oplus}$.

Algorithm 11 calls Algorithms 12 and 13. Algorithm 12 is called to decide whether two regions, $\mathfrak{r}$ and $\mathfrak{r}'$, are compatible. It is called twice in Algorithm 11, for pairs $(\mathfrak{r},\mathfrak{r}')$ and $(\mathfrak{r}',\mathfrak{r})$. Algorithm 13 is called in Algorithm 11 to compute the composition $\mathfrak{r} \oplus \mathfrak{r}'$, where there is a need to calculate the set H for $\mathfrak{r} \oplus \mathfrak{r}'$.

Algorithm 11 Using reduction rule 2 for a given ST-system $\mathfrak{ts} = (Q,A,q_0)$ to delete all non-minimal regions from $\mathfrak{R}_{\mathfrak{ts}}$ (see Reduction Rule 2).

```
 1: function APPLY_REDUCTION_RULE_2($\mathfrak{R}_{\mathfrak{ts}}$)
 2:     initialise $\mathfrak{R}_{\mathfrak{ts}}^{min}$ to $\varnothing$          ▷ $\mathfrak{R}_{\mathfrak{ts}}^{min}$ is the set of minimal regions of $\mathfrak{ts}$
 3:     initialise $\mathfrak{R}_{\mathfrak{ts}}^{\oplus}$ to $\varnothing$          ▷ $\mathfrak{R}_{\mathfrak{ts}}^{\oplus}$ is the set of non-minimal regions of $\mathfrak{ts}$
 4:     for every $\mathfrak{r} = (in,r,out) \in \mathfrak{R}_{\mathfrak{ts}}$ do
 5:         for every $\mathfrak{r}' = (in',r',out') \in \mathfrak{R}_{\mathfrak{ts}}$ do
 6:             if $\mathfrak{r}$ and $\mathfrak{r}'$ are compatible regions then     ▷ where $\mathfrak{r} \neq \mathfrak{r}'$, see Algorithm 12
 7:                 calculate $H$                              ▷ see Algorithm 13
 8:                 calculate $\mathfrak{r} \oplus \mathfrak{r}' = (in \cup in' \setminus H, r \cup r', out \cup out' \setminus H)$
 9:                 add $\mathfrak{r} \oplus \mathfrak{r}'$ to $\mathfrak{R}_{\mathfrak{ts}}^{\oplus}$
10:     $\mathfrak{R}_{\mathfrak{ts}}^{min} = \mathfrak{R}_{\mathfrak{ts}} \setminus \mathfrak{R}_{\mathfrak{ts}}^{\oplus}$
11:     return $\mathfrak{R}_{\mathfrak{ts}}^{min}$
```

Before we present Algorithm 12, we need to introduce additional notations for two specific sets of transitions in an ENLST-system $\mathfrak{ts} = (Q,A,q_0)$. For a region $\mathfrak{r} = (in,r,out)$ of $\mathfrak{ts}$ we denote:

$$\begin{aligned} U_e^{out} &= \{(q,U,q') \in A \mid e \in out \cap U\}; \\ U_e^{in} &= \{(q,U,q') \in A \mid e \in in \cap U\}. \end{aligned}$$

Algorithm 12 Checking whether region $\mathfrak{r} = (in,r,out) \in \mathfrak{R}_{\mathfrak{ts}}$ is compatible with region $\mathfrak{r}' = (in',r',out') \in \mathfrak{R}_{\mathfrak{ts}}$ ($\mathfrak{ts} = (Q,A,q_0)$).

```
 1: function IS_FIRST_COMPATIBLE_WITH_SECOND($\mathfrak{r}$,$\mathfrak{r}'$)
 2:     initialise result to false
 3:     if $r \cap r' = \varnothing$ then
 4:         initialise $count_1$ to 0
 5:         for every $e \in out$ do
 6:             initialise $U_e^{out}$ to $\varnothing$
 7:             initialise $hold_1$ and $hold_2$ to false
 8:             add all the transitions $(q,U,q') \in A$ such that $e \in U$ to $U_e^{out}$
 9:             if $q' \in r'$ for all transitions from $U_e^{out}$ then
10:                 $hold_1$ = true
```

```
11:             if $q' \notin r'$ for all transitions from $U_e^{out}$ then
12:                 $hold_2$ = true
13:             if $hold_1 \vee hold_2$ then
14:                 ++$count_1$
15:         if $count_1 == |out|$ then
16:             initialise $count_2$ to 0
17:             for every $e \in in$ do
18:                 initialise $U_e^{in}$ to $\varnothing$
19:                 initialise $hold_3$ and $hold_4$ to false
20:                 add all the transitions $(q,U,q') \in A$ such that $e \in U$ to $U_e^{in}$
21:                 if $q \in r'$ for all transitions from $U_e^{in}$ then
22:                     $hold_3$ = true
23:                 if $q \notin r'$ for all transitions from $U_e^{in}$ then
24:                     $hold_4$ = true
25:                 if $hold_3 \vee hold_4$ then
26:                     ++$count_2$
27:             if $count_2 == |in|$ then
28:                 $result$ = true
29:     return $result$
```

Before we present Algorithm 13, which calculates the set H that is required to compute non-minimal regions, we need to introduce additional notations for two specific sets of events. For a region $\mathfrak{r} = (in, r, out)$ of an ENLST-system $\mathfrak{ts} = (Q, A, q_0)$ we denote:

$$\begin{aligned} {}^{\blacktriangledown}r &= \{e \in E \mid \forall (q,U,q') \in A : \ (e \in U \ \Rightarrow \ q' \in r)\}; \\ r^{\blacktriangledown} &= \{e \in E \mid \forall (q,U,q') \in A : \ (e \in U \ \Rightarrow \ q \in r)\}. \end{aligned}$$

Algorithm 13 Calculating H, a set of events that belong only to steps labelling transitions hidden in the set of states $r \cup r'$ of the composition region $\mathfrak{r} \oplus \mathfrak{r}'$ of two compatible regions $r = (in, r, out)$ and $r' = (in', r', out')$ of $\mathfrak{ts} = (Q, A, q_0)$.

```
1: function CALCULATE_H($r$,$r'$)
2:     initialise $H$, ${}^{\blacktriangledown}r$, $r^{\blacktriangledown}$, ${}^{\blacktriangledown}r'$, and $r'^{\blacktriangledown}$ to $\varnothing$
3:     for every $e \in E$ do
4:         initialise $check_1, check_2, check_3, check_4$ and $belongs$ to true
5:         for every transition $(q,U,q') \in A$ do
6:             if $e \in U$ then
7:                 if $q' \notin r$ then
8:                     $check_1$ = false
```

```
9:                 if $q \notin r$ then
10:                    $check_2$ = false
11:                if $q' \notin r'$ then
12:                    $check_3$ = false
13:                if $q \notin r'$ then
14:                    $check_4$ = false
15:            else
16:                $belongs$ = false
17:        if $belongs$ then
18:            if $check_1$ then
19:                add $e$ to ${}^\blacktriangledown r$
20:            if $check_2$ then
21:                add $e$ to $r^\blacktriangledown$
22:            if $check_3$ then
23:                add $e$ to ${}^\blacktriangledown r'$
24:            if $check_4$ then
25:                add $e$ to $r'^\blacktriangledown$
26:    calculate $H = {}^\blacktriangledown r \cap r'^\blacktriangledown \cup {}^\blacktriangledown r' \cap r^\blacktriangledown$
27:    return $H$
```

Before we present Algorithm 14 and Algorithm 15, which implement Reduction Rule 3, we need to introduce additional notations. For an ENLST-system $\mathfrak{ts} = (Q, A, q_0)$ on E we denote:

$$\begin{aligned} \mathfrak{R}^r_{\mathfrak{ts}} &= \{\mathfrak{r} \in \mathfrak{R}^{min}_{\mathfrak{ts}} \mid \exists\, in, out \subseteq E : \ \mathfrak{r} = (in, r, out)\}; \\ R &= \{r \subset Q \mid \exists\, \mathfrak{r} \in \mathfrak{R}^{min}_{\mathfrak{ts}} : \ \mathfrak{r} = (in, r, out)\}; \\ \mathfrak{R}^{comp}_{\mathfrak{ts}} &= \{\mathfrak{R}^r_{\mathfrak{ts}} \subset \mathfrak{R}^{min}_{\mathfrak{ts}} \mid r \in R\}. \end{aligned}$$

In the above, $\mathfrak{R}^r_{\mathfrak{ts}}$ is a set of regions based on the set of states r (note that these regions are selected from the set of minimal regions here rather than from $\mathfrak{R}_{\mathfrak{ts}}$); R comprises all possible sets of states of minimal regions of $\mathfrak{ts}$; and $\mathfrak{R}^{comp}_{\mathfrak{ts}}$ is a set of sets of companion regions based on different sets of states.

Algorithm 14 Using reduction rule 3 to delete redundant companion regions that might be present among the minimal regions $\mathfrak{R}_{\mathfrak{ts}}^{min}$ (see Reduction Rule 3).

```
1: function APPLY_REDUCTION_RULE_3($\mathfrak{R}_{\mathfrak{ts}}^{min}$)
2:     $\mathfrak{R}_{\mathfrak{ts}}^{comp}$ = get the set of sets of companion regions from $\mathfrak{R}_{\mathfrak{ts}}^{min}$     ▷ see Algorithm 15
3:     for every set of companion regions $\mathfrak{R}_{\mathfrak{ts}}^{r} \in \mathfrak{R}_{\mathfrak{ts}}^{comp}$ do
4:         for every $\mathfrak{r} = (in, r, out) \in \mathfrak{R}_{\mathfrak{ts}}^{r}$ do
5:             initialise $in_{set}$ and $out_{set}$ to $\varnothing$
6:             for every $\mathfrak{r}' = (in', r, out') \in \mathfrak{R}_{\mathfrak{ts}}^{r}$ do     ▷ where $\mathfrak{r}' \neq \mathfrak{r}$
7:                 add elements of $in'$ to $in_{set}$
8:                 add elements of $out'$ to $out_{set}$
9:             if $in \subseteq in_{set}$ and $out \subseteq out_{set}$ then
10:                delete $\mathfrak{r}$ from $\mathfrak{R}_{\mathfrak{ts}}^{r}$ and from $\mathfrak{R}_{\mathfrak{ts}}^{min}$
11:    return $\mathfrak{R}_{\mathfrak{ts}}^{min}$     ▷ the returned set is a subset of the original set $\mathfrak{R}_{\mathfrak{ts}}^{min}$ with redundant
       companion regions deleted
```

Algorithm 15 Identifying sets of companion regions in the set of minimal regions $\mathfrak{R}_{\mathfrak{ts}}^{min}$ if there are any (see Reduction Rule 3).

```
1: function GET_COMPANION_REGIONS($\mathfrak{R}_{\mathfrak{ts}}^{min}$)
2:     initialise $\mathfrak{R}_{\mathfrak{ts}}^{comp}$ to $\varnothing$ ▷ $\mathfrak{R}_{\mathfrak{ts}}^{comp}$ is a set of sets of companion regions, each based on a
       different set of states and having at least 2 elements
3:     for every $\mathfrak{r} = (in, r, out) \in \mathfrak{R}_{\mathfrak{ts}}^{min}$ do
4:         initialise $\mathfrak{R}_{\mathfrak{ts}}^{r}$ to $\varnothing$     ▷ $\mathfrak{R}_{\mathfrak{ts}}^{r}$ is a set of companion regions based on $r$
5:         for every $\mathfrak{r}' = (in', r', out') \in \mathfrak{R}_{\mathfrak{ts}}^{min}$ do     ▷ where $\mathfrak{r}' \neq \mathfrak{r}$
6:             if $r' == r$ then
7:                 add $\mathfrak{r}'$ to $\mathfrak{R}_{\mathfrak{ts}}^{r}$
8:         if $\mathfrak{R}_{\mathfrak{ts}}^{r} \neq \varnothing$ then
9:             add $\mathfrak{r}$ to $\mathfrak{R}_{\mathfrak{ts}}^{r}$
10:            add $\mathfrak{R}_{\mathfrak{ts}}^{r}$ to $\mathfrak{R}_{\mathfrak{ts}}^{comp}$     ▷ duplicate elements will be rejected by $\mathfrak{R}_{\mathfrak{ts}}^{comp}$
11:    return $\mathfrak{R}_{\mathfrak{ts}}^{comp}$
```

The next two algorithms considered, Algorithm 16 and Algorithm 17, are the algorithms that implement Reduction Rule 1 to delete any redundant complementary regions that might be present among the minimal regions $\mathfrak{R}_{\mathfrak{ts}}^{min}$ after deleting redundant companion regions, and to check whether the synthesised net $\mathfrak{enl}_{\mathfrak{ts}}^{\widehat{=}}$ is still an ENL$_{\widehat{=}}$-system after removal of some redundant regions.

Algorithm 16 Using reduction rule 1 to delete any redundant complementary regions that might be present among the minimal regions $\mathfrak{R}_{\mathfrak{ts}}^{min}$ after deleting redundant companion regions.

```
 1: function APPLY_REDUCTION_RULE_1($\mathfrak{R}_{\mathfrak{ts}}^{min}$,$\mathfrak{R}_{\mathfrak{ts}}$)    ▷ $\mathfrak{R}_{\mathfrak{ts}}^{min}$ here is a subset of the original set $\mathfrak{R}_{\mathfrak{ts}}^{min}$ with redundant companion regions deleted
 2:     initialise $\mathfrak{R}_{\mathfrak{ts}}^{(\mathfrak{r},\bar{\mathfrak{r}})}$ to empty map    ▷ $\mathfrak{R}_{\mathfrak{ts}}^{(\mathfrak{r},\bar{\mathfrak{r}})}$ is a map containing pairs of complementary regions of $\mathfrak{R}_{\mathfrak{ts}}$
 3:     fill in $\mathfrak{R}_{\mathfrak{ts}}^{(\mathfrak{r},\bar{\mathfrak{r}})}$ with pairs of complementary regions of $\mathfrak{R}_{\mathfrak{ts}}$
 4:     for every pair $(\mathfrak{r},\bar{\mathfrak{r}}) \in \mathfrak{R}_{\mathfrak{ts}}^{(\mathfrak{r},\bar{\mathfrak{r}})}$ do
 5:         if $\mathfrak{r} \in \mathfrak{R}_{\mathfrak{ts}}^{min}$ and $\bar{\mathfrak{r}} \in \mathfrak{R}_{\mathfrak{ts}}^{min}$ then
 6:             delete $\bar{\mathfrak{r}}$ from $\mathfrak{R}_{\mathfrak{ts}}^{min}$
 7:             if $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ based on $\mathfrak{R}_{\mathfrak{ts}}^{min}$ is not an ENL$_{\hat{=}}$-system then    ▷ see Algorithm 17
 8:                 add $\bar{\mathfrak{r}}$ to $\mathfrak{R}_{\mathfrak{ts}}^{min}$
 9:                 delete $\mathfrak{r}$ from $\mathfrak{R}_{\mathfrak{ts}}^{min}$
10:                 if $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ based on $\mathfrak{R}_{\mathfrak{ts}}^{min}$ is not an ENL$_{\hat{=}}$-system then
11:                     add $\mathfrak{r}$ to $\mathfrak{R}_{\mathfrak{ts}}^{min}$
12:     return $\mathfrak{R}_{\mathfrak{ts}}^{min}$    ▷ $\mathfrak{R}_{\mathfrak{ts}}^{min}$ here is a subset of the original set $\mathfrak{R}_{\mathfrak{ts}}^{min}$ with redundant companion and complementary regions deleted
```

Algorithm 17 Checking whether the synthesised net $\mathfrak{enl}_{\mathfrak{ts}}^{\hat{=}}$ is still an ENL$_{\hat{=}}$-system after removal of some redundant regions.

```
 1: function IS_ENL$_{\hat{=}}$_SYSTEM($\mathfrak{R}_{\mathfrak{ts}}$)    ▷ the argument is $\mathfrak{R}_{\mathfrak{ts}}$ or its subset
 2:     initialise result to true
 3:     for every $e \in E$ do
 4:         initialise $^{\circ}e$ and $e^{\circ}$ to $\varnothing$
 5:         for every $\mathfrak{r} = (in, r, out) \in \mathfrak{R}_{\mathfrak{ts}}$ do
 6:             if $e \in out$ then
 7:                 add $\mathfrak{r}$ to $^{\circ}e$
 8:             if $e \in in$ then
 9:                 add $\mathfrak{r}$ to $e^{\circ}$
10:         if $^{\circ}e = \varnothing \vee e^{\circ} = \varnothing$ then
11:             $result = false$
12:             break
13:     return result
```

6.6 Results of the experiments

Table 6.3 below shows the comparison between the efficiency of Strategy (2,3,1) and Strategy (3,2,1). For this comparison we use the ST-systems from the $\mathfrak{ts}^{co-loc}$ class of ST-systems (see pp. 83-84 for the description of these systems). The times in columns 7 and 8 include the time of verifying with Method II, whether the saturated net is a solution to the synthesis problem (see col. 6), which in turn includes the time of executing Algorithm 1 to derive all non-trivial regions for each of the selected ST-systems (see col. 4). Overall, the results in Table 6.3 show that the times for Strategy (2,3,1) are worse than the times for Strategy (3,2,1), but this is probably a consequence of Strategy (2,3,1) taking extra time to remove more regions. As a result, Strategy (2,3,1) is still better, because it has significantly better effectiveness in region removal (see Table 6.2) and time increases are not too big (no greater than 10 times).

Table 6.3 Shows: the execution time taken to derive all non-trivial regions of selected ST-systems (col. 4); the number of all non-trivial regions (col. 5); the time needed to verify, with Method II, whether the saturated net is a solution to the synthesis problem (col. 6); and the time needed to minimise the net following Strategy (2,3,1) (col. 7) and Strategy (3,2,1) (col. 8).

$\mathfrak{ts}^{co-loc}$	$\|\mathbf{Q}\|$	$\|\mathbf{E}\|$	Extracting $\mathfrak{R}_{\mathfrak{ts}}$	$\|\mathfrak{R}_{\mathfrak{ts}}\|$	Verifying $\mathfrak{enl}^{\hat{}}_{\mathfrak{ts}}$	Strategy (2,3,1)	Strategy (3,2,1)
$\mathfrak{ts}^{co-loc}_{2,2}$	3	4	5.5 ms	16	14.3 ms	17.3 ms	16.7 ms
$\mathfrak{ts}^{co-loc}_{2,3}$	4	6	8.1 ms	52	20.8 ms	35.3 ms	33.1 ms
$\mathfrak{ts}^{co-loc}_{2,4}$	5	8	13.3 ms	160	34.3 ms	121.4 ms	56.6 ms
$\mathfrak{ts}^{co-loc}_{2,5}$	6	10	51.1 ms	484	115.9 ms	911.4 ms	184.3 ms
$\mathfrak{ts}^{co-loc}_{3,2}$	3	6	6.4 ms	30	17.2 ms	30.5 ms	28.0 ms
$\mathfrak{ts}^{co-loc}_{3,3}$	4	9	12.6 ms	126	36.0 ms	78.2 ms	38.4 ms
$\mathfrak{ts}^{co-loc}_{3,4}$	5	12	51.2 ms	510	107.4 ms	1208.4 ms	257.4 ms
$\mathfrak{ts}^{co-loc}_{3,5}$	6	15	298.6 ms	2046	1196.8 ms	30968.4 ms	3824.2 ms
$\mathfrak{ts}^{co-loc}_{4,2}$	3	8	7.6 ms	48	20.4 ms	35.3 ms	32.1 ms
$\mathfrak{ts}^{co-loc}_{4,3}$	4	12	8.8 ms	248	48.6 ms	260.2 ms	113.8 ms
$\mathfrak{ts}^{co-loc}_{4,4}$	5	16	86.2 ms	1248	419.9 ms	10966.7 ms	2593.9 ms
$\mathfrak{ts}^{co-loc}_{4,5}$	6	20	2267.4 ms	6248	9227.9 ms	748713.5 ms	75694.3 ms

Table 6.4 below shows the efficiency and effectiveness of strategy (2,3,1) by using ST-systems generated by nets modelling server-client systems like the net in Figure 4.10 on p. 46 ($\mathfrak{ts}^{ser-cl}$ ST-systems). Column 7 shows the time needed to minimise the net following Strategy (2,3,1), which includes the time for extraction of all non-trivial regions (see col. 4), and the time to verify, whether the saturated net is a solution to the synthesis problem (see col. 6). Note that the times in column 6 include the times in column 4. Column 8 shows the number of the remaining regions after each step of the Strategy (2,3,1). Considering the growing sizes and complexities of the ST-systems used in this experiment, we found the results encouraging.

Table 6.4 Shows: the execution time taken to derive non-trivial regions of selected ST-systems (col. 4); the number of all non-trivial regions (col. 5); the time needed to verify, with Method II, whether the saturated net is a solution to the synthesis problem (col. 6); the time needed to minimise the net following Strategy (2,3,1) (col. 7); and the number of the remaining regions after each step of the Strategy (2,3,1) (col. 8).

$\mathfrak{ts}^{ser-cl}$	$\lvert\mathbf{Q}\rvert$	$\lvert\mathbf{E}\rvert$	Extracting $\mathfrak{R}_{\mathfrak{ts}}$	$\lvert\mathfrak{R}_{\mathfrak{ts}}\rvert$	Verifying $\mathfrak{enl}^{\widehat{}}_{\mathfrak{ts}}$	Strategy (2,3,1)	Remaining regions
$\mathfrak{ts}^{ser-cl}_{1,2}$	15	8	0.0114 s	30	0.0402 s	0.0464 s	9-9-9
$\mathfrak{ts}^{ser-cl}_{1,3}$	54	12	0.2724 s	50	0.3675 s	0.3721 s	13-13-13
$\mathfrak{ts}^{ser-cl}_{2,2}$	47	16	0.9029 s	256	1.1693 s	1.4993 s	16-16-16
$\mathfrak{ts}^{ser-cl}_{2,3}$	176	20	24.0480 s	276	24.8932 s	27.5769 s	20-20-20
$\mathfrak{ts}^{ser-cl}_{3,2}$	97	24	122.4600 s	2050	127.7800 s	173.8000 s	23-23-23
$\mathfrak{ts}^{ser-cl}_{3,3}$	1446	36	68806.3500 s	3094	68972.3000 s	73123.6500 s	35-35-35
$\mathfrak{ts}^{ser-cl}_{4,2}$	165	32	61182.4100 s	16388	61427.6900 s	66330.6800 s	30-30-30

Chapter 7

Case studies

This chapter discusses the selected case studies that have been used throughout our research to test and check the correctness and performance of the implemented algorithms in order to evaluate and assess the capability of our tool.

The initial pool of case studies and the taxonomy used for them was described in [2]. This pool was growing as the research progressed and more interesting examples were identified. We describe the taxonomy in the next section.

7.1 Taxonomy of case studies

In the course of our research we considered many examples of Petri nets and step transition systems. The analysis usually started by looking at an example in its net form: ENL-system form. However, the synthesis problems for ENL-systems, the problems for which we designed our algorithmic solutions, take step transition systems as inputs. Therefore, the classification of case studies that was initially done for ENL-systems is presented in Table 7.1 from the step transition systems' point of view.

The classification takes into consideration the following characteristics of an ST-system[6]:

1. The co-location relation $\simeq$ used for its events, which can be defined in three possible ways:

   (a) All events of the ST-system have different localities:

$$\simeq = \bigcup_{e \in E} \{e\} \times \{e\}.$$

[6]Definition 2.1.1 of a step transition system does not include a co-location relation, but it is important in all our synthesis procedures.

(b) All events of the ST-system share one locality:

$$\hat{=} = E \times E.$$

(c) There are several localities into which the events of the ST-system are partitioned into, with at least one locality containing more than one event:

$$\hat{=} = \bigcup_{i=1}^{k} E_i \times E_i,$$

where $E = \biguplus_{i=1}^{k} E_i$ and $1 < k < |E|$ and for every i, $1 \leqslant i \leqslant k$, E_i denotes a subset of events from E that share one locality.

2. The conflicts between events from E, *i.e.*, whether the ST-system is conflict-free or not (see the definition of conflicts in an ST-system on p. 10).

3. The structure of the ST-system, *i.e.*, whether it is thin or not (see the definition of thin ST-systems on p. 10).

The table below shows the classification of the selected ST-systems used in our research.

Table 7.1 Shows the classification of the selected examples of ST-systems that have been used throughout our research as case studies.

$\mathfrak{ts}$	Figure	$\mathfrak{ts}$ is thin	$\mathfrak{ts}$ has conflict	$\bumpeq = \bigcup_{e \in E} \{e\} \times \{e\}$	$\bumpeq = E \times E$	$\bumpeq = \bigcup_{i=1}^{k} E_i \times E_i$
$\mathfrak{ts}_1$	Figure 3.5	NO	YES	NO	YES	NO
$\mathfrak{ts}_2$	Figure 3.3	YES	YES	NO	NO	YES
$\mathfrak{ts}_3$	Figure 3.1	YES	NO	NO	NO	YES
$\mathfrak{ts}_4$	Figure 3.6	NO	NO	NO	NO	YES
$\mathfrak{ts}_5$	The ST-system generated by the ENL-system in Figure 3.7	YES	NO	NO	NO	YES
$\mathfrak{ts}_6$	Figure 5.1(a)	NO	YES	NO	NO	YES
$\mathfrak{ts}_7$	Figure 5.2(a)	YES	YES	NO	NO	YES
$\mathfrak{ts}_8$	Figure 5.6	YES	YES	NO	NO	YES
$\mathfrak{ts}_9$	The ST-system generated by the ENL-system in Figure 4.10	YES	YES	NO	NO	YES
$\mathfrak{ts}_{i,j}$	Extended ST-systems of $\mathfrak{ts}_5$	YES	NO	NO	NO	YES
$\mathfrak{ts}_{i,j}^{co-loc}$	Extended ST-systems of the ST-system in Figure 2.1(a)	NO	NO	NO	YES	NO
$\mathfrak{ts}_{i,j}^{ser-cl}$	Extended ST-systems of $\mathfrak{ts}_9$	YES	YES	NO	NO	YES

7.2 Benchmarks and the strategy for experiments

In order to evaluate our tool to make sure that all the involved algorithms work and derive the required results we used the following strategy for testing and experiments. First, we identified small examples of ENL-systems to be used as case studies. They were chosen specifically to represent different categories of ENL-systems, and were used for testing and checking the functional correctness of the algorithms. Then we extended some of these examples to build benchmarks of ENL-systems for testing the performance efficiency of the developed algorithms.

7.2.1 Checking the correctness of algorithms

Checking the correctness of algorithms is essential to prove that our tool is reliable and delivers the expected results. So, we used the small examples as input into the tool to perform tests and then compared the obtained results with the hand-computed results.

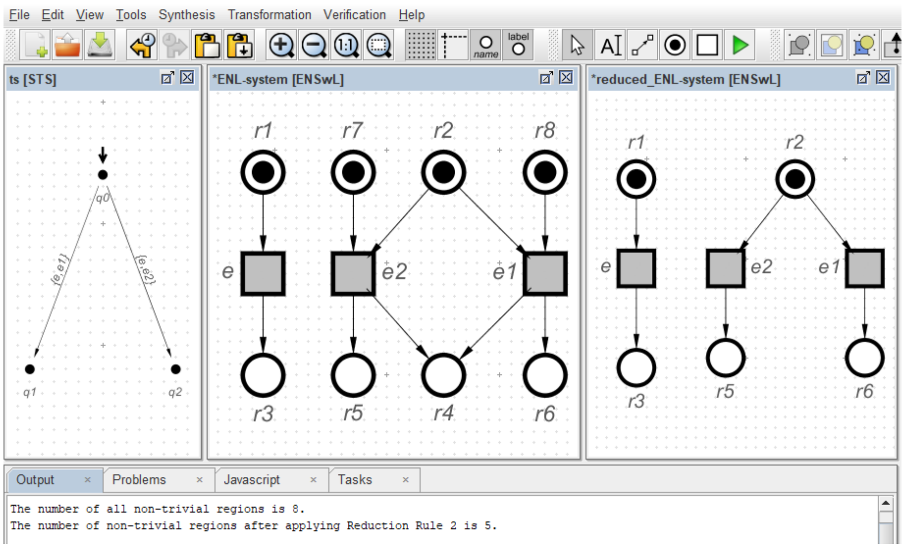
Algorithm 1 was the first algorithm we tested to check whether it computes correctly all non-trivial regions as expected (see Chapter 3). Figure 3.4 on p. 30, for example, shows the screenshot from WORKCRAFT with the extracted non-trivial regions of the ENLST-system $\mathfrak{ts}_2$ in Figure 3.3(a), on p. 22, which confirmed that Algorithm 1 generated correctly all the expected regions for $\mathfrak{ts}_2$. We tested Algorithm 1 using all the examples in our pool of examples, but $\mathfrak{ts}_2$ was especially important here as it was a counterexample showing us that the optimisation in Algorithm 1 based on ignoring thick transitions cannot be applied to all thin ST-systems ($\mathfrak{ts}_2$ is thin).

Next, we tested the implementation of Method I and Method II to check the correctness of the two approaches for the synthesis procedure (see Chapter 4). Method I involves two algorithms: Algorithm 2 and Algorithm 3. The latter is checking axioms **A1**–**A4** for a candidate ST-system. We tested this algorithm on various examples and showed the positive output for an ST-system satisfying all four axioms in Figure 4.2, on p. 39, and the negative outputs for ST-systems that violate one of the axioms in Figures 4.3, 4.4, 4.5 and 4.6, on pp 40-41. Method II involves three algorithms: Algorithm 2, Algorithm 4, and Algorithm 5. The most important here is Algorithm 4, which tests whether the ST-system generated by the synthesised net is isomorphic to the initial ST-system used as an input. We showed, as an example, the output for the positive case in Figure 4.7, and the output for the negative case in Figure 4.8, on p. 43. The experimental results showed that all algorithms implemented to apply Method I and Method II for the synthesis procedure worked correctly on examples from our pool of examples.

Chapter 5 describes the results of testing and checking the correctness of the algorithms involved in the synthesis procedure of ENL/LC-systems, for which one can calculate a canonical co-location relation $\simeq^{\mathfrak{ts}}_{min}$ from which any other suitable co-location relations can be derived. The most important algorithm of this chapter is Algorithm 7 that calculates $\simeq^{\mathfrak{ts}}_{min}$ and decides whether an ST-system is synthesisable to an ENL/LC-system or not. Identifying the ST-systems $\mathfrak{ts}_6$ and $\mathfrak{ts}_7$ (in Figures 5.1(a) and 5.2(a) on p. 50) not only allowed us to test the outcome for one of the conditional branches of the algorithm, but also helped us to establish a theoretical result that the class of ENL/LC-systems is a proper subclass of the class of ENL-systems.

Finally, we tested the correctness of algorithms for the minimisation of the synthesised nets (see Chapter 6). To test Algorithms 11, 12, and 13, which implement Reduction

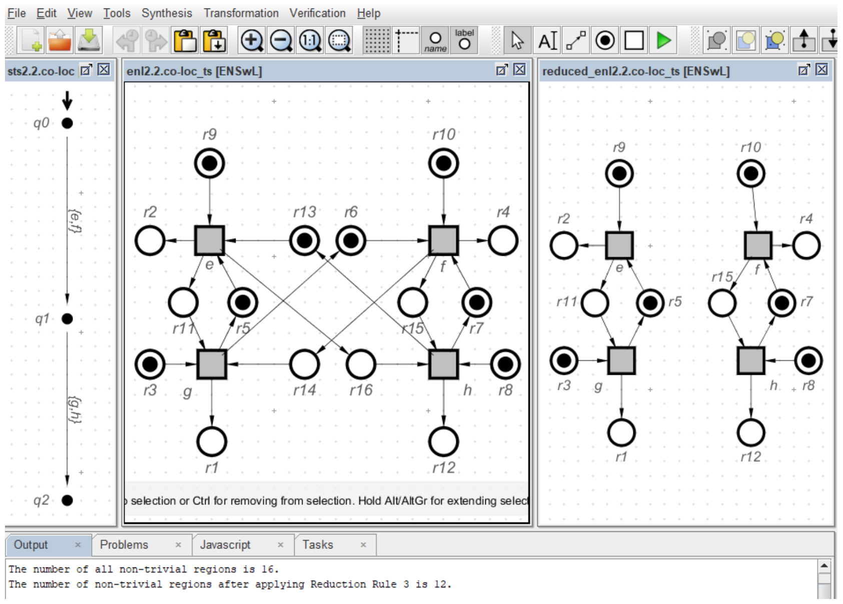
Rule 2 (aimed at the elimination of non-minimal regions), we used all the examples in our pool of examples. At this testing stage, the most important example was the ENLST-system in Figure 6.2(a), on p. 68. The identification of this example was a part of the work on proving properties of the composition operator $\oplus$, which is vital for the definition of minimal regions. The $\oplus$ operator (introduced originally in [28]) appeared to be not associative. This example was important for the introduction of the new composition operator $\oplus_s$ and the new definition of minimal regions. Figure 7.1 shows a screenshot from WORKCRAFT when we used the ENLST-system in Figure 6.2(a) to obtain its saturated synthesised net and the reduced net that resulted from applying to it the Reduction Rule 2.



```
The number of all non-trivial regions is 8.
The number of non-trivial regions after applying Reduction Rule 2 is 5.
```

Fig. 7.1 A screenshot from WORKCRAFT showing the ENLST-system from Figure 6.2(a); the ENL-system resulting from its synthesis; and the reduced ENL-system that uses only minimal regions (only Reduction Rule 2 applied).

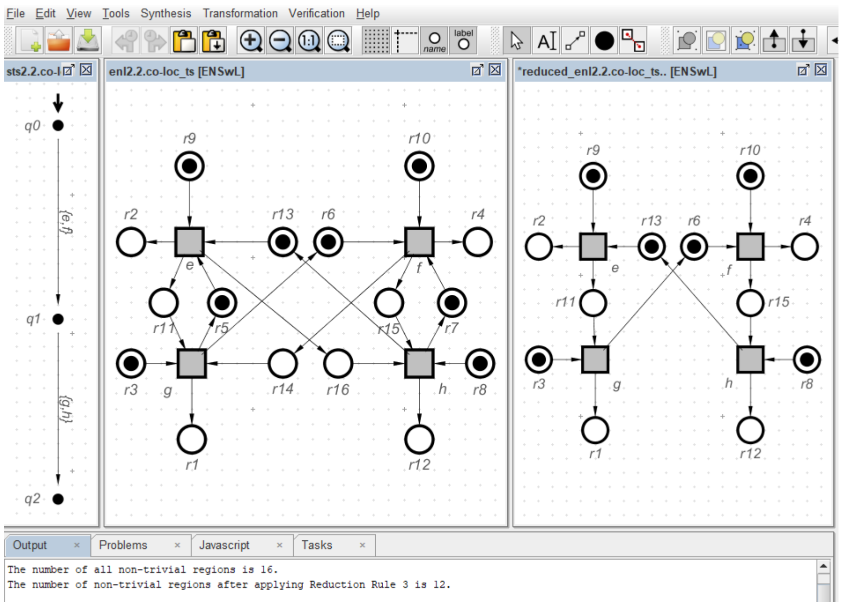
To test Algorithms 14 and 15, which implement Reduction Rule 3 (aimed at the elimination of redundant companion regions) we used all the examples in our pool of examples, but the most important of them were two sets of examples. The first of them contained thin ST-systems that have no companion regions as all their regions are based on different sets of states. The second important set of test examples was the set of $\mathfrak{ts}_{i,j}^{co-loc}$ ST-systems, where members have a lot of companion regions. Figure 7.2 shows a screenshot from WORKCRAFT with the results for the ENLST-system $\mathfrak{ts}_{2,2}^{co-loc}$ from Figure 6.4(a), on p. 83. Note that the resultant reduced ENL-system does not need to be unique. Figure 7.3 shows a screenshot from WORKCRAFT for $\mathfrak{ts}_{2,2}^{co-loc}$ with a different resultant reduced net.


File Edit View Tools Synthesis Transformation Verification Help
sts2.2.co-loc
enl2.2.co-loc_ts [ENSwL]
reduced_enl2.2.co-loc_ts [ENSwL]
q0
q1
q2
{e,f}
{g,h}
r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r13 r14 r15 r16
e f g h
Output Problems Javascript Tasks

selection or Ctrl for removing from selection. Hold Alt/AltGr for extending select

```
The number of all non-trivial regions is 16.
The number of non-trivial regions after applying Reduction Rule 3 is 12.
```

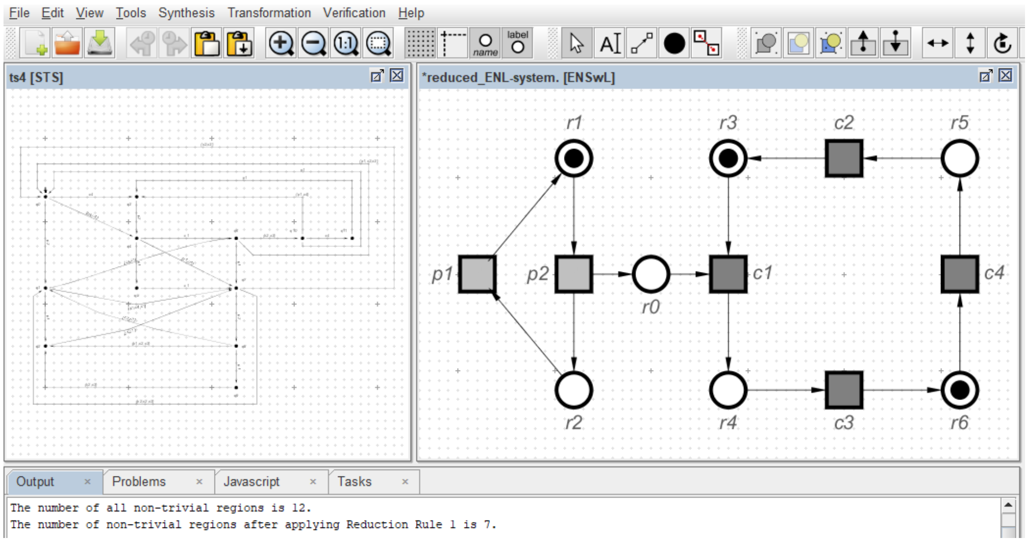
Fig. 7.2 A screenshot from WORKCRAFT showing the ENLST-system $\mathfrak{ts}_{2,2}^{co-loc}$ from Figure 6.4(a); the ENL-system resulting from its synthesis; and one of the possible reduced ENL-systems generating it obtained by removing a set of redundant companion regions (only Reduction Rule 3 applied).


File Edit View Tools Synthesis Transformation Verification Help
sts2.2.co-l
enl2.2.co-loc_ts [ENSwL]
*reduced_enl2.2.co-loc_ts.. [ENSwL]
q0
q1
q2
{e,f}
{g,h}
r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r13 r14 r15 r16
e f g h
Output Problems Javascript Tasks

```
The number of all non-trivial regions is 16.
The number of non-trivial regions after applying Reduction Rule 3 is 12.
```

Fig. 7.3 A screenshot from WORKCRAFT showing the ENLST-system $\mathfrak{ts}_{2,2}^{co-loc}$ from Figure 6.4(a); the ENL-system resulting from its synthesis; and one of the possible reduced ENL-systems generating it obtained by removing a set of redundant companion regions (only Reduction Rule 3 applied).

To test Algorithms 16 and 17, which implement Reduction Rule 1 (aimed at the elimination of redundant complementary regions) we used all the examples in our pool of examples. The application of Reduction Rule 1 might lead to many versions of smaller nets as shown in screenshots from WORKCRAFT in Figure 6.6, on p. 87, and Figure 7.4, where the Reduction Rule 1 is applied to the ST-system $\mathfrak{ts}_4$ in Figure 3.6, on p. 31. Some of the resultant nets for this example might be easier to understand, while other might be more different to interpret, when we analyse them as models of concurrent systems.


File Edit View Tools Synthesis Transformation Verification Help
ts4 [STS]
*reduced_ENL-system. [ENSwL]
r1
r3
c2
r5
p1
p2
r0
c1
c4
r2
r4
c3
r6
Output
Problems
Javascript
Tasks

```
The number of all non-trivial regions is 12.
The number of non-trivial regions after applying Reduction Rule 1 is 7.
```

Fig. 7.4 A screenshot from WORKCRAFT showing ST-system $\mathfrak{ts}_4$ from Figure 3.6; and one of the possible reduced ENL-systems generating it obtained by removing a set of complementary regions (only Reduction Rule 1 applied).

7.2.2 Checking the efficiency of algorithms

The experimental results allow us to analyse the performance efficiency of our tool and enable us to compare the execution time and memory consumption for all algorithms implemented to build our tool. These comparisons provide good optimization opportunities for developing all algorithms. To check the efficiency of algorithms we built benchmarks of ENL-systems with larger size and rising complexity that are described below.

There are three selected classes of nets/ST-systems. In the first class, we have ST-systems $\mathfrak{ts}_{i,j}$ generated by nets composed of several sequential subsystems, where every locality represents a sequential subsystem, and there is no conflict between events. Figure 2.2, on p. 14, shows the simplest example of such a system, $\mathfrak{ts}_{2,1}$, and the ST-system $\mathfrak{ts}_{3,3}$ generated by the ENL-system in Figure 3.7, on p. 32, shows how we extended them. We used this class of examples to obtain the experimental results of comparing the execution times of Algorithm 1 and Algorithm 1* as shown in Table 3.2 (see Chapter 3), and comparing the execution times of Algorithm 1 and Algorithm 7 as shown in Table 5.1 (see Chapter 5).

In the second class of ST-systems, we have ST-systems $\mathfrak{ts}_{i,j}^{co-loc}$ generated by nets composed of several sequential subsystems, where all events share the same locality, and there is no conflict between events. The smallest example of such a system, $\mathfrak{ts}_{2,1}^{co-loc}$, is shown in Figure 2.1, on p. 14, and Figure 6.4, on p. 83, shows the ENLST-system $\mathfrak{ts}_{2,2}^{co-loc}$ as an extended example. Such systems have a lot of companion regions, so we used them to check the efficiency of algorithms that implement Reduction Rules 1, 2, and 3. We used these systems to compare the effectiveness of Strategy (3,2,1) and Strategy (2,3,1), and to compare the efficiency of Strategy (3,2,1) and Strategy (2,3,1). The results of these comparisons are presented respectively in Table 6.2 and Tables 6.3 (see Chapter 6).

In the third class of ST-systems, we have the server-client ST-systems, $\mathfrak{ts}_{i,j}^{ser-cl}$, which contain conflicts events. The ENL-system generating a small example from this class, $\mathfrak{ts}_{2,2}^{ser-cl}$, is shown in Figure 4.10, on p. 46. We used this class of examples to test the efficiency of the algorithms that implement Method I and Method II for the synthesis procedure (see Section 4.1) in order to see how their performance scale with the increasing sizes of inputs. These experimental results are summarised in Section 4.1 Table 4.1 (see Chapter 4). In addition, we used these examples to show the effectiveness and the efficiency of Strategy (2,3,1) as shown in Table 6.4 (see Chapter 6).

Chapter 8

The WORKCRAFT framework and the implemented tool

This chapter describes a tool that implements existing and new theoretical results for the synthesis of ENL-systems.The new theoretical results were proved to support the development of algorithms in many ways: to justify the optimisations used in their implementations (see, for example, Chapter 3); to show the existence of several approaches for solving the same problems (Chapter4); to establish some important properties of step transition systems that were used as inputs for algorithms (see, for example, Chapter 5) or to establish some important properties of regions to reason about the net minimisation strategies (Chapter 6). The algorithms were implemented within a framework called WORKCRAFT [39, 38, 41].

8.1 The WORKCRAFT framework

The tool has been developed and implemented as a Java plugin within WORKCRAFT software framework. The WORKCRAFT is a collection of opensource tools for design, capturing, visual editing, simulation, synthesis, and verification of graph models. It supports a wide range of popular graph formalisms in the field of concurrent systems design, such as Petri nets, data-flow structures, gate-level circuits, *etc.* All such formalisms have an underlying static graph structure. Their semantics are defined using additional entities, e.g. tokens or node-arc states, which together form the overall state of the system. The similarities in notation allow a number of basic operations on these formalisms, such as editing, visualisation, serialisation [7], and translation from one formalism into another, to be generalised. More complex operations on the models can also be utilised, such as interfacing one model type with another. This

[7] WORKCRAFT provides an automatic serialisation facility for all models, i.e., , saving a new model plug-in into disk without effort from the developer to add additional code [38].

enables the researchers to model software and hardware systems using the most appropriate formalism, while still retaining the ability to simulate and analyse the overall system. In addition, WORKCRAFT provides a plugin-based framework for researchers who would like to define, model, and analyse new model types. Its plugin based architecture makes it an easily extensible and very flexible environment [39, 48]. A detailed WORKCRAFT description and manual can be found in [38].

WORKCRAFT is used for different purposes, such as education, research, and industrial circuit design. Therefore there are several categories of users: undergraduate students, academics, researchers, PhD students, developers, and industrial engineers. Since using WORKCRAFT makes it feasible for the user to harness the power of research tools it has been used and is planned to be used, in several research proposals as a valuable framework to solve and develop solutions for the problems stated in the project proposals [41, 30]. Furthermore, the application areas of WORKCRAFT are wide-ranging: from modelling concurrent systems and biological systems to designing asynchronous circuits and investigating crimes. For instance, WORKCRAFT was used to capture the behaviour of concurrent systems such as vending machines as Finite State Machines (FSM) [41]; to formally specify and derive the implementation of Speed-Independent (SI) circuits [43]; to specify and explore processor Instruction Set Architectures (ISAs); and synthesise efficient hardware implementations for the microcontrollers [31, 41]. Also, WORKCRAFT was used for investigating accidents. The Structured Occurrence Nets, SON model, was used for modelling and capturing the details of the Ladbroke Grove rail crash [47, 41, 30]. Also, WORKCRAFT has attracted industrial interest and it is used by several hardware design companies to develop real-life electronic circuits. From their point of view it is cheaper to use WORKCRAFT to check the correctness of the specifications of the electronic circuits and then to synthesise their Petri net models (correct by design) rather than developing electronic circuits and then testing their behaviour [41, 42, 29].

We took advantage of the WORKCRAFT plugin based architecture to implement our tool support for the synthesis of ENL-systems. In the next section, we describe the backend and frontend tools of our developed algorithms.

8.2 SYNTHCRAFT – a tool for synthesising ENL-systems

To use WORKCRAFT framework, one requires Java JDK 8 or later version, as well as installs and needs to include the WORKCRAFT library to a Java path. More importantly, Eclipse IDE [46] was used as an integrated environment for developing and debugging WORKCRAFT graph models [48]. By having such development environment, it was possible to implement our designed algorithms as WORKCRAFT plugins. As the synthesis procedure involves two different graph models, ENL-systems and ST-systems, we built SYNTHCRAFT tool, which introduced two plugins into WORKCRAFT: one to represent ENL-systems, and another to represent ST-systems.

Introducing the new two plugins into the WORKCRAFT was essential for implementing our synthesis procedures. The main reason was that the existing net models in WORKCRAFT like Petri Net (PN), Signal Transition Graph (STG), and Structured Occurrence Nets (SON) cannot be used to represent ENL-systems which have specific local maximal concurrency semantics. The only net model that associates transitions with localities is the Policy Net model, but in this model, transitions can have more than one locality which is not the case in the ENL-systems. Similarly, the provided formalisms for capturing the behaviour of concurrent systems in WORKCRAFT like Finite State Machines (FSM) and Finite State Transducers (FST) have transitions labelled by only one event, which was not suitable for transition system specifications in our tool. So we needed to extend this set of models to implement ST-systems and add more functionality. However, the new plugins use, wherever possible, the icons and the visualisation elements that are familiar to the WORKCRAFT users to ensure the consistency of our plugins with the rest of the existing tool.

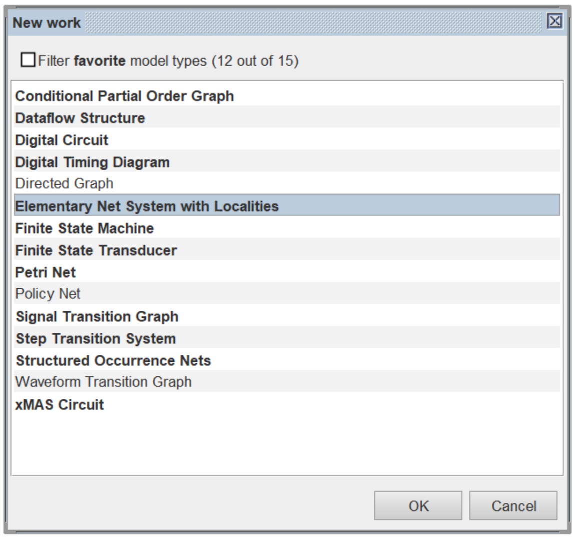
8.2.1 ENL-system plugin

This plugin is intended for capturing, editing, simulation and verifying of ENL-systems.

8.2.1.1 Capturing

To create a new work file to hold an ENL-system we select from the main menu of WORKCRAFT "File⟶Create work" to get menu item and in the "New work" dialogue we choose "Elementary Net System with Localities" as the model type (see Figure 8.1). The graphical interface of the ENL-system model is shown in Figure 8.2. The generic information for platform interface and settings of WORKCRAFT is provided in [48]. Our main goal here is to focus on features and functions related and designed for the ENL-system model.

In order to create a condition or an event as new nodes we can activate the condition generator ◉ or the event generator ☐ as shown in Figure 8.3 and click on the preferred position in the editor panel to place it there. In this way, we can create a series of conditions or events respectively. By activating the connection tool ⤢ as shown in Figure 8.3 we can link conditions and events by directed arcs. Note that conditions can only be connected to events and events can only be connected to conditions. So, in the case of connecting two nodes of the same type, a warning will be issued that the connection is invalid (see Figure 8.4).

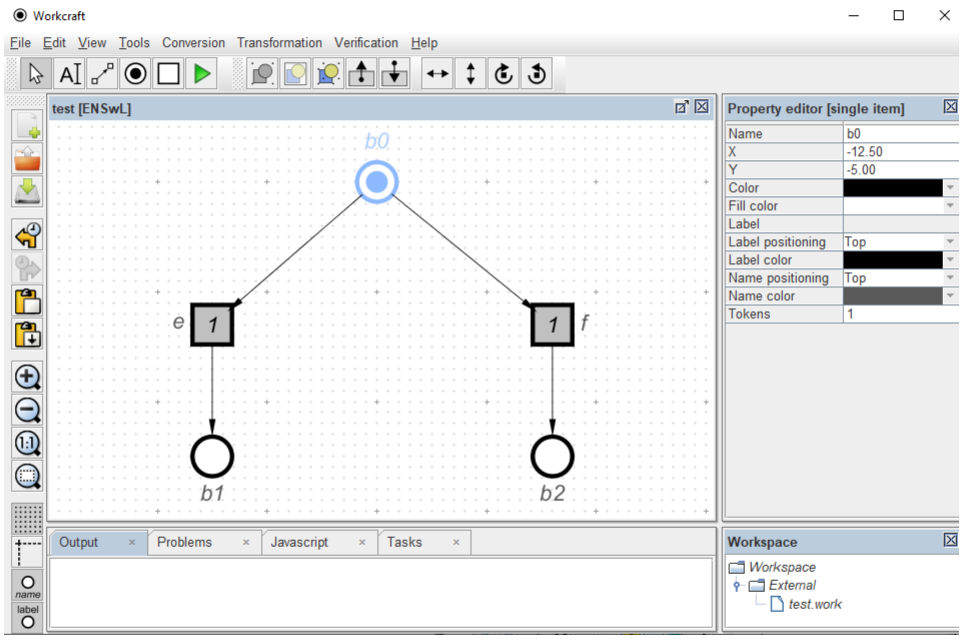

New work

☐ Filter **favorite** model types (12 out of 15)

Conditional Partial Order Graph
Dataflow Structure
Digital Circuit
Digital Timing Diagram
Directed Graph
Elementary Net System with Localities
Finite State Machine
Finite State Transducer
Petri Net
Policy Net
Signal Transition Graph
Step Transition System
Structured Occurrence Nets
Waveform Transition Graph
xMAS Circuit

OK Cancel

Fig. 8.1 Shows the screenshot from WORKCRAFT with the list of modules to create a new work.



Property editor [single item]

Name	b0
X	-12.50
Y	-5.00
Color	■
Fill color	
Label	
Label positioning	Top
Label color	■
Name positioning	Top
Name color	■
Tokens	1

Workspace

Workspace
External
test.work

Fig. 8.2 The ENL-system plugin interface.

8.2.1.2 Editing

Activating the selection tool as shown in Figure 8.3 enables us to edit an ENL-system model. Within the ENL-system plugin we can use all the standard editing features of WORKRAFT models such as copy, select, delete, drag-and-drop, undo, group, *etc*. They are described

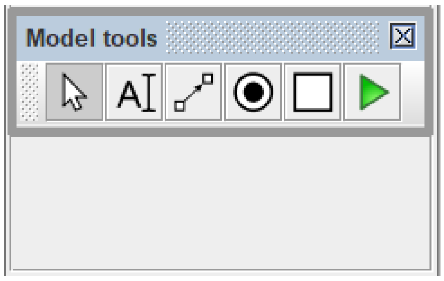
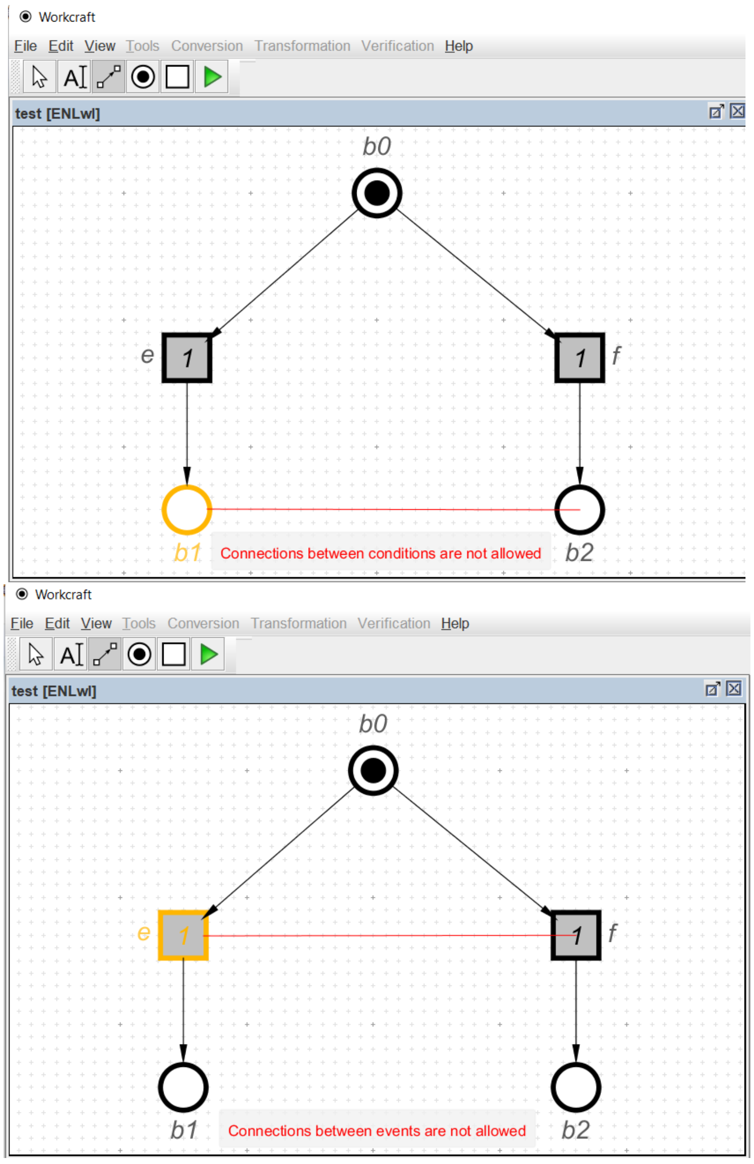


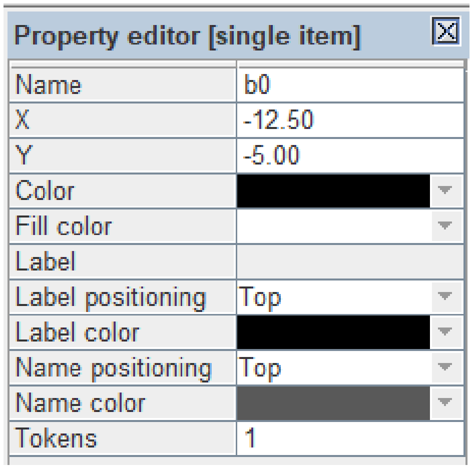
Fig. 8.3 The “Model tools” of ENL-system plugin.


Workcraft
File Edit View Tools Conversion Transformation Verification Help
test [ENLwl]
b0
e
1
f
b1
b2
Workcraft
File Edit View Tools Conversion Transformation Verification Help
test [ENLwl]
b0
e
1
f
b1
b2

Connections between conditions are not allowed

Connections between events are not allowed

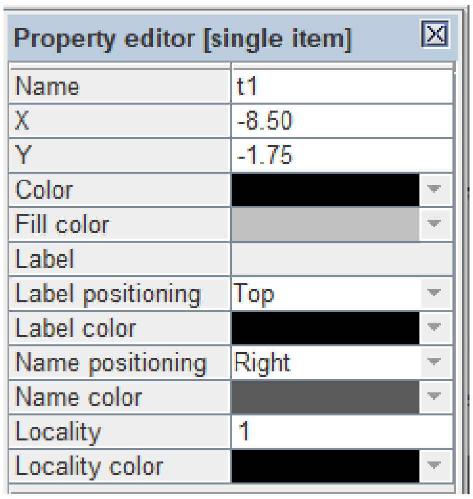
Fig. 8.4 Shows the screenshots from WORKCRAFT with the cases of invalid connections between conditions and events.



Property editor [single item]

Name	b0
X	-12.50
Y	-5.00
Color	■
Fill color	□
Label	
Label positioning	Top
Label color	■
Name positioning	Top
Name color	▨
Tokens	1

Fig. 8.5 Conditions property editor.



Property editor [single item]

Name	t1
X	-8.50
Y	-1.75
Color	■
Fill color	▨
Label	
Label positioning	Top
Label color	■
Name positioning	Right
Name color	▨
Locality	1
Locality color	■

Fig. 8.6 Events property editor.

under the generic WORKCRAFT help facility on “Selection controls” and “Property editor” [49, 50]. The features described below are related exclusively to the ENL-system plugin:

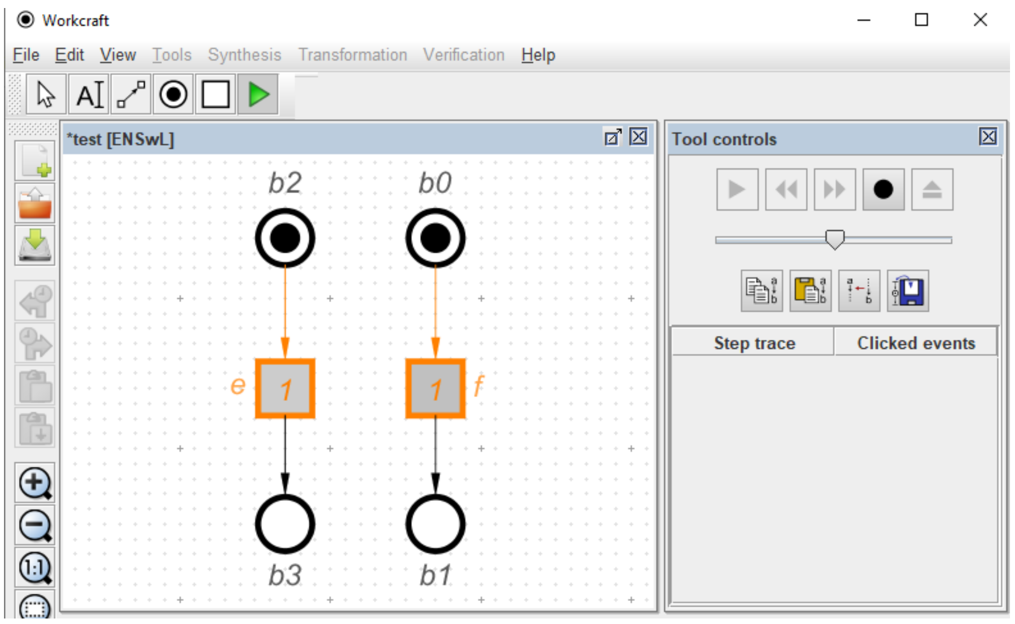
- Double-click on an empty condition to mark it with a token or edit “Tokens” option in the “Conditions property editor” (see Figure 8.5), remembering that each condition can only be marked with one token.
- Double-click a condition marked with a token to remove the token.
- Co-located events share the same colour and an associated number (see Figure 8.6).

Text notes can be created by activating the AI tool as shown in Figure 8.3 and pressing on the appropriate position in the editor panel to place the text there. By double-clicking on the note the user can write any text directly. Alternatively, a note can be edited by using the “Label” field in the “Property editor” when the note is selected.

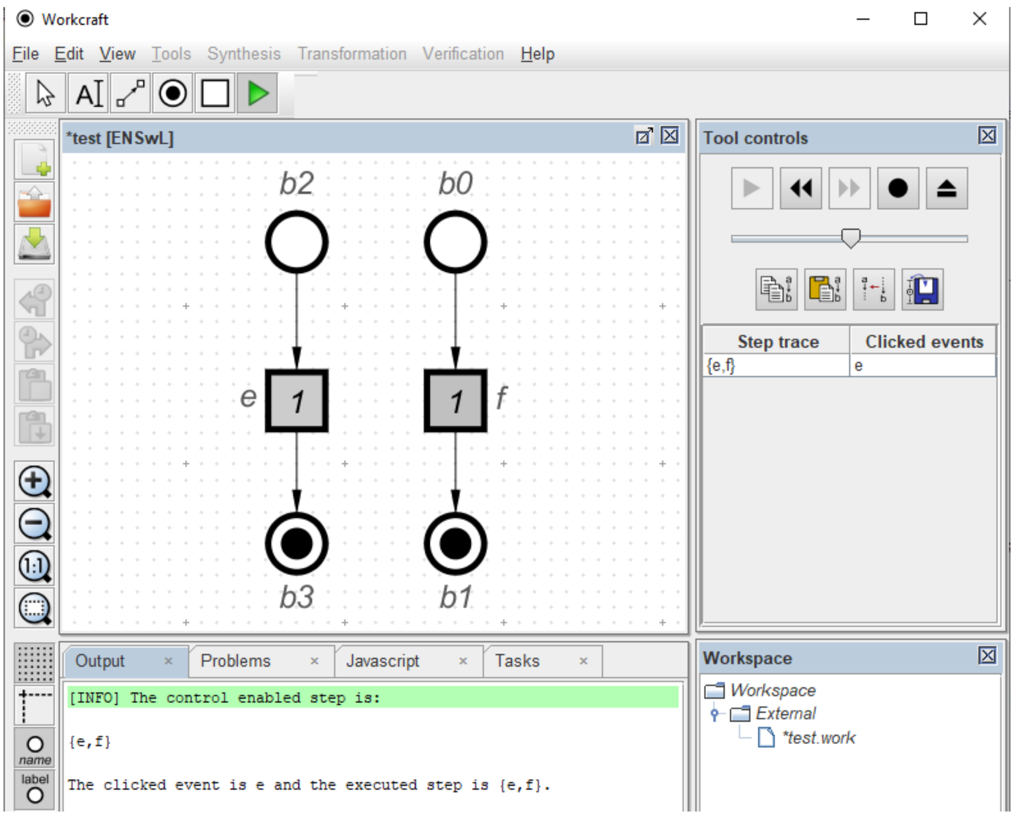
8.2.1.3 Simulation

The simulation function in the ENL-system model can be activated by clicking on the simulation tool ▶ as shown in Figure 8.3. The enabled events will be then highlighted in orange, and can be fired by clicking on them. In each step of the simulation we click on just one event. However, the steps of events will be executed according to the local maximal concurrency semantics of ENL-systems which is based on steps of simultaneously executed events. If we click on an enabled event and there are several control enabled steps containing this event, one of these steps will be chosen randomly to be fired. Figures 8.7 and 8.9 show the simulation for two examples. The first one shows an ENL-system with two co-located events e and f, and the initial case $c_0 = \{b_0, b_2\}$ (see Figure 8.7(a)). We have here: $\mathit{resenabled}(c_0) = \{\{e\},\{f\},\{e,f\}\}$ and $\mathit{enabled}(c_0) = \{\{e,f\}\}$. At c_0 we execute the step $\{e,f\}$ moving to $c_1 = \{b_1, b_3\}$. In this example, the enabled events e and f must fire together whether we clicked on e or f (see Figures 8.7(b) and 8.8(b)). The second one shows an ENL-system, where e and f events are not co-located, and the initial case is $c_0 = \{b_0, b_2\}$ (see Figure 8.9(a)). We have here $\mathit{resenabled}(c_0) = \mathit{enabled}(c_0) = \{\{e\},\{f\},\{e,f\}\}$. At c_0 we can execute the step $\{e\}$ moving to $c_1 = \{b_0, b_3\}$ (see Figure 8.9(b)). At the case c_1, $\mathit{enabled}(c_1) = \{\{f\}\}$. At c_1 we execute the step $\{f\}$ moving to $c_2 = \{b_1, b_3\}$ (see Figure 8.9(c)). However, there are several scenarios that may occur when we simulate the behaviour of the net in this example. Figure 8.9 shows the first scenario with the executed steps as described above; and Figure 8.10 shows the second scenario where at c_0 we can execute the step $\{e,f\}$ moving to $c_1 = \{b_1, b_3\}$. The main difference between the two scenarios is that when we clicked on the enabled event e, there are two control enabled steps containing e which are $\{e\}$ and $\{e,f\}$. So, in the first scenario step $\{e\}$ is executed first and in the second one the step $\{e,f\}$ is executed first. Similarly, there are two scenarios for this example assuming that event f was clicked at the beginning of the simulation.

After activating the simulation tool ▶, the buttons of the "Tool controls" panel (at the right hand side) become active (see Figure 8.7), and provide access to several additional simulation functions to analyse and navigate through the simulation trace. These are standard features of WORKCRAFT and they are described under its generic help facility on "Simulation controls" [51]. In addition, the simulation tool controls provide the means to analyse and navigate a recorded simulation. There are two sets of data for a simulation record. "Clicked events" records the sequence of clicked enabled events of the ENL-system. But because the enabled events will be executed based on the local maximal concurrency semantics of the net, the second set of data, "Step trace", records the sequence of the executed steps. For example, Figures 8.7 and 8.8 show that clicking on one of the enabled events e or f of the first example net would lead to both events being fired at the same time, because they belong to the same locality. Also, if there are several control enabled steps containing the clicked event (see Figures 8.9 and 8.10), one of them will be chosen randomly to be executed. Furthermore, more information is presented in the "Output" (lower left panel) showing a set of the control enabled steps, the enabled event clicked by the user, and the executed step.



(a)

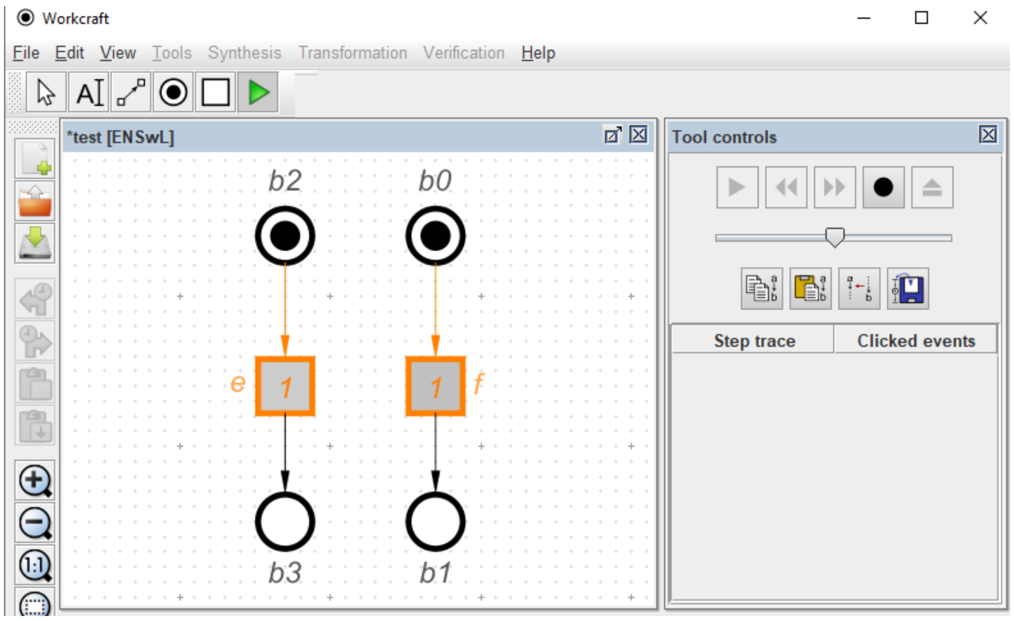


Step trace	Clicked events
{e,f}	e

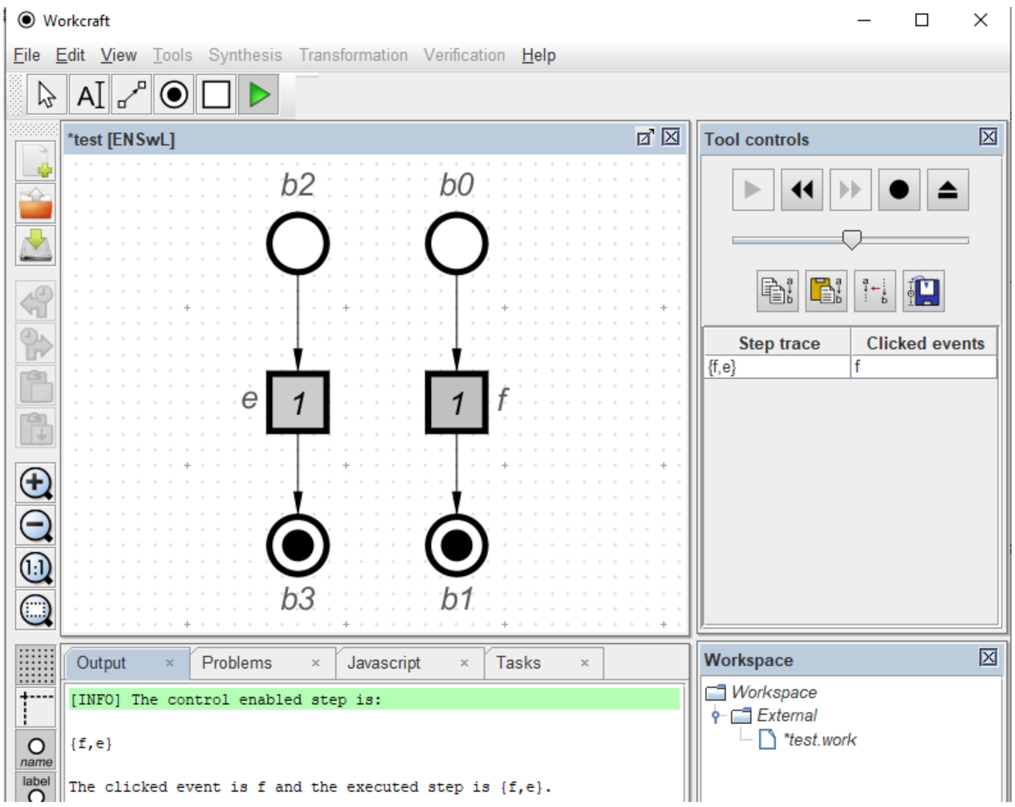
```
[INFO] The control enabled step is:

{e,f}

The clicked event is e and the executed step is {e,f}.
```

(b)

Fig. 8.7 Shows the ENL-system from Figure 2.1(b) with two co-located events e and f and the initial case $c_0 = \{b_0, b_2\}$ (a). After clicking on the enabled event e the step $\{e, f\}$ was executed and the system moved to $c_1 = \{b_1, b_3\}$ (b).



(a)

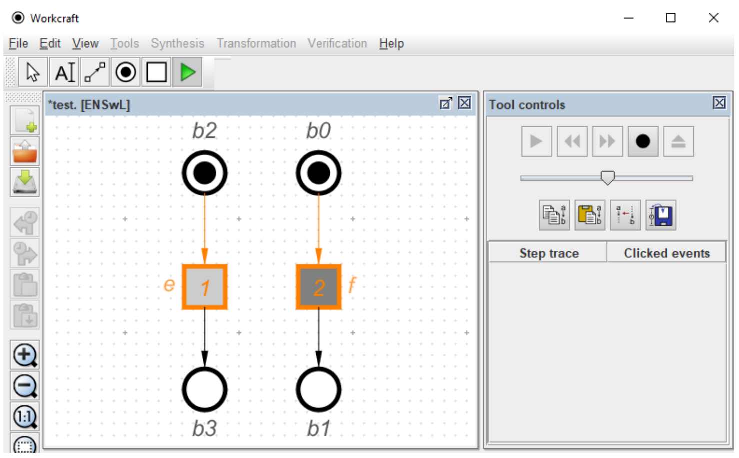


Step trace	Clicked events
{f,e}	f

```
[INFO] The control enabled step is:

{f,e}

The clicked event is f and the executed step is {f,e}.
```

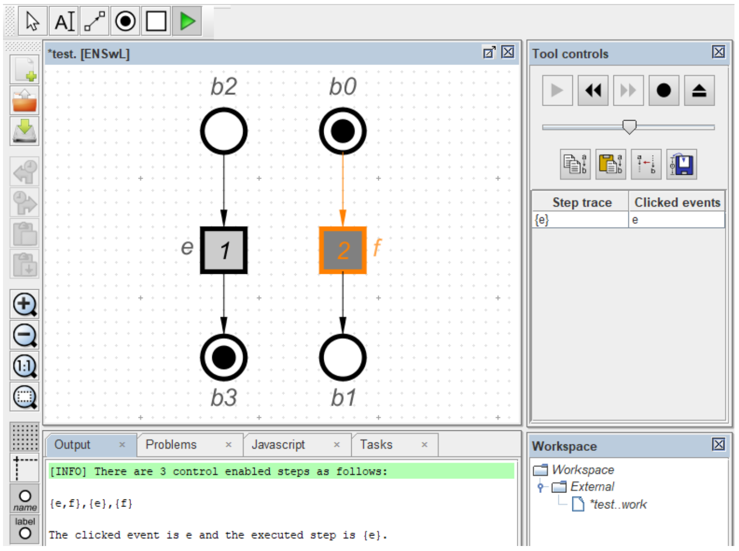
(b)

Fig. 8.8 Shows the ENL-system from Figure 2.1(b) with two co-located events e and f and the initial case $c_0 = \{b_0, b_2\}$ (a). After clicking on the enabled event f the step $\{e, f\}$ was executed and the system moved to $c_1 = \{b_1, b_3\}$ (b).


Workcraft
File Edit View Tools Synthesis Transformation Verification Help
*test. [ENSwL]
Tool controls
b2
b0
e
1
2
f
b3
b1

Step trace	Clicked events

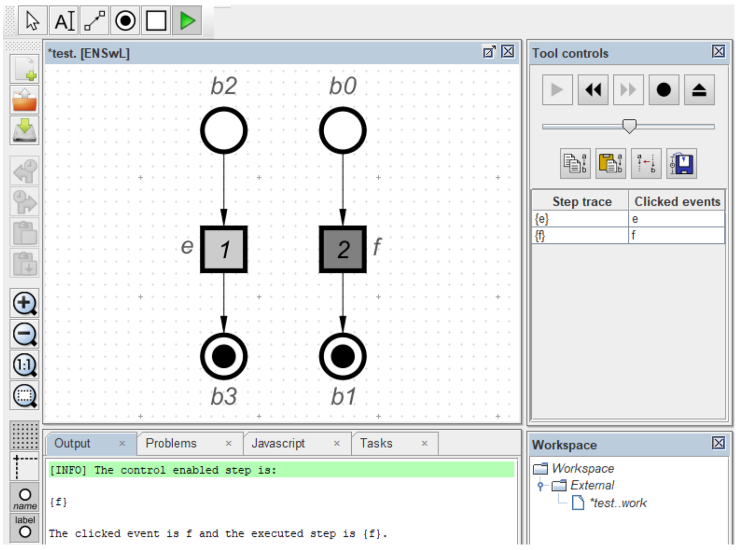
(a)



Step trace	Clicked events
{e}	e

```
[INFO] There are 3 control enabled steps as follows:

{e,f},{e},{f}

The clicked event is e and the executed step is {e}.
```

(b)

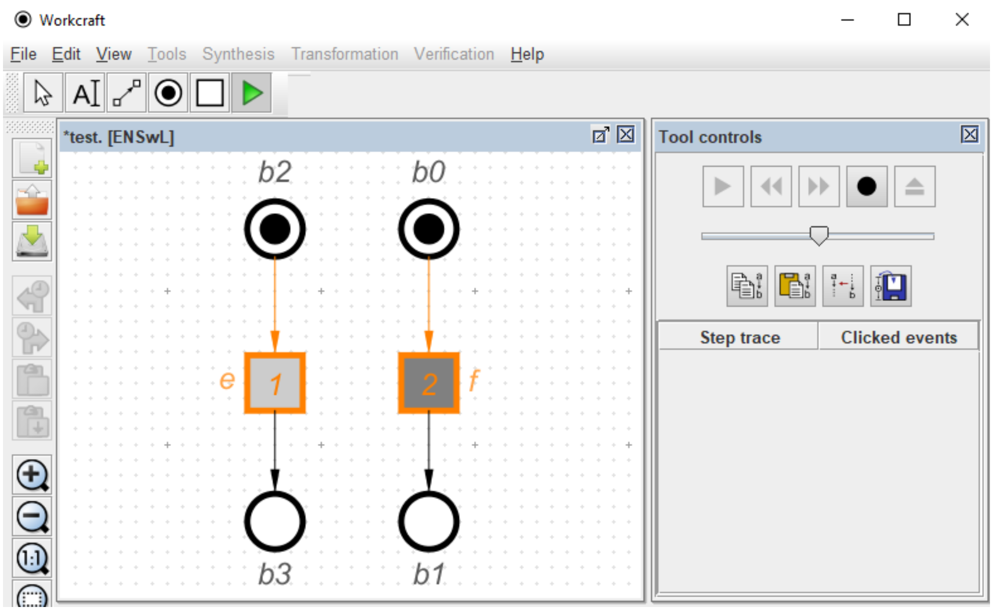


Step trace	Clicked events
{e}	e
{f}	f

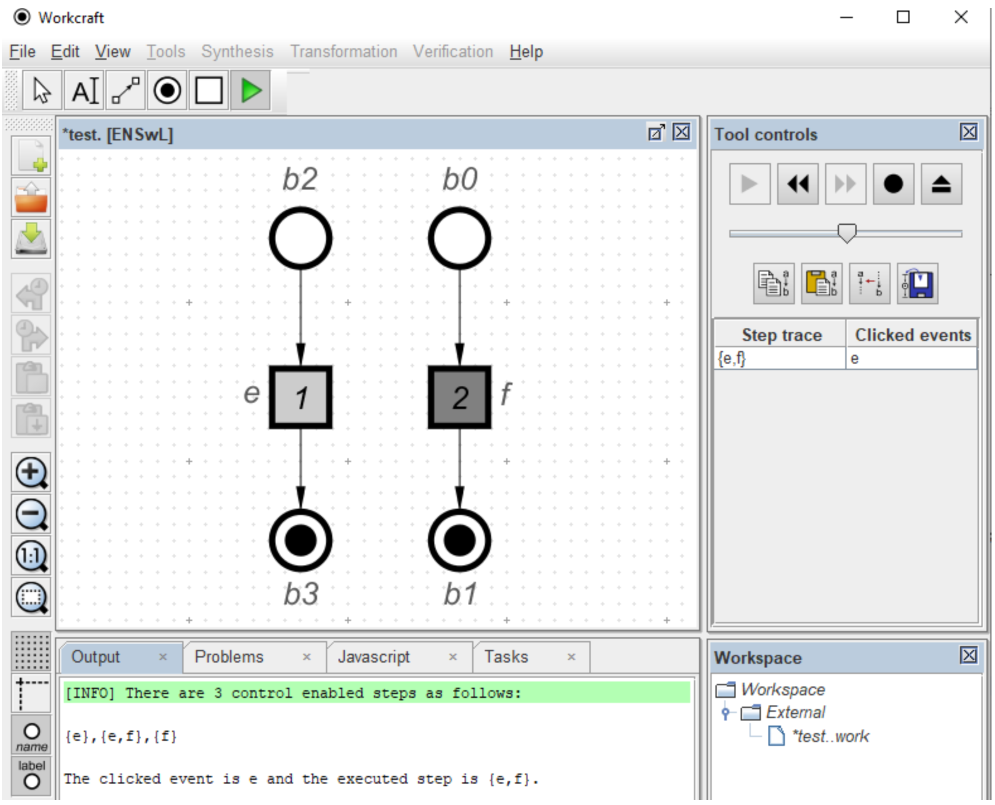
```
[INFO] The control enabled step is:

{f}

The clicked event is f and the executed step is {f}.
```

(c)

Fig. 8.9 Shows scenario 1 for the ENL-system from Figure 2.2(b) with non co-located events e and f and the initial case $c_0 = \{b_0, b_2\}$ (a). After clicking on the enabled event e the step $\{e\}$ was executed at c_0 and the system moved to $c_1 = \{b_0, b_3\}$ (b). After clicking on the enabled event f the step $\{f\}$ was executed at c_1 and the system moved to $c_2 = \{b_1, b_3\}$ (c).



(a)



Step trace	Clicked events
{e,f}	e

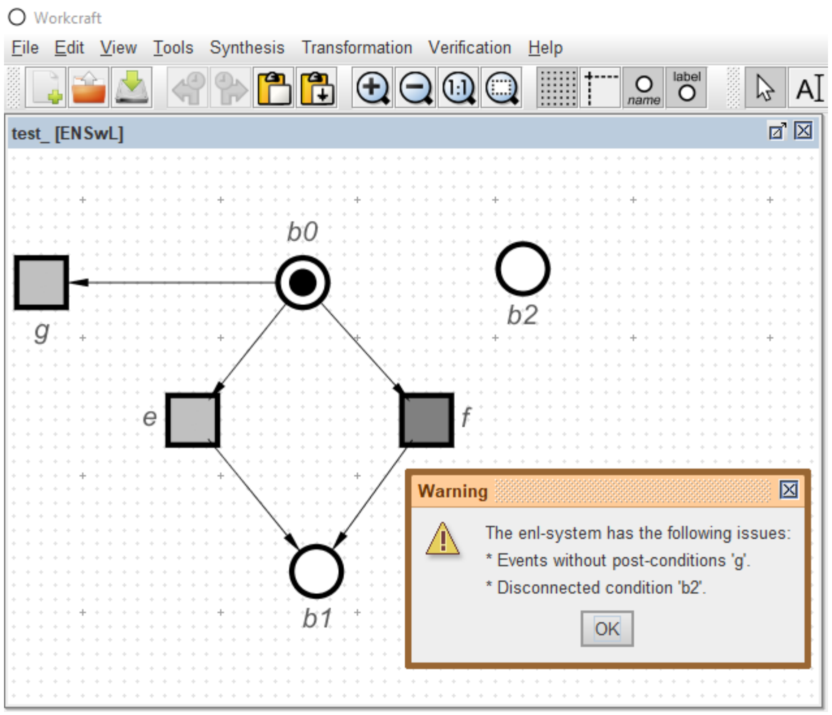
```
[INFO] There are 3 control enabled steps as follows:

{e},{e,f},{f}

The clicked event is e and the executed step is {e,f}.
```

(b)

Fig. 8.10 Shows scenario 2 for the ENL-system from Figure 2.2(b) with non co-located events e and f and the initial case $c_0 = \{b_0, b_2\}$ (a). After clicking on the enabled event e the step $\{e, f\}$ was executed at c_0 and the system moved to $c_1 = \{b_1, b_3\}$ (b).

8.2.1.4 Verification

An ENL-system can be verified for soundness via “Verification⟶Soundness” menu, which will identify any disconnected conditions or events as shown in Figure 8.11.


Workcraft
File Edit View Tools Synthesis Transformation Verification Help
test_ [ENSwL]
b0
b2
g
e
f
b1
Warning
OK

The enl-system has the following issues:
* Events without post-conditions 'g'.
* Disconnected condition 'b2'.

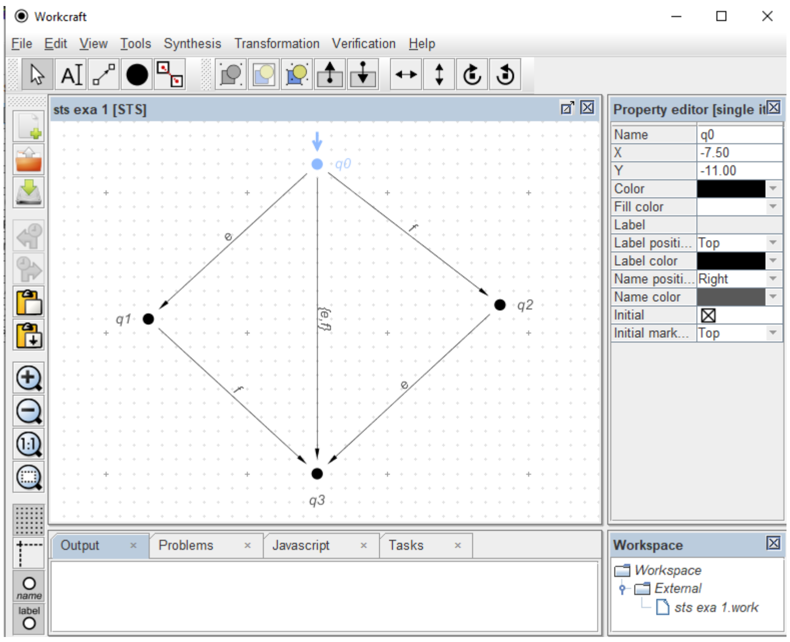
Fig. 8.11 Shows the verification result for an ENL-system with disconnected condition b_2 and event g that has no post-condition.

8.2.2 ST-system plugin

This plugin provides the facilities for capturing, editing, verification, and synthesis of ENL-systems from the specifications given in the form of ST-systems.

8.2.2.1 Capturing

To create an ST-system model from the main menu of WORKCRAFT we choose “File⟶Create work” to get menu item and in the “New work” dialogue we select the “Step Transition System” as the model type (see Figure 8.1). The graphical interface of the ST-system model is shown in Figure 8.12. The generic information for platform interface and settings of WORKCRAFT is provided in [48]. In this section, we will focus on the facilities designed for the ST-system plugin.



Name	q0
X	-7.50
Y	-11.00
Color	■
Fill color	
Label	
Label positi...	Top
Label color	■
Name positi...	Right
Name color	■
Initial	☒
Initial mark...	Top

Fig. 8.12 The ST-system plugin interface.

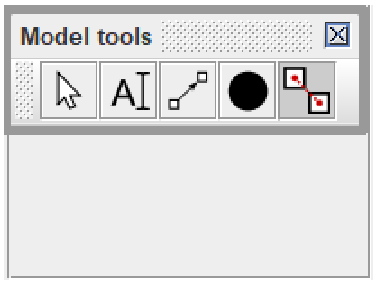
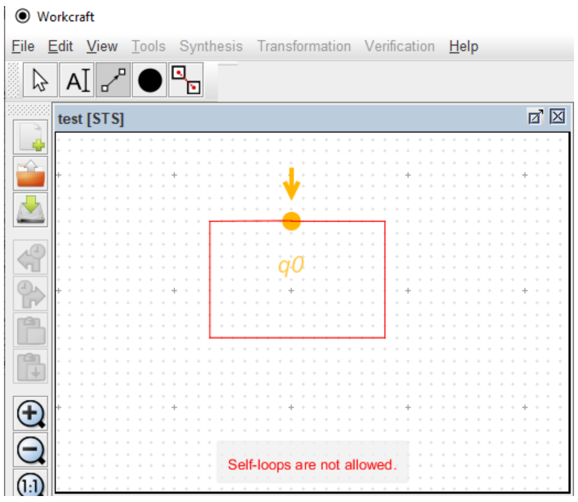


Fig. 8.13 The "Model tools" of ST-system plugin.

In order to create a new state as a node we need to activate the state generator as shown in Figure 8.13, then click the editor panel in the desired position where a new state appears. In this way, we can create as many states as we need. To connect two states together we can activate the connection tool as shown in Figure 8.13 to link them by a directed arc, where first state will be the source and the second state will be the target of the newly created arc. Note that self-loops are not allowed in ST-systems (see Figure 8.14).

8.2.2.2 Editing

To edit an ST-system model we activate the selection tool as shown in Figure 8.13. As described in Section 8.2.1.2 all the standard editing features of WORKCRAFT models are the same. So, the functions mentioned below are designed exclusively for the ST-system plugin:


Workcraft
File Edit View Tools Synthesis Transformation Verification Help
test [STS]
q0

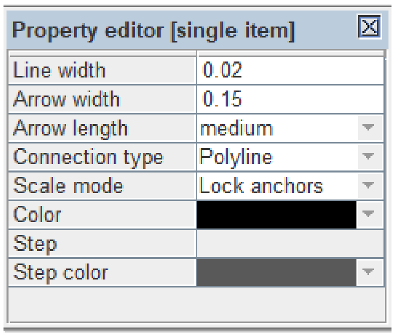
Self-loops are not allowed.

Fig. 8.14 Shows a self-loop, which is not allowed.

- Click on an arc to edit its “Step” field as shown in Figure 8.15 remembering that singleton steps annotating transitions are denoted without brackets (e.g., e instead of $\{e\}$).
- Select an arc to colour its step by editing “Step colour” field as shown in Figure 8.15.

For every event $e \in E$, $[e]_{\bumpeq}$ is the equivalence class of the co-location relation $\bumpeq$ to which e belongs (i.e., the *locality* of e). By activating the events localities tool as shown in Figure 8.13 we can use the “Tool controls” panel to link every event e with its locality as shown in Figure 8.16. If there are events without associated localities then a warning will be issued (see Figure 8.17).

Textual comments can be created by activating the AI tool as shown in Figure 8.13 and it works as described in Section 8.2.1.2, which is similar for all the other models in WORKCRAFT.



Property editor [single item]

Line width	0.02
Arrow width	0.15
Arrow length	medium
Connection type	Polyline
Scale mode	Lock anchors
Color	■
Step	
Step color	■

Fig. 8.15 Property editor for arcs.

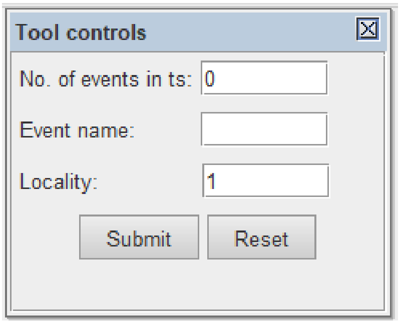
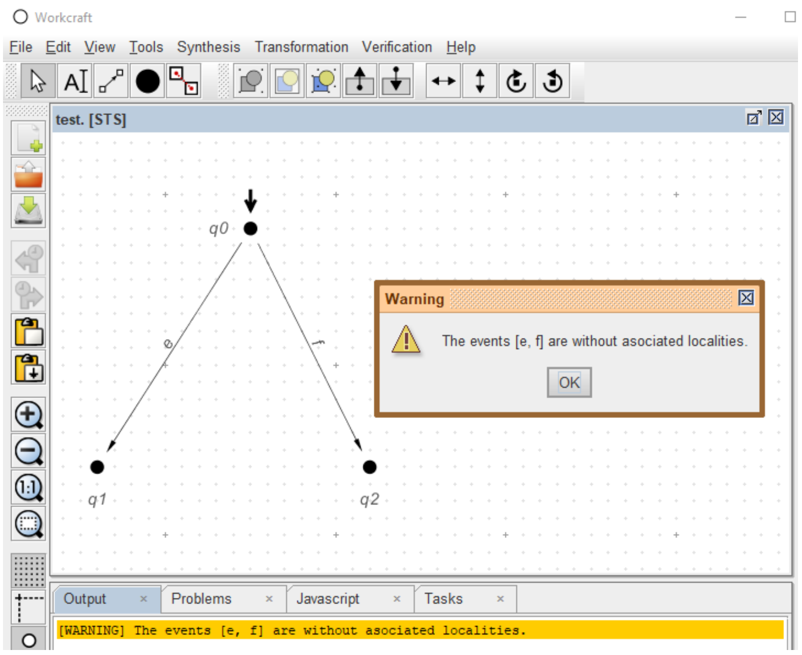


Fig. 8.16 An interface for associating each event with its locality.



The events [e, f] are without asociated localities.

```
[WARNING] The events [e, f] are without asociated localities.
```

Fig. 8.17 A warning message that events e and f are without associated localities.

8.2.2.3 Verification

An ST-system can be verified for satisfying axioms **A1–A4** (see Definition 2.1.7). Figure 8.18 shows the available options under the "Verification" menu, and example results of checking these options were presented in Chapter 4:

- Figure 4.2, on p. 39, shows the verification result for an ST-system, which satisfies all the axioms (**A1–A4**).
- Figure 4.3, on p. 40, shows the verification result for an ST-system, which violates the axiom **A1**.

- Figure 4.4, on p. 40, shows the verification result for an ST-system, which violates the axiom **A2**.
- Figure 4.5, on p. 41, shows the verification result for an ST-system, which violates the axiom **A3**.
- Figure 4.6, on p. 41, shows the verification result for an ST-system, which violates the axiom **A4**.

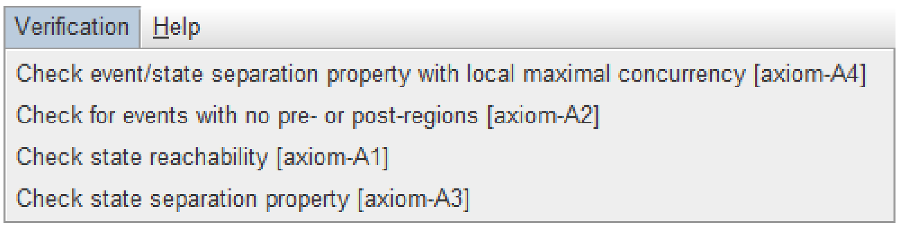

Verification
Help
Check event/state separation property with local maximal concurrency [axiom-A4]
Check for events with no pre- or post-regions [axiom-A2]
Check state reachability [axiom-A1]
Check state separation property [axiom-A3]

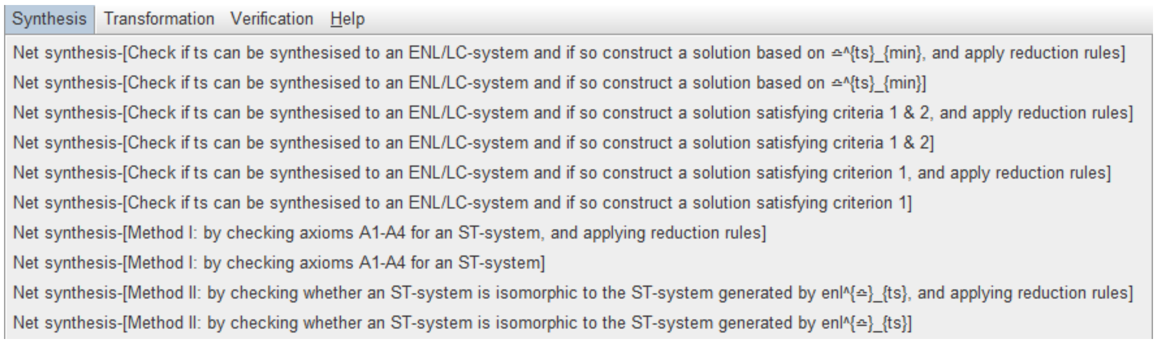
Fig. 8.18 The “Verification” menu in the ST-system plugin.

8.2.2.4 Synthesis

To apply the synthesis procedure, the specification of an ST-system should be used as an input to the tool, which would decide whether it can be synthesised as an ENL-system or not. In the case of a positive answer, the tool would produce an ENL-system with the specified concurrent behaviour (see examples in Figures 8.20 and 8.21). Otherwise, the tool would produce a message with a negative result (see Figures 8.22 and 8.23). Figure 8.20 shows that a given ST-system can be synthesised to an ENL-system, because it satisfies all axioms **A1**–**A4**. On the other hand, Figure 8.22 shows that an ST-system cannot be synthesised to an ENL-system, because it does not satisfy one of the axioms, **A4**. Figure 8.21 shows that a given ST-system $\mathfrak{ts}$ can be synthesised to an ENL-system, because the ST-system generated by $\mathfrak{enl}_{\widehat{\mathfrak{ts}}}$ is isomorphic to $\mathfrak{ts}$. Figure 8.23 shows a negative result obtained after the application of Method II to an ST-system.

Figure 8.19 shows the full “Synthesis” menu in the ST-system plugin, which provides a list of options for running different synthesis algorithms for a given ST-system.

Note that the produced ENL-system may have some redundant conditions. These conditions (or most of them) can be removed by using the reduction rules described in Chapter 6. Therefore, the “Synthesis” menu, for every synthesis option that leads to a saturated net, offers an option that invokes minimisation algorithms. However, even after the application of the reduction rules there is no guarantee that there are no redundant regions as was shown in Chapter 6 for the case of $\mathfrak{ts}_2$ (see Figure 6.10 on p. 91)

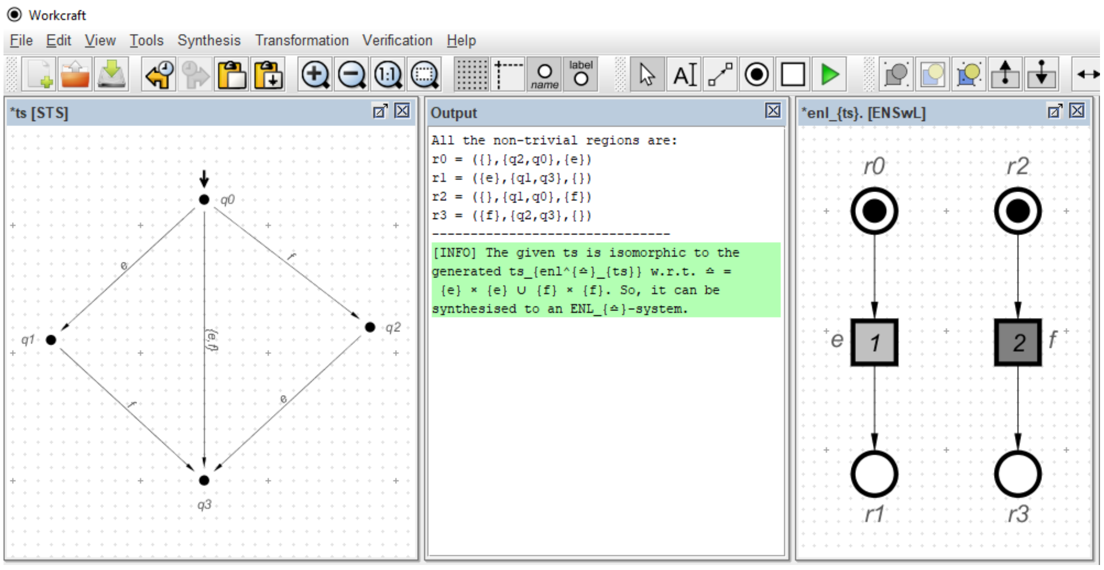


Net synthesis-[Check if ts can be synthesised to an ENL/LC-system and if so construct a solution based on ≏^{ts}_{min}, and apply reduction rules]
Net synthesis-[Check if ts can be synthesised to an ENL/LC-system and if so construct a solution based on ≏^{ts}_{min}]
Net synthesis-[Check if ts can be synthesised to an ENL/LC-system and if so construct a solution satisfying criteria 1 & 2, and apply reduction rules]
Net synthesis-[Check if ts can be synthesised to an ENL/LC-system and if so construct a solution satisfying criteria 1 & 2]
Net synthesis-[Check if ts can be synthesised to an ENL/LC-system and if so construct a solution satisfying criterion 1, and apply reduction rules]
Net synthesis-[Check if ts can be synthesised to an ENL/LC-system and if so construct a solution satisfying criterion 1]
Net synthesis-[Method I: by checking axioms A1-A4 for an ST-system, and applying reduction rules]
Net synthesis-[Method I: by checking axioms A1-A4 for an ST-system]
Net synthesis-[Method II: by checking whether an ST-system is isomorphic to the ST-system generated by enl^{≏}_{ts}, and applying reduction rules]
Net synthesis-[Method II: by checking whether an ST-system is isomorphic to the ST-system generated by enl^{≏}_{ts}]

Fig. 8.19 The Synthesis menu in the ST-system plugin, where criterion 1 refers to checking if ST-system can be synthesised to an ENL/LC-system and if so constructing a solution with the smallest number of localities, and criterion 2 requires in addition to have the most balanced distribution of events between them.

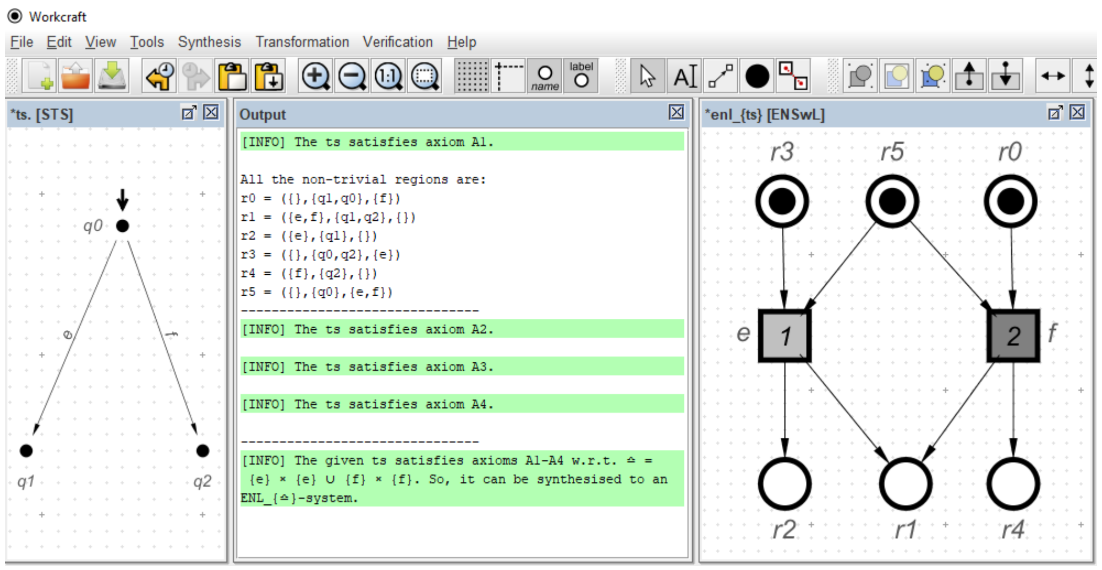
Below are further results of testing the synthesis options that are available under the "Synthesis" menu:

- Figure 8.24 shows that a given ST-system $\mathfrak{ts}$ can be synthesised to an ENL/LC-system. Figure 5.3 , on p. 54, discussed in Chapter 5, shows the negative result for this synthesis procedure.
- Figure 8.25 shows that a given ST-system, $\mathfrak{ts}_3$, can be synthesised to an ENL-system, because the ST-system generated by $\mathfrak{enl}^{\widehat{=}}_{\mathfrak{ts}_3}$ is isomorphic to $\mathfrak{ts}_3$. This figure shows the results of the synthesis procedure that involved the minimisation of the net.



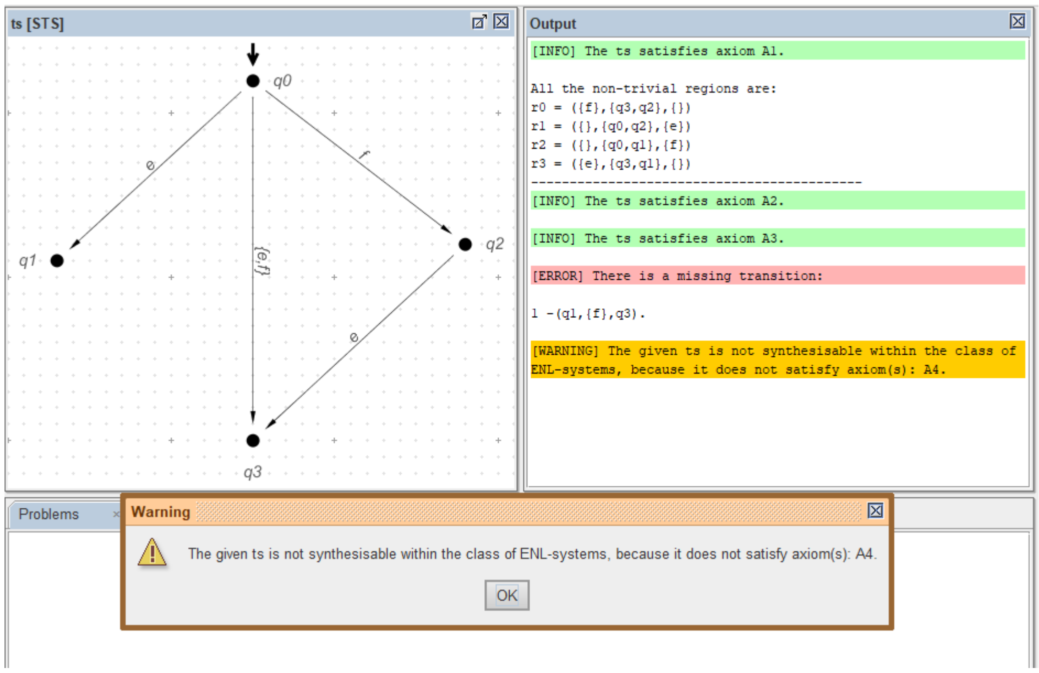
```
All the non-trivial regions are:
r0 = ({},{q2,q0},{e})
r1 = ({e},{q1,q3},{})
r2 = ({},{q1,q0},{f})
r3 = ({f},{q2,q3},{})
-------------------------------
[INFO] The given ts is isomorphic to the
generated ts_{enl^{≏}_{ts}} w.r.t. ≏ =
 {e} × {e} ∪ {f} × {f}. So, it can be
synthesised to an ENL_{≏}-system.
```

Fig. 8.21 An example of a successful synthesis of the ST-system in Figure 2.2(a) by using Method II (see Section 4.1).



```
[INFO] The ts satisfies axiom A1.

All the non-trivial regions are:
r0 = ({},{q1,q0},{f})
r1 = ({e,f},{q1,q2},{})
r2 = ({e},{q1},{})
r3 = ({},{q0,q2},{e})
r4 = ({f},{q2},{})
r5 = ({},{q0},{e,f})
--------------------------------
[INFO] The ts satisfies axiom A2.

[INFO] The ts satisfies axiom A3.

[INFO] The ts satisfies axiom A4.

--------------------------------
[INFO] The given ts satisfies axioms A1-A4 w.r.t. ≙ =
 {e} × {e} ∪ {f} × {f}. So, it can be synthesised to an
ENL_{≙}-system.
```

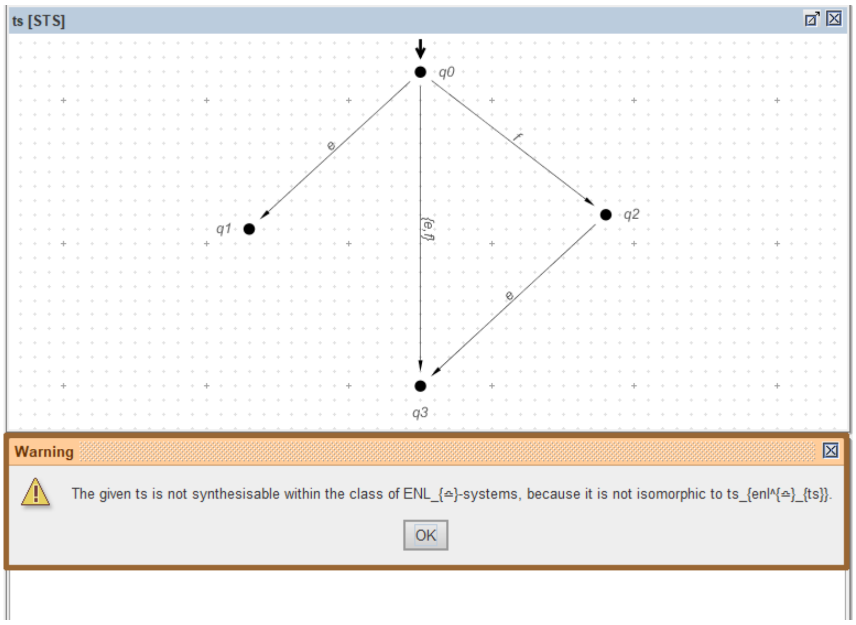
Fig. 8.20 An example of a successful synthesis of the ST-system in Figure 6.3(a) by using Method I (see Section 4.1).



```
[INFO] The ts satisfies axiom A1.

All the non-trivial regions are:
r0 = ({f},{q3,q2},{})
r1 = ({},{q0,q2},{e})
r2 = ({},{q0,q1},{f})
r3 = ({e},{q3,q1},{})
--------------------------------------------
[INFO] The ts satisfies axiom A2.

[INFO] The ts satisfies axiom A3.

[ERROR] There is a missing transition:

1 -(q1,{f},q3).

[WARNING] The given ts is not synthesisable within the class of
ENL-systems, because it does not satisfy axiom(s): A4.
```

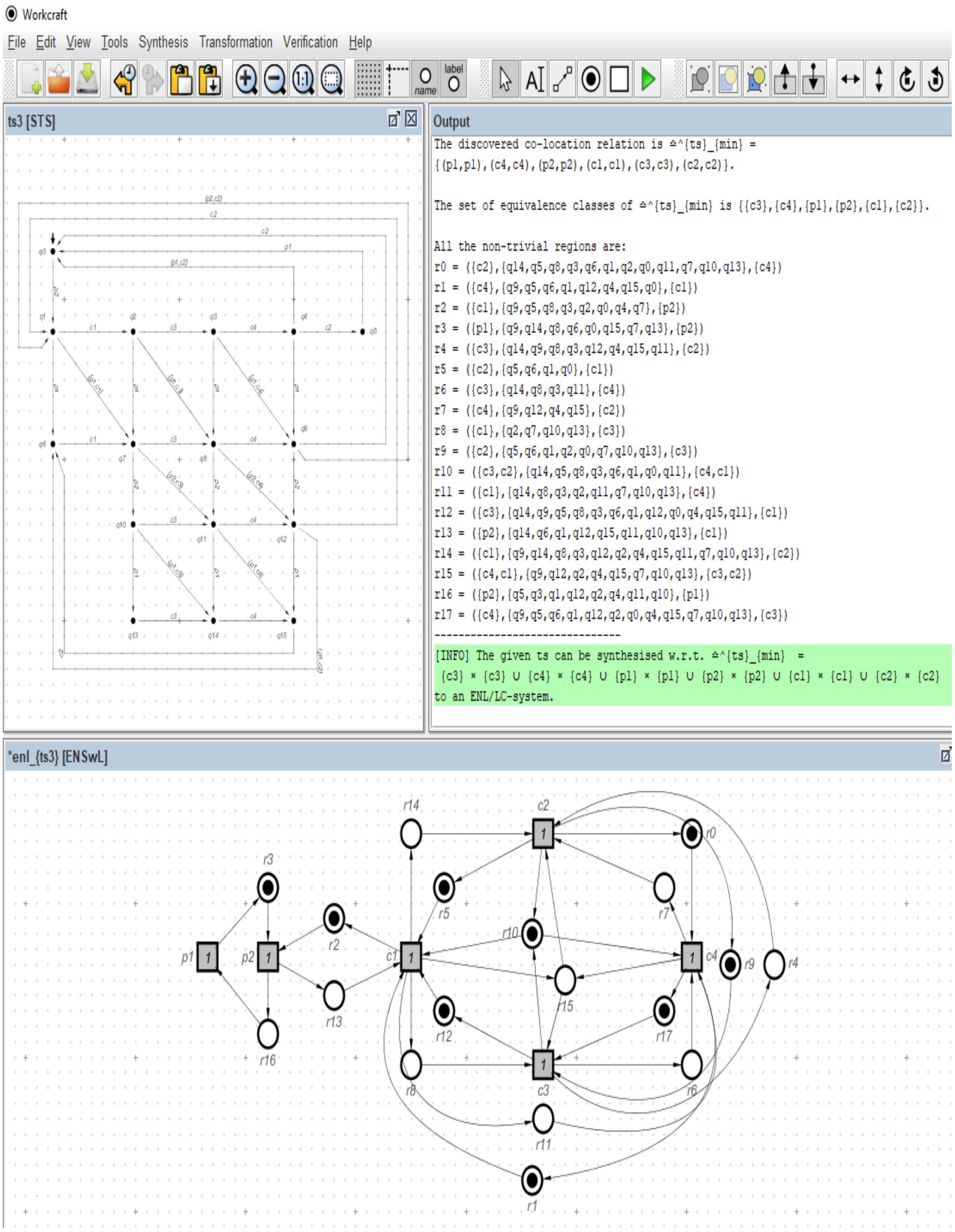
The given ts is not synthesisable within the class of ENL-systems, because it does not satisfy axiom(s): A4.

Fig. 8.22 An example of a negative case for the synthesis procedure when using Method I (see Section 4.1).



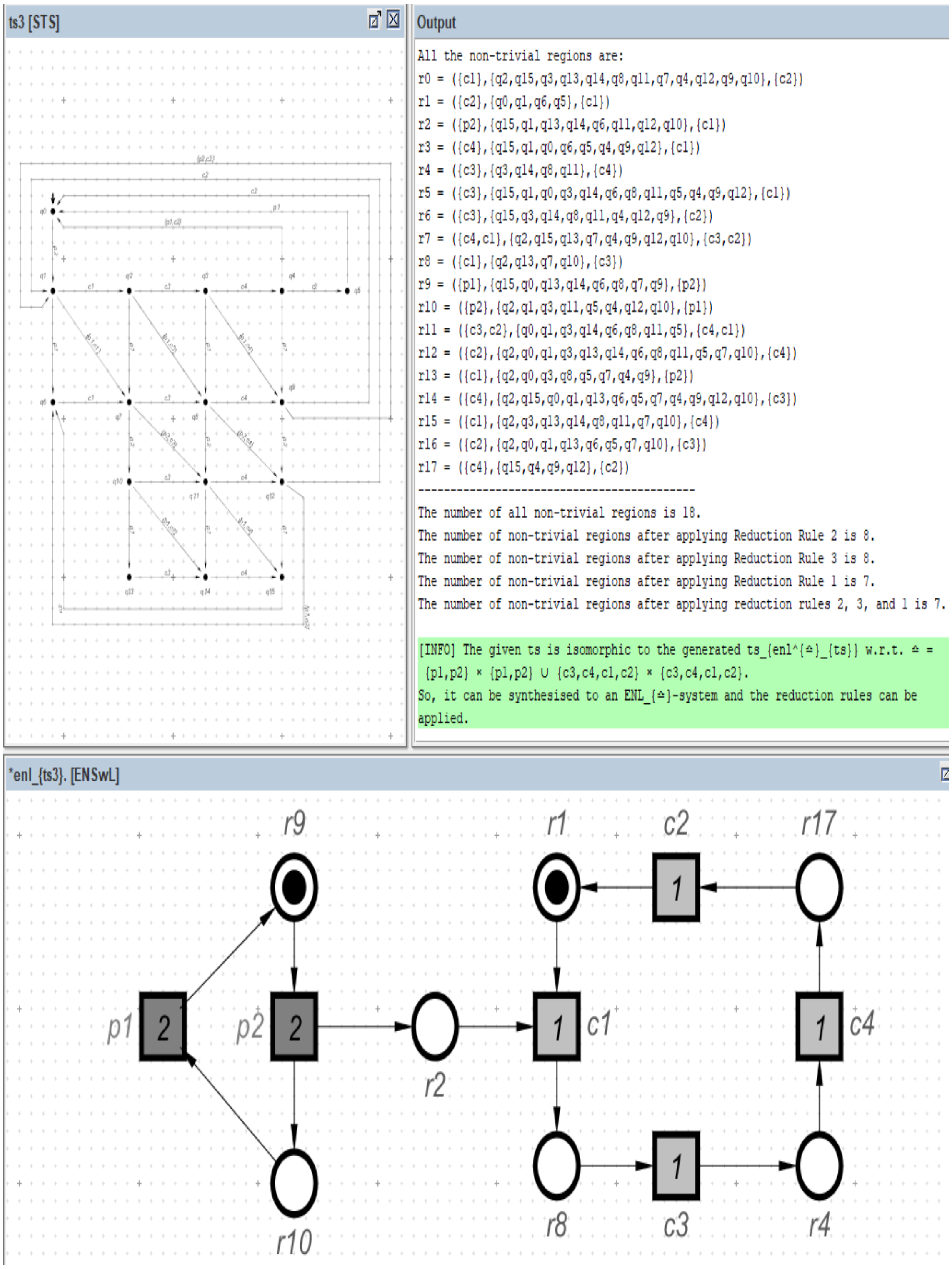
The given ts is not synthesisable within the class of ENL_{≏}-systems, because it is not isomorphic to ts_{enl^{≏}_{ts}}.

Fig. 8.23 An example of a negative case for the synthesis procedure when using Method II (see Section 4.1).



```
The discovered co-location relation is ≙^{ts}_{min} =
{(p1,p1),(c4,c4),(p2,p2),(c1,c1),(c3,c3),(c2,c2)}.

The set of equivalence classes of ≙^{ts}_{min} is {{c3},{c4},{p1},{p2},{c1},{c2}}.

All the non-trivial regions are:
r0 = ({c2},{q14,q5,q8,q3,q6,q1,q2,q0,q11,q7,q10,q13},{c4})
r1 = ({c4},{q9,q5,q6,q1,q12,q4,q15,q0},{c1})
r2 = ({c1},{q9,q5,q8,q3,q2,q0,q4,q7},{p2})
r3 = ({p1},{q9,q14,q8,q6,q0,q15,q7,q13},{p2})
r4 = ({c3},{q14,q9,q8,q3,q12,q4,q15,q11},{c2})
r5 = ({c2},{q5,q6,q1,q0},{c1})
r6 = ({c3},{q14,q8,q3,q11},{c4})
r7 = ({c4},{q9,q12,q4,q15},{c2})
r8 = ({c1},{q2,q7,q10,q13},{c3})
r9 = ({c2},{q5,q6,q1,q2,q0,q7,q10,q13},{c3})
r10 = ({c3,c2},{q14,q5,q8,q3,q6,q1,q0,q11},{c4,c1})
r11 = ({c1},{q14,q8,q3,q2,q11,q7,q10,q13},{c4})
r12 = ({c3},{q14,q9,q5,q8,q3,q6,q1,q12,q0,q4,q15,q11},{c1})
r13 = ({p2},{q14,q6,q1,q12,q15,q11,q10,q13},{c1})
r14 = ({c1},{q9,q14,q8,q3,q12,q2,q4,q15,q11,q7,q10,q13},{c2})
r15 = ({c4,c1},{q9,q12,q2,q4,q15,q7,q10,q13},{c3,c2})
r16 = ({p2},{q5,q3,q1,q12,q2,q4,q11,q10},{p1})
r17 = ({c4},{q9,q5,q6,q1,q12,q2,q0,q4,q15,q7,q10,q13},{c3})
------------------------------
[INFO] The given ts can be synthesised w.r.t. ≙^{ts}_{min}  =
 {c3} × {c3} ∪ {c4} × {c4} ∪ {p1} × {p1} ∪ {p2} × {p2} ∪ {c1} × {c1} ∪ {c2} × {c2}
to an ENL/LC-system.
```

Fig. 8.24 An example of a successful synthesis for the ST-system $\mathfrak{ts}_3$ in Figure 3.1 to an ENL/LC-system.



```
All the non-trivial regions are:
r0 = ({c1},{q2,q15,q3,q13,q14,q8,q11,q7,q4,q12,q9,q10},{c2})
r1 = ({c2},{q0,q1,q6,q5},{c1})
r2 = ({p2},{q15,q1,q13,q14,q6,q11,q12,q10},{c1})
r3 = ({c4},{q15,q1,q0,q6,q5,q4,q9,q12},{c1})
r4 = ({c3},{q3,q14,q8,q11},{c4})
r5 = ({c3},{q15,q1,q0,q3,q14,q6,q8,q11,q5,q4,q9,q12},{c1})
r6 = ({c3},{q15,q3,q14,q8,q11,q4,q12,q9},{c2})
r7 = ({c4,c1},{q2,q15,q13,q7,q4,q9,q12,q10},{c3,c2})
r8 = ({c1},{q2,q13,q7,q10},{c3})
r9 = ({p1},{q15,q0,q13,q14,q6,q8,q7,q9},{p2})
r10 = ({p2},{q2,q1,q3,q11,q5,q4,q12,q10},{p1})
r11 = ({c3,c2},{q0,q1,q3,q14,q6,q8,q11,q5},{c4,c1})
r12 = ({c2},{q2,q0,q1,q3,q13,q14,q6,q8,q11,q5,q7,q10},{c4})
r13 = ({c1},{q2,q0,q3,q8,q5,q7,q4,q9},{p2})
r14 = ({c4},{q2,q15,q0,q1,q13,q6,q5,q7,q4,q9,q12,q10},{c3})
r15 = ({c1},{q2,q3,q13,q14,q8,q11,q7,q10},{c4})
r16 = ({c2},{q2,q0,q1,q13,q6,q5,q7,q10},{c3})
r17 = ({c4},{q15,q4,q9,q12},{c2})
-------------------------------------------
The number of all non-trivial regions is 18.
The number of non-trivial regions after applying Reduction Rule 2 is 8.
The number of non-trivial regions after applying Reduction Rule 3 is 8.
The number of non-trivial regions after applying Reduction Rule 1 is 7.
The number of non-trivial regions after applying reduction rules 2, 3, and 1 is 7.

[INFO] The given ts is isomorphic to the generated ts_{enl^{≙}_{ts}} w.r.t. ≙ =
 {p1,p2} × {p1,p2} ∪ {c3,c4,c1,c2} × {c3,c4,c1,c2}.
So, it can be synthesised to an ENL_{≙}-system and the reduction rules can be
applied.
```

Fig. 8.25 An example of a successful synthesis for the ST-system $\mathfrak{ts}_3$ in Figure 3.1 to a small (minimised) ENL-system.

8.3 Availability

The latest version of SYNTHCRAFT is available in [44]. There is no automatic installer for the tool; the files need to be extracted manually from the link. In addition, we provided all examples of ENL-systems and ST-systems they have been used throughout our research in [45] so that can be used to test our tool.

Chapter 9

Conclusions

The aim of our research was to develop algorithms and tool support for the synthesis of ENL-systems. In our work, we mainly concentrated on the general class of ENL-systems. However, in addition, we also considered a synthesis problem for a subclass of ENL-systems, where conflicts between events are localised. The synthesis problem, for any class of Petri nets that might be considered, has always two aspects: a feasibility problem and an effective construction problem. While the feasibility problem is about finding a complete characterisation of the transition systems that can be realised by Petri nets of a particular class, the effective construction problem is about finding effective algorithms for deriving Petri nets from transition systems. In our research, we concentrated on solving the effective construction problems for ENL-systems; the solution to the feasibility problem for ENL-systems was given in [25], by axiomatising the class of ST-systems that can be realised by ENL-systems (see Definition 2.1.7).

In this thesis, we presented an algorithm for deriving non-trivial regions of ST-systems, which is a fundamental algorithm for the synthesis of ENL-systems. We presented two algorithms for verifying whether a given step transition system can be synthesised to an $\mathrm{ENL}_{\bumpeq}$-system and provided results to compare the efficiency of the two approaches for the synthesis procedure. Also, we presented algorithms for synthesising ENL/LC-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. We discussed the minimisation of the synthesised ENL-systems and the strategy to eliminate redundant regions that involves three reduction rules. Also, we investigated the properties of minimal regions that play a crucial role in the minimisation process. We showed that synthesised and minimised nets that are based on all minimal regions (after the application of the Reduction Rule 2) do not lose the property of the saturated ENL-systems of being state machine decomposable. Furthermore, we built a set of benchmarks to test the performance efficiency of our developed algorithms on bigger examples and see how they scale with the increasing sizes of ST-systems. We described the SYNTHCRAFT tool that implements our designed algorithms for the synthesis of ENL-systems. Through the SYNTHCRAFT tool, we introduced two plugins into WORKCRAFT framework to represent ENL-systems and ST-systems, two kinds of graphs, which are involved in the the synthesis procedure we consider.

9.1 Directions for future work

There are several directions for extending or improving the results presented in this thesis.

The main achievement of our research, Algorithm 1, is used in many other algorithms of the SYNTHCRAFT tool, as extracting non-trivial regions from an ST-system is the fundamental task in the synthesis procedure. Therefore, its performance has a considerable bearing on the performance of the other algorithms of the tool. In the future, we plan to improve further the performance of Algorithm 1 by parallelising its code.

In Chapter 5 we discussed the synthesis procedure for ENL/LC-systems. When researching this subject, we identified two transition systems, $\mathfrak{ts}_6$ and $\mathfrak{ts}_7$ (respectively in Figure 5.1(a) and Figure 5.2(a) on p. 50), which cannot be synthesised to any ENL/LC-system. These ST-systems have an interesting characteristic. The conflict between two of their events, which do not share a locality, is ‘witnessed’ by a third event that is independent (not in conflict) from one of them. We plan to generalise this observation to give sufficient and necessary conditions for deciding that an ST-system cannot be synthesised to an ENL/LC-system.

Many ideas discussed in Chapter 6 about the minimisation of the synthesised nets, can be also investigated further. We proved that the saturated nets, which result from the synthesis procedure, and their minimised versions obtained after the application of the Reduction Rule 2 (which deletes non-minimal regions) are state machine decomposable. We believe that after the application of the Reduction Rule 3 this property still holds for the resultant net. However, this result still needs to be formally proved. We could observe that after applying Reduction Rule 1, some synthesised ENL-systems are no longer state machine decomposable. We can take, for example, the ENLST-system generated by the ENL-system in Figure 1.1, on p. 4. The synthesis procedure for this example will produce a saturated net that has only minimal regions (12 regions). Some of them are redundant and can be deleted according to Reduction Rule 1. Figure 1.1 shows one of the possible minimised versions of this net that is not state machine decomposable. Also, talking about Reduction Rule 1, we want to develop an improved algorithm implementing this rule, which would allow to target certain regions for deletion from the pairs of complementary regions. Furthermore, in relation to state machine decomposability of ENL-systems, we plan to investigate the relationship between the split of ENL-systems into state machine components (based on conditions) and the split into localities (based on events).

Furthermore, there are possibilities for improving the developed SYNTHCRAFT tool. WORKCRAFT provides a feature to check a trace leading to a deadlock in a model such as Signal Transition Graph (STG). We plan to provide similar feature in our tool for ENL-systems.

Finally, we plan to keep track of new graph models available within the WORKCRAFT framework that might be ‘compatible’ with our developed SYNTHCRAFT tool. Recently, a new model of Burst Automata (BA) was implemented within WORKCRAFT [15]. BAs model the behaviour of asynchronous circuits, where signals (events) are allowed to change in groups called ‘bursts’. This is reflected in the definition of BAs, where they are represented as a kind of step transition systems. It would be interesting to research whether SYNTHCRAFT can be adopted to allow for the synthesis of BAs.

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] Ahmed, A.: Case Studies Document. Unpublished manuscript.

[3] Ahmed, A., Koutny, M., Pietkiewicz-Koutny, M.: Synthesising Elementary Net Systems with Localities. Theoretical Computer Science **908** (2022) 123–140.

[4] Ahmed, A., Pietkiewicz-Koutny, M.: Algorithms for the Synthesis of Elementary Net Systems with Localities. In: Köhler-Bussmeier, M., Kindler, E., Rölke, H. (eds.): Proc. of the International Workshop on Petri Nets and Software Engineering (PNSE'2020) vol. **2651** CEUR, 86-107.

[5] Ahmed, A., Pietkiewicz-Koutny, M.: Minimising the synthesised ENL-systems. In: Lorenz, R., van der Werf, J.M., van Zelst, S.J. (eds.): Proc. of the International Workshop on Algorithms & Theories for the Analysis of Event Data (ATAED 2022) vol. 3167 CEUR, 43-59.

[6] Badouel, E., Bernardinello, L., Darondeau, Ph.: The Synthesis Problem for Elementary Net Systems is NP-complete. Theoretical Computer Science **186** (1997) 107–134.

[7] Badouel, E., Bernardinello, L., Darondeau, P.: Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series. Springer (2015).

[8] Badouel, E., Caillaud, B., Darondeau, P.: Distributing finite automata through Petri net synthesis. Formal Aspects of Computing, 13(6), (2002) 447-470.

[9] 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.

[10] Bergenthum, R.: Prime Miner - Process Discovery using Prime Event Structures. Proc. of the International Conference on Process Mining (ICPM 2019), IEEE 2019, 41–48

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] Chan, A., Sokolov, D., Khomenko, V., Lloyd, D., Yakovlev, A.: Burst automaton: Framework for speed-independent synthesis using burst-mode specifications. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. (2022), doi: 10.1109/TCAD.2022.3206732.

[16] 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.

[17] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Logic Synthesis of Asynchronous Controllers and Interfaces. Springer (2002).

[18] 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.

[19] 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.

[20] Desel, J., Reisig, W.: The Synthesis Problem of Petri Nets. Acta Informatica **33** (1996) 297–315.

[21] 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.

[22] Fernandes, J., Koutny, M., Mikulski, Ł., Pietkiewicz-Koutny, M., Sokolov, D. and Yakovlev, A.: Persistent and nonviolent steps and the design of GALS systems. Fundamenta Informaticae, **137** (2015) 143–170.

[23] Kishinevsky, M., Kondratyev, A., Taubin, A., Varshavsky, V.: Concurrent hardware: the theory and practice of self-timed design. John Wiley & Sons, Inc., (1994).

[24] 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.

[25] 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.

[26] Koutny, M., Pietkiewicz-Koutny, M.: Synthesis of Elementary Net Systems with Context Arcs and Localities. Fundamenta Informaticae **88** (2008) 307–328.

[27] Koutny, M., Pietkiewicz-Koutny, M.: Synthesis of Petri Nets with Localities. Scientific Annals of Computer Science **19** (2009) 1–23.

[28] Koutny, M., Pietkiewicz-Koutny, M.: Minimal regions of ENL-transition systems. Fundamenta Informaticae **101(1-2)** (2010) 45–58.

[29] Khomenko, V., Sokolov, D., Yakovlev, A. and Lloyd, D.: Handshake Verification in WORKCRAFT. In 2020 26th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC) (2020) 63-64. IEEE.

[30] Li, B., Randell, B., Bhattacharyya, A., Alharbi, T. and Koutny, M.: Soncraft: A tool for construction, simulation, and analysis of structured occurrence nets. In 2018 18th International Conference on Application of Concurrency to System Design (ACSD) (2018) 70–74. IEEE.

[31] Mokhov, A. and Yakovlev, A.: Conditional partial order graphs: Model, synthesis, and application. IEEE Transactions on Computers, **59(11)**, (2010) 1480-1493.

[32] Mannel, L.L., Bergenthum, R., van der Aalst, W.M.P.: Removing Implicit Places Using Regions for Process Discovery. Proc. of the International Workshop on Algorithms & Theories for the Analysis of Event Data (ATAED 2020), CEUR Workshop Proceedings, vol. **2625**, 20–32

[33] Mukund, M.: Petri Nets and Step Transition Systems. International Journal of Foundations of Computer Science **3** (1992) 443–478.

[34] Nielsen, M., Rozenberg, G., Thiagarajan, P.S.: Elementary Transition Systems. Theoretical Computer Science **96** (1992) 3–33.

[35] Păun, G.: Membrane Computing, An Introduction. Springer-Verlag, Berlin Heidelberg New York (2002).

[36] Pietkiewicz-Koutny, M.: Synthesis of ENI-systems Using Minimal Regions. In: 9^{th} International Conference on Concurrency Theory (CONCUR'1998) 565–580.

[37] Pietkiewicz-Koutny, M.: The Synthesis Problem for Elementary Net Systems with Inhibitor Arcs. Fundamenta Informaticae **40** (1999) 251–283.

[38] Poliakov, I.: Interpreted graph models. PhD Thesis, Newcastle University (2011).

[39] 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.

[40] Solé, M., Carmona, J.: Rbminer: A Tool for Discovering Petri Nets from Transition Systems. Lecture Notes in Computer Science **6252**, Springer (2010) 396–402.

[41] Sokolov, D., Khomenko, V. and Mokhov, A.: Workcraft: Ten years later. This asynchronous world. Essays dedicated to Alex Yakovlev on the occasion of his 60th birthday, (2016) 269–293.

[42] Sokolov, D., Khomenko, V., Mokhov, A., Dubikhin, V., Lloyd, D. and Yakovlev, A.: Automating the design of asynchronous logic control for AMS electronics. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **39(5)**, (2019) 952-965.

[43] Sokolov, D., Khomenko, V., Yakovlev, A. and Lloyd, D.: Design and Verification of Speed-Independent Circuits with Arbitration in WORKCRAFT. In 2018 24th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC) (2018) 30–31. IEEE.

[44] SYNTHCRAFT code page. SYNTHCRAFT-TOOL.com

[45] SYNTHCRAFT examples. SYNTHCRAFT-TOOL-EXAMPLES.com

[46] The Eclipse Foundation — http://www.eclipse.org/.

[47] The Rt Hon Lord Cullen PC. The Ladbroke Grove rail inquiry, 2001.

[48] WORKCRAFT. https://workcraft.org/

[49] WORKCRAFT. SELECTION-CONTROLS.com.

[50] WORKCRAFT. PROPERTY-EDITOR.com.

[51] WORKCRAFT. SIMULATION-CONTROLS.com.

[52] Yakovlev, A., Lavagno, L. and Sangiovanni-Vincentelli, A.: A unified signal transition graph model for asynchronous control circuit synthesis. Formal Methods in System Design, **9(3)**, (1996) 139–188.