CARL VON OSSIETZKY universität OLDENBURG

Characterisation of a Class of Petri Net Solvable Transition Systems

Von der Fakultät für Informatik, Witschafts- und Rechtswissenschaften
der Carl von Ossietzky Universität Oldenburg
zur Erlangung des Grades und Titels eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

angenommene Dissertation

von Herrn Evgeny Erofeev
geboren am 22.07.1990 in Komsomolsk-am-Amur

Gutachter: Prof. Dr. Eike Best, Carl von Ossietzky Universität Oldenburg
Prof. Dr. Lucia Pomello, Università degli studi di Milano-Bicocca

Tag der Disputation: 20. April 2018

Abstract

The problem of Petri net synthesis is widely studied in the literature, being set in a range of contexts. At the same time, the question for a characterisation of synthesisable state spaces, which is related to synthesis, seems to be less investigated and developed. The current work is an attempt to put more insight into the latter, keeping in mind the former as the main goal. We study a relatively restricted class of transition systems representing the reachability graph of the sought Petri net. Looking at the case with only two transitions allows to obtain notable results, not only at characterising of state spaces but also at synthesising of the Petri net.

Among such results is a language-theoretical characterisation of finite binary sequences and cycles synthesisable into a Petri net. Synthesis algorithms, based on the characterisation, demonstrate a better runtime in comparison to the region based synthesis.

A classification for a complete enumeration of minimal unsolvable binary words is presented. This classification gives rise to a characterisation for the class of all minimal unsolvable words in the form of extended regular expressions. A procedure for pre-synthesis, which recognises failures early and uses the characterisation, demonstrates good results for rejecting a given input, without having to start the synthesis process itself.

For Petri nets over the binary transition set, a graph-theoretical characterisation of their reachability graphs is presented in the form of state spaces which are geometrically convex sets. The characterisation relies on the notion of generalised cycles. Based on this characterisation and the absence of Parikh-non-zero g-cycles, an algorithm for over-approximating a finite language by a Petri net language is suggested.

Zusammenfassung

Das Problem der Petrinetzsynthese wurde schon umfassend untersucht und passt in viele Kontexte. Gleichzeitig scheint die Frage nach einer Charakterisierung von synthetisierbaren Zustandsräumen, welche mit der Synthese zusammenhängt, weniger untersucht und entwickelt zu sein. Die vorliegende Arbeit ist ein Versuch mehr Einsicht für letzteres zu geben, während ersteres als Hauptziel im Auge behalten wird. Wir untersuchen eine relativ eingeschränkte Klasse von Transitionssysteme, die den Erreichbarkeitsgraph des gesuchten Petrinetzes repräsentieren. Die Einschränkung auf den Fall mit nur zwei Transitionen erlaubt es nennenswerte Resultate zu erzielen, nicht nur bei der Charakterisierung von Zustandsräumen, sondern auch bei der Synthese von Petrinetzen.

Zu diesen Resultaten gehört eine sprach-theoretische Charakterisierung von endlichen, binären Sequenzen und Kreisen, die in Petrinetze synthetisierbar sind. Auf dieser Charakterisierung basierende Synthesealgorithmen zeigen eine bessere Laufzeit im Vergleich zur Regionen-basierte Synthese.

Eine Klassifizierung zur vollständigen Aufzählung aller minimal-unlösbaren, binärer Wörter wird vorgestellt. Diese Klassifizierung führt zu eine Charakterisierung der Klasse aller minimal-unlösbaren Wörter in Form eines erweiterten regulären Ausdrucks. Eine Prozedur zur Präsynthese, die Fehlschläge früh erkennt und die Charakterisierung verwendet, demonstriert gute Resultate darin eine gegebene Eingabe abzulehnen, ohne dass der Synthese-Prozess selbst gestartet werden muss.

Für Petrinetze über der binären Transitionsmenge wird eine Graph-theoretische Characterisierung ihrer Erreichbarkeitsgraphen vorgestellt. Die Charakterisierung setzt auf das Konzept eines generalisierten Kreises auf. Basierend auf dieser Charakterisierung und dem Fehlen von Parikh-nicht-null g-Kreisen wird ein Algorithmus zur Überapproximation einer endlichen Sprache durch eine Petrinetzsprache vorgeschlagen.

Acknowledgements

First and foremost I would like to thank my supervisor, Eike Best, whose continuous support, kindness and constructive criticism were substantial inspiration sources for me, and without whose guidance this thesis would not have been written. For the many productive discussions, I am very thankful to Uli Schlachter, Harro Wimmel, Thomas Hujsa, Valentin Spreckels, who were always open to all sorts of questions.

A great thank you goes to Kamila Barylska, Łukasz Mikulski and Marcin Piątkowski. I have benefited a lot from seminal collaborations with them.

I would furthermore like to thank Lucia Pomello for taking over the co-refereeing, for valuable comments and the discussion on my thesis.

I thank Ernst-Rüdiger Olderog, Raymond Devillers, Irina Virbitskaite and Maciej Koutny for their interest to the work on different stages and the feedbacks.

In Oldenburg I experienced a pleasant working environment which was created by many people together (in alphabetical order): Marion Bramkamp, Björn Engelmann, Nils-Erik Flick, Manuel Gieseking, Andrea Göken, Annegret Habel, Martin Hilscher, Sven Linker, Heinrich Ody, Christoph Peuser, Maike Schwammberger, Thomas Stratmann, Mani Swaminathan, Ira Wempe, Elke Wilkeit.

On the non-scientific side, I am immensely thankful to my family for their encouragement and patience.

Contents

Chapter 1

Introduction

The relationship between a Petri net and its reachability graph can be viewed from a system analysis or from a system synthesis viewpoint. In system analysis, a system could, for instance, be modelled by a marked Petri net whose (unique up to isomorphism) reachability graph serves to facilitate its behavioural analysis [Rei13]. We may get various kinds of interesting structural results for special classes of Petri nets. For example, if the given system is described by a marked graph, then its reachability graph enjoys a long list of useful properties (e.g., in [CHEP71] the authors establish that for every finite, strongly connected graph there exists a live and safe marking, and that live markings can be partitioned into equivalence classes). In system synthesis, a behavioural specification is typically given, and a system implementing it is sought. For example, one may try to find a Petri net whose reachability graph is isomorphic to a given edge-labelled directed graph (or a labelled transition system) [BBD15]. We may get structural results of a different nature in this case. For instance, in [BD14b] the authors describe a complete structural characterisation of the class of marked graph reachability graphs in terms of a carefully chosen list of graph-theoretical properties (e.g. reversibility, persistency, uniformity of small cycles etc.).

In this work, we investigate labelled transition systems which are finite and totally reachable (and satisfy some other necessary properties like persistency), and consider some special classes thereof. The ultimate aim is to characterise, graph-theoretically, exactly which ones of them are synthesisable into an unlabelled place/transition Petri net [Mur89]. Such a characterisation is difficult and

has not yet been achieved in general. We begin to study the problem by restricting attention to a limited special case: non-branching, linearly ordered transition systems having at most two edge labels, and continue by involving cycle-shaped transition systems, and also some branching representatives thereof in the later sections. That is, we study the class of binary sequences, cycles and finite labelled transition systems over binary set of labels, and our aim is to characterise the Petri net synthesisable ones amongst them.

The theory of regions [BD98] provides an indirect characterisation of this class by means of an algorithm based on solving systems of linear inequalities and synthesising a Petri net if possible. Our aim is to initiate a combinatorial approach and to provide a complete characterisation of a generative nature for a special kind of labelled transition systems. In this work, we describe two alternative, more direct, characterisations, and provide proofs of their validity. The first condition characterises the class of Petri net synthesisable binary words in terms of a letter-counting relationship. The second condition characterises the same class in terms of a pseudo-regular expression, characterising also minimality of the sequences. Both conditions seem to be more efficient to check feasibility of synthesis than by using the general synthesis algorithm.

Furthermore, we proceed with a characterisation of synthesisability of finite labelled transition systems of an arbitrary shape with a binary label set, and figure out what kind of shape those can have which are produced by unlabelled Petri nets.

The further content of the present thesis is structured as follows. In Chapter 2 the main concepts of Petri nets, labelled transition systems and synthesis of Petri nets from transition systems are introduced. An overview of the general tool for Petri net synthesis – the theory of regions – is made, demonstrating the weak connection between the graph-theoretical properties of a labelled transition system and its synthesisability into a Petri net. With the aim to find a structural characterisation of solvable transition systems, we restrict our attention to a special class thereof: linear non-branching with a binary set of labels. This class is associated in a natural way with the class of finite sequences over two-letter alphabet, whose language-theoretic properties are then gradually studied with the focus on solvability by Petri nets [BBE$^+$15, BBE$^+$16]. This yields a criterion for solvability of the class of binary sequences (see also [BESW16]) and hence for the

class of linear transition systems. An algorithm for synthesis of Petri nets from transition systems of this class is presented. The characterisation, as well as the algorithm, are extended to the class of cyclical labelled transition systems.

Chapter 3 deals mainly with the concept of a minimal unsolvable word, and has as its main goal to provide a complete characterisation of minimal unsolvable binary sequences. The characterisation is presented in the form of (a pair of reciprocal) morphisms (see [EBMP16]), which demonstrate the generic nature of minimal unsolvable sequences. The interest for such a characterisation is justified by the search for a quick-fail procedure which allows to check solvability of given sequences without initiating the process of synthesis itself. The algorithm for such an examination of a sequence is described.

Chapter 4 tackles the problem of solvability of labelled transition systems over a binary set of labels in a more general context. The property of existence of (generalised) cycles with non-zero Parikh vectors in a transition system is crucial for the chapter. For the class of transition systems with such a cycles the classification of possible shapes is presented [EW17], with the corresponding Petri nets as a possible synthesis output. For the class of transition systems without such cycles we present a geometrical characterisation of Petri net solvable among them, which relies on the notion of a convex hull introduced for the set of states. This characterisation yields an algorithm for the minimal over-approximating of a finite language with a Petri net language, which is presented in the end of the chapter.

A summary of the results of the work and concluding remarks are given in Chapter 5.

Chapter 2

Synthesisability of simple labelled transition systems

This chapter explores various questions of the synthesisability of labelled transition systems having uncomplicated shapes with Petri nets. Among those we are mostly focused on linear transition systems having two labels and transition systems involving cycles in their structure with the same restriction. Starting from the basic definitions of the main formalisms – labelled transition systems and Petri nets – used throughout the text, and explanations of their properties, generic attributes and possible interrelations, we will introduce the task of synthesis of a Petri net from a given formal specification which is in our case represented as a labelled transition system. We will have a brief look over the general approach for the Petri net synthesis – the theory of regions, which establishes a characterisation for a labelled transition system to be isomorphic to the reachability graph of some Petri net. After the consideration of this characterisation we will initiate a discussion about possible shapes of synthesisable transition systems of a relatively simple class mentioned above. A number of necessary and sufficient conditions will be presented with the corresponding proofs, which give rise to the algorithms for constructing a Petri net synthesising a given transition system. The algorithms happen to be more efficient than the general algorithm which directly relies on the theory of regions. The price of this runtime improvement is the range of applicability of the suggested algorithms – the restricted size of the set of labels and the shape of the initial transition system. The contents of the chapter are

oriented on [BBE+15, BBE+16, BESW16] by the author and co-authors, where many results were first presented.

2.1 Basic notions and notations

Labelled transition systems

When modelling the behaviour of real systems there is often a need to describe sequential changing of configurations or states in the operating process of a given system in an abstract manner. One of the convenient tools for this purpose are (labelled) transition systems. A transition system consists of a set of states and a collection of transitions. The states of the transition system correspond to the possible configurations of the system under consideration, and the transitions describe how the system can switch from one state to some other ones. For example, one can describe a chess game by a transition system whose set of states is the set of all possible placements of pieces on the board, and the set of transitions is determined by the legal moves of all the pieces. Depending on the context, a transition system is often equipped with a labelling function which assigns a label to each transition, and possibly with an initial state which defines the starting configuration of the system. In case of a chess game a suitable labelling function is the one which assigns a particular piece to a transition which represents a move of this piece, and for the initial state the natural candidate is the starting placement of pieces on the board.

Definition 1. *A* labeled transition system *with initial state, abbreviated lts, is a tuple* $(S, T, \to, s_0)$ *where*

- S *is a nonempty set of* states *(nodes),*
- T *is a set of* labels *with* $T \cap S = \emptyset$*,*
- $\to \subseteq S \times T \times S$ *is the* transition relation *(set of edges),*
- $s_0 \in S$ *is an* initial state*.*

A label t is enabled *(or* activated*, or* fireable*) at state $s \in S$, denoted by $s[t\rangle$, if there exists some state $s' \in S$ such that $(s,t,s') \in \rightarrow$. Instead of $(s,t,s') \in \rightarrow$ we also use the notation $s[t\rangle s'$ to denote that s' is* reachable *from s through the execution of t. A state s' is* reachable *from state s through the execution of a sequence of labels $\sigma \in T^*$, denoted by $s[\sigma\rangle s'$, if there is a directed path from s to s' whose edges are labelled consecutively by σ. The set of states reachable from s is denoted by $[s\rangle$. A sequence $\sigma \in T^*$ is* allowed *(or* firable*) from a state s, denoted by $s[\sigma\rangle$, if there is some state s' such that $s[\sigma\rangle s'$. For clarity, in case of long formulas we write $|_r\, \alpha\, |_s\, \beta\, |_q$ instead of $r\ [\alpha\rangle\ s\ [\beta\rangle\ q$.*

An lts is usually graphically represented as a directed graph. For a triple (s,t,s') with $s,s' \in S$ and $t \in T$, an arrow labelled by t is drawn from s to s'. Later, the set T will correspond exactly to the set of transitions of a Petri net. This is why the letter T is used for arc labelings. Note however that there could be many different arrows (s,t,s'), (r,t,r'), ..., all with the same label t, denoting different state-to-state changes effected by a single transition, t.

To avoid a source of confusion that is not uncommon, we mention that the triples $(s,t,s') \in S \times T \times S$ of an lts are often called "transitions" in the literature. However, such triplets should not be confused with the transitions $t \in T$ of a Petri net. When speaking of *labels*, or *letters*, or *transitions*, elements of the set T (and thus the transitions of a Petri net, or the labels of an lts) are meant. By contrast, triples such as (s,t,s') will not normally be given any special names in this text (except perhaps calling them *arcs*, or *arrows*). They usually occur as the building block of paths in an lts.

The following properties of labelled transition systems will be important for our following consideration.

Definition 2. *A labelled transition system $(S,T,\rightarrow,s_0)$ is called*

- totally reachable *if $[s_0\rangle = S$, i.e, every state is reachable from the initial state,*
- finite *if S and T (hence also $\rightarrow$) are finite sets,*
- label-deterministic*, or just* deterministic*, iff, for all $s \in [s_0\rangle$ and for all $t \in T$, if $s[t\rangle s'$ and $s[t\rangle s''$ then $s' = s''$, i.e., from any state, the same label may not lead to two different successor states.*

Example 1. *In the left of Fig. 2.1 an example of a finite, totally reachable labelled transition system*

$$TS = (\{s_0, \ldots, s_3\}, \{a, b, c\}, \{(s_0, a, s_1), (s_0, c, s_2), (s_1, b, s_2), (s_1, b, s_3), (s_3, a, s_0)\}, s_0)$$

with the initial state s_0 and having three labels, a, b and c, is depicted. This transition system is not deterministic since $s_1[b\rangle s_2$, $s_1[b\rangle s_3$ and $s_2 \neq s_3$. In the right of the figure, the arrow (s_1, b, s_3) is dropped. Since there is no path $\sigma \in \{a, b, c\}^*$ such that $s_0[\sigma\rangle s_3$, the new transition system TS' is no longer totally reachable, although it is deterministic.

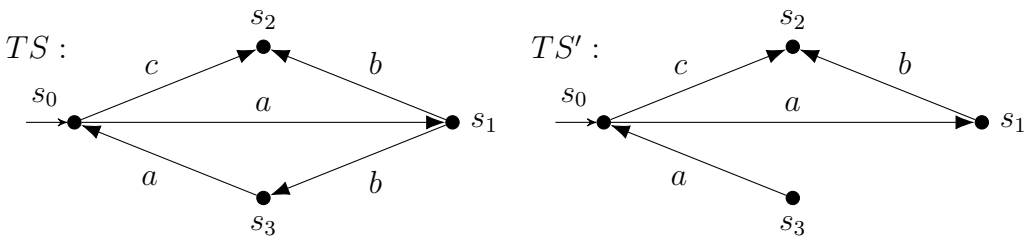


Figure 2.1: Finite, totally reachable, non-deterministic lts (left); finite, deterministic, not totally reachable lts (right).

In what follows, when talking about labelled transition systems, the properties from Definition 2 will be satisfied all of the time, unless we explicitly claim the opposite.

Petri nets

Petri nets are a well-known formalism, which is widely used for modelling and investigating parallel and distributed systems. A Petri net consists of places, transitions, and a flow function connecting them. In a graphical representation places are usually shown as hollow circles, transitions as squares, and a flow function as directed arcs with weights.

Definition 3. *A* finite Petri net *is a tuple (P, T, F) such that*

- *P is a finite set of* places*,*

- *T is a finite set of* transitions*, with $P \cap T = \emptyset$,*
- *F is a* flow *function $F\colon ((P \times T) \cup (T \times P)) \to \mathbb{N}$, specifying the arc weights.*

According to the definition, in a Petri net arcs can connect a transition with a place, or a place with a transition, but never two places or two transitions. Places of a Petri net can carry tokens on them, and the particular distribution of tokens over the places determines the (global) state of a Petri net. This distribution of tokens can be formally defined as a function, which corresponds a natural number to each place, and is called *marking of a Petri net*. An *initial marking* of a Petri net determines the initial state of the model, and a Petri net is called *initially marked* if it is equipped with an initial marking.

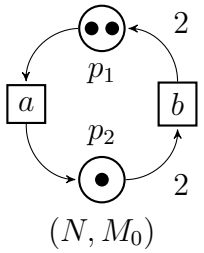
Definition 4. *A* marking *M of a Petri net (P, T, F) is a function $M\colon P \to \mathbb{N}$. An* initially marked *Petri net is a 4-tuple (P, T, F, M_0), where (P, T, F) is a Petri net and M_0 is the* initial marking*.*

Let us note that the total number of tokens in all places of a Petri net in general changes by the execution of the transitions.

Example 2. *The Petri net*

$$N = (\{p_1, p_2\}, \{a, b\}, \{((p_1, a), 1), ((a, p_2), 1), ((p_2, b), 2), ((b, p_1), 2)\})$$

on the left of Fig. 2.2 has two places p_1 and p_2, and two transitions a and b. Place p_1 has initially two tokens on it, while p_2 has only one token, so the initial marking of N is $M_0(p_1, p_2) = (2, 1)$.



$$M_0 \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Figure 2.2: Petri net N with its initial marking M_0.

Transitions of a Petri net represent activities (or actions) within the modelled system. For a transition, a place is *an input place* if there is a non-zero-weighted arc from the place to the transition, and *an output place* – if there is a non-zero-weighted arc from the transition to the place. A place is said to be a *side-condition* for a given transition, if it is an input and an output place for the transition.

Definition 5. *For a given transition $t_0 \in T$ of net $N = (P, T, F)$, place $p \in P$ is called an* input place for t_0 *iff $t_0 \in p^\bullet$, where $p^\bullet = \{t \in T \mid F(p,t)>0\}$, an* output place for t_0 *iff $t_0 \in {}^\bullet p$, where ${}^\bullet p = \{t \in T \mid F(t,p) > 0)\}$. Place $p \in P$ is a* side-place *(or* side-condition*) iff $p^\bullet \cap {}^\bullet p \neq \emptyset$, and is a* side place for t_0 *iff $t_0 \in p^\bullet \cap {}^\bullet p$. Petri net N is called* pure *or* side-place free *if it has no side-places. If for every $p \in P$ it holds that $|p^\bullet| \leq 1$, net N is called* output-non-branching *(or* ON-net*).*

The occurrence of an action in the modelled system is associated with the firing of the corresponding transition of a Petri net representing this system. While firing, a transition consumes tokens from its input places (as many as the arc-weights require) and produces tokens on its output places (also, according to the arc-weights). So, firing a transition can change the marking (the distribution of tokens over places) of a Petri net, i.e. change the global state of the whole model. This switching between markings is called the *token game*, and it is regulated by the *firing rule*. The firing rule allows a transition for firing only if each of its input places have not less tokens on them than the weight of the arc from the place to this transition. The new marking, achieved after the firing of a transition, is defined for each place as the sum of the previous marking and the difference between arc-weights from the transition to the place and from the place to the transition.

Definition 6. *In a given initially marked Petri net $N = (P, T, F, M_0)$, a transition $t \in T$ is* enabled for firing *(or simply* enabled*) at a marking M, denoted by $M[t\rangle$, if $M(p) \geq F(p,t)$ for all places $p \in P$. The* firing *of t leads from marking M to the marking M', denoted by $M[t\rangle M'$, if $M[t\rangle$ and $M'(p) = M(p) + F(t,p) - F(p,t)$ for all places $p \in P$. This can be inductively extended to $M[\sigma\rangle M'$ for sequences of transitions $\sigma \in T^*$.*

For the transitions of a Petri net the notion of *effect* can be introduced. The effect of a transition is a function which for every place of the net determines how the marking of the place is changing while firing the transition.

Definition 7. *In a given initially marked Petri net $N = (P, T, F, M_0)$, the* effect of a transition $t \in T$ *for place $p \in P$ is the function which assigns an integer value for each p and t as follows $\mathbb{E}(p)(t) = F(t, p) - F(p, t)$. In what follows we will also use the notation $\mathbb{E}_p(t)$ instead of $\mathbb{E}(p)(t)$. We can extend the notion of effect $\mathbb{E}$ for a place p to a sequence $\tau \in T^*$. The effect of the empty sequence ϵ is $\mathbb{E}_p(\epsilon) = 0$. The effect of a sequence $t\tau$ with $t \in T$ is defined as $\mathbb{E}_p(t\tau) = \mathbb{E}_p(t) + \mathbb{E}_p(\tau)$.*

Let us note that since the effect is completely defined by the flow function of the Petri net, it does not give any new information about the net but serves mostly for the sake of convenience. On the other hand, one cannot substitute the flow function by the effect in general, due to possible side-conditions.

Markings (or states) of a given Petri net differ from each other by the number of tokens assigned to places by them. For an initially marked Petri net, a marking is called *reachable*, if it can be achieved from the initial marking by the token game. Markings of a Petri net allow to investigate behavioural aspects of the Petri net, and hence the modelled system – among such aspects are problems of safety [Yen91, CEP93], reachability [Kos82, Lam92, May81, Esp98], reversibility [KPP06, ÖA08, WDC11, HDK15], behavioural homogeneity [CHEP71], existence of a home-state [BE16, BS15, BV84] or checking for it [AK77], etc.

Definition 8. *For the initially marked Petri net $N = (P, T, F, M_0)$, and its markings M and M', M' is called* reachable *from M, if there is a sequence $\sigma \in T^*$ such that $M[\sigma\rangle M'$. $[M\rangle$ denotes the set of all markings reachable from a given marking M. $[M_0\rangle$ denotes the set of all markings reachable in net N.*

Example 3. *In the initially marked Petri net (N, M_0) (see the top left of Fig. 2.3) transition a is enabled for firing, since its input place p_1 has two tokens on it, which is greater than the weight of the arc from p_1 to a: $M_0(p_1) = 2 \geq 1 = F(p_1, a)$. At the same time, transition b is not enabled (it is* disabled*) at marking M_0, because place p_2, which is an input place for b, has a single token on it, and the weight of the arc from p_2 to b is equal to 2: $M_0(p_2) = 1 \ngeq 2 = F(p_2, b)$. By firing, transition a consumes $F(p_1, a) = 1$ token from its input place p_1 and produces $F(a, p_2) = 1$ token on its output place p_2. After the firing of the transition a at marking M_0, Petri net N reaches the new marking $M_1(p_1, p_2) = (1, 2)$ (see the bottom right of Fig. 2.3). At marking M_1, both transitions a and b are enabled for firing, since $M_1(p_1) = 1 \geq 1 = F(p_1, a)$ and $M_1(p_2) = 2 \geq 2 = F(p_2, b)$. By firing a or b*

from the marking M_1, Peri net N reaches the marking $M_2 = (0, 3)$ or $M_3 = (3, 0)$, respectively (on the top right and bottom left of Fig. 2.3, resp.), from both of which the initial marking M_0 can be reached again.

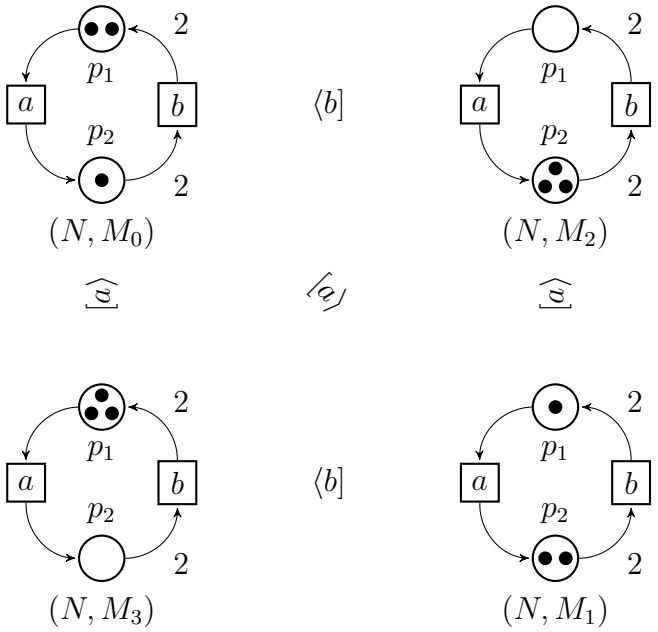


Figure 2.3: Petri net N with its initial marking M_0 (on the top left), and the markings M_2, M_1, M_3 (clockwise) reachable from M_0.

According to the definition of markings of a Petri net, the number of tokens on each place is always finite for any marking. It can happen that the token game leads to an indefinitely increasing of the amount of tokens on some place(s). If this is the case, such a Petri net is called *unbounded*. If for any reachable marking of a given Petri net, the number of tokens in each place does not exceed a certain finite number, the Petri net is called *bounded*.

Definition 9. *An initially marked Petri net $N = (P, T, F, M_0)$ is called* bounded *(or k-*bounded*) iff $M(p) \leq k$ for all $p \in P$, $M \in [M_0\rangle$, where $k \in \mathbb{N}$. If for every $k \in \mathbb{N}$ there exists $M \in [M_0\rangle$ and $p \in P$ such that $M(p) > k$, Petri net N is called* unbounded*. A 1-bounded Petri net is also called* safe*.*

Example 4. *Petri net N_u depicted in Fig. 2.4 is unbounded. Indeed, at the marking $M_0(p, q) = (1, 0)$ (on the left of the figure), transition t is enabled. After the*

firing of t, the number of tokens on place p remains unchanged, since t consumes one token from this place, and then produces one token on both p and q. The new marking reached from M_0 through the firing of t is $M_1(p,q) = (1,1)$ (see in the middle of the figure). Transition t is again enabled at the marking M_1, and $M_1[t\rangle M_2$, where $M_2(p,q) = (1,2)$ (on the right). So, consecutive firings of t leave the single token on place p and continuously increase the number of tokens on place q.

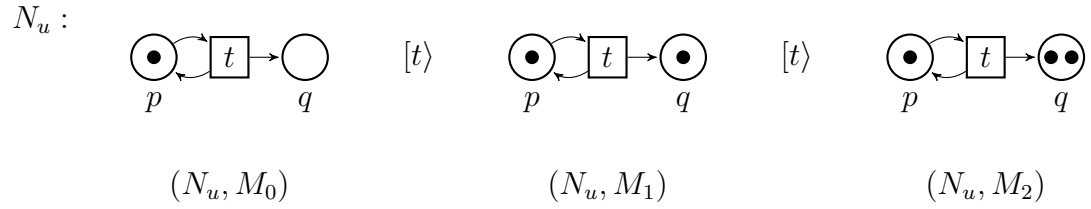


Figure 2.4: Unbounded Petri net N_u.

Since the number of places of a Petri net is finite, there is always only a finite number of reachable markings for a bounded initially marked Petri net. This allows to introduce a notion of a *reachability graph* of a Petri net, as a labelled directed graph whose set of nodes consists of all reachable markings of the Petri net, the set of labels is given by the set of transitions of the Petri net, and the set of edges is defined by enabledness of transitions at the corresponding markings.

Definition 10. *The* reachability graph *$RG(N)$ of a Petri net $N = (P, T, F, M_0)$ is the labelled transition system with the set of nodes $[M_0\rangle$, i.e., all markings reachable from M_0, the initial state M_0, i.e., the initial marking of N, the set of arc labels T, i.e., the set of transitions of N, and the set of edges $\{(M, t, M') \mid M, M' \in [M_0\rangle \wedge M[t\rangle M'\}$, i.e., there is an edge from M to M' labelled with t iff $M[t\rangle M'$.*

Petri net properties are classified into structural ones and behavioural ones. Structural properties are concerned with the sets P and T, their connection relation (the flow function) F, and possibly also the initial marking M_0. Behavioural properties are concerned with the reachability graph of the net. We will consider some properties of these kinds throughout the next sections.

Example 5. *The initially marked Petri net N from Fig. 2.2 has four reachable markings: $M_0 = (2,1)$, $M_1 = (1,2)$, $M_2 = (0,3)$, $M_3 = (3,0)$ (see Fig. 2.3). The reachability graph $RG = ([M_0\rangle, \{a,b\}, \{(M_0,a,M_1), (M_1,a,M_2), (M_1,b,M_3), (M_2,b,M_0), (M_3,a,M_0)\}, M_0)$ of N is shown on the left-hand side of Fig. 2.5. Its initial state is the initial marking M_0 of N, and the edge labels a and b define the changing of markings of N by firing the transitions a and b in the net.*

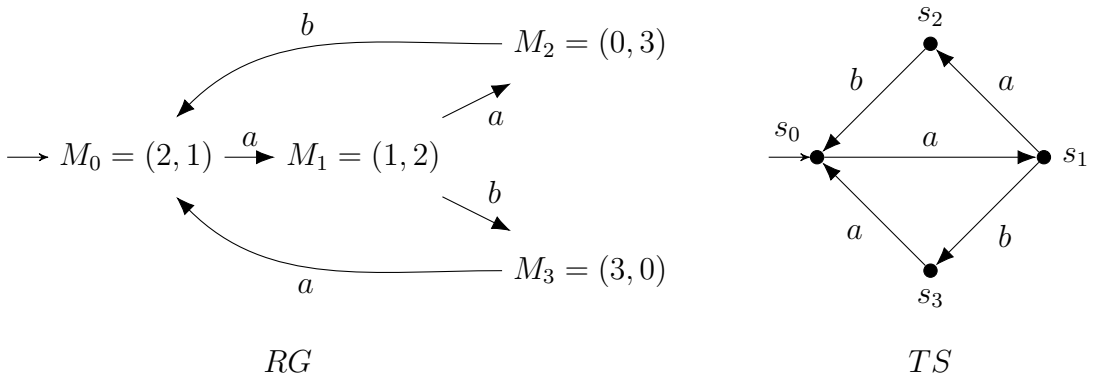


Figure 2.5: *RG* (on the left) is the reachability graph of the Petri net N (see Fig. 2.2), TS (on the right) is isomorphic to *RG*.

In what follows we will associate the reachability graph of the sought Petri net with the labelled transition system describing the desired behaviour. In order to clarify when they coincide, we introduce the notion of *isomorphism* of two transition systems, and we will not distinguish isomorphic transition systems unless the opposite is claimed.

Definition 11. *Two labelled transition systems $TS_1 = (S_1, T, \rightarrow_1, s_{01})$ and $TS_2 = (S_2, T, \rightarrow_2, s_{02})$ are* isomorphic *if there is a bijection $\zeta\colon S_1 \rightarrow S_2$ with $\zeta(s_{01}) = s_{02}$ and $(s,t,s') \in \rightarrow_1 \Leftrightarrow (\zeta(s), t, \zeta(s')) \in \rightarrow_2$, for all $s, s' \in S_1$.*

Consideration of reachability graphs up to isomorphism allows to abstract from precise markings of Petri nets, and to pay attention only to the behavioural aspects of nets and their state spaces, in particular admissible firing sequences.

Example 6. *For the transition system $TS = (\{s_0, \ldots, s_3\}, \{a,b\}, \rightarrow, s_0)\}$, where $\rightarrow = \{(s_0,a,s_1), (s_1,a,s_2), (s_1,b,s_3), (s_2,b,s_0), (s_3,a,s_0)\}$, on the right-hand side*

of Fig. 2.5 and the reachability graph RG on the left-hand side of Fig. 2.5, there is a bijection $\zeta : [M_o\rangle \to S$*, where* $\zeta(M_i) = s_i$ *for* $0 \leq i \leq 3$*. Since the set of labels of* TS *coincides with the set of transitions of* N*, and due to the fact that* $M_i[t\rangle M_j \iff (s_i, t, s_j) \in \to$*, for* $0 \leq i, j \leq 3$ *and* $t \in \{a, b\}$*, the transition system* TS *and the reachability graph RG are isomorphic. This means that in Petri net* N *in Fig. 2.2, for which RG is the reachability graph, we can fire exactly those sequences from* T^* *which are allowed in* TS*.*

Isomorphism between a labelled transition system and the reachability graph of some Petri net means that this net realises (or implements) the behaviour prescribed by the transition system. An lts for which such a Petri net exists is called *solvable* with a Petri net.

Definition 12. *If a labelled transition system* TS *is isomorphic to the reachability graph of a Petri net* N*, we say that* N PN-solves *(or simply* solves*)* TS*.*

Since RG and TS in Fig. 2.5 are isomorphic, the transition system TS is *solvable* e.g. by the Petri net N from Fig. 2.2. In the following sections and chapters we shall investigate the conditions under which a given labelled transition system is solvable (or unsolvable, i.e. there is no Petri net with the reachability graph isomorphic to the lts), and what structural shape solvable transition systems can have.

2.2 Theory of regions and separation problems

The theory of regions was introduced by Ehrenfeucht and Rozenberg in [ER90] where the authors investigate labelled partial (set) 2-structures, applications to the theory of concurrent systems and the problems of characterising and synthesising of state spaces of basic Petri net classes. Later the theory was gradually developed to be applicable for the range of nets from elementary nets to Petri nets (see [BD98]).

For a given labelled transition system each region consists of a set of functions from the sets of states and labels of the lts into the set of non-negative integers. The set of labels of the lts is used as the set of transitions the sought Petri net, and one needs to construct a suitable set of places. From each region a place of the sought Petri net can be constructed in a determined way. The functions of

a region are interconnected, which, being translated into the place with its arc weights, imposes restrictions for the behaviour of the net. Having enough many valid regions, one can transform them into the places and obtain the Petri net for the initial transition system.

Definition 13. *A* region *of a labelled transition system* $(S, T, \rightarrow, s_0)$ *is a triple of functions*

$$\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F}) \quad \in \quad (S \rightarrow \mathbb{N},\ T \rightarrow \mathbb{N},\ T \rightarrow \mathbb{N})$$

such that for every arrow $s[t\rangle s'$ *with* $s \in [s_0\rangle$,

$$\mathbb{R}(s) \geq \mathbb{B}(t) \quad \textit{and} \quad \mathbb{R}(s') = \mathbb{R}(s) - \mathbb{B}(t) + \mathbb{F}(t)$$

hold. For the sake of succinctness, we abbreviate $\mathbb{E} = \mathbb{F} - \mathbb{B}$ *(*$\mathbb{E}$ *for* effect*, which could be negative).*

We can extend the notion of effect $\mathbb{E}$ for a region to a sequence $\tau \in T^*$. The effect of the empty sequence is $\mathbb{E}(\epsilon) = 0$. The effect of a sequence $t\tau$ with $t \in T$ is defined as $\mathbb{E}(t\tau) = \mathbb{E}(t) + \mathbb{E}(\tau)$. For instance, $\mathbb{E}(abbaa) = 3{\cdot}\mathbb{E}(a) + 2{\cdot}\mathbb{E}(b)$. In general, $\mathbb{E}(\tau) = \sum_{t \in T} \#_t(\tau) \cdot \mathbb{E}(t)$, where $\#_t(\tau)$ denotes the number of times t occurs in τ.

Example 7. *For the lts* TS *in the Fig. 2.6 each region must satisfy the following system of linear equations, according to the Definition 13:*

$$\begin{cases} \mathbb{R}(s_0) & \geq & \mathbb{B}(a) \\ \mathbb{R}(s_1) & = & \mathbb{R}(s_0) - \mathbb{B}(a) + \mathbb{F}(a) \\ \mathbb{R}(s_1) & \geq & \mathbb{B}(a) \\ \mathbb{R}(s_1) & \geq & \mathbb{B}(b) \\ \mathbb{R}(s_2) & = & \mathbb{R}(s_1) - \mathbb{B}(a) + \mathbb{F}(a) \\ \mathbb{R}(s_2) & \geq & \mathbb{B}(b) \\ \mathbb{R}(s_3) & = & \mathbb{R}(s_1) - \mathbb{B}(b) + \mathbb{F}(b) \\ \mathbb{R}(s_3) & \geq & \mathbb{B}(a) \\ \mathbb{R}(s_0) & = & \mathbb{R}(s_0) - 2 \cdot \mathbb{B}(a) + 2 \cdot \mathbb{F}(a) - \mathbb{B}(b) + \mathbb{F}(b) \end{cases} \tag{2.1}$$

The triple of functions $\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ *with*

$$\begin{array}{lll} \mathbb{R}: & s_0 \to 2, \quad s_2 \to 0 & \mathbb{B}: \quad a \to 1 \qquad \mathbb{F}: \quad a \to 0 \\ & s_1 \to 1, \quad s_3 \to 3 & \qquad\quad b \to 0 \qquad\quad b \to 2 \end{array}$$

is a valid region of TS*, since it satisfies the system (2.1). For instance, for the label* a *and the arrow* (s_0, a, s_1)

$$\mathbb{R}(s_0) = 2 \geq 1 = \mathbb{B}(a) \qquad \textit{and} \qquad \mathbb{R}(s_1) = \mathbb{R}(s_0) - \mathbb{B}(a) + \mathbb{F}(a) = 1$$

which correspond the first and the second line of (2.1), respectively. The other conditions can be checked similarly.

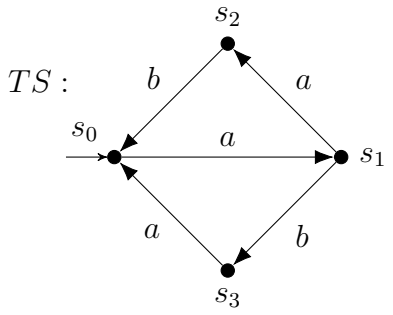


Figure 2.6: Transition system TS.

The theory of regions establishes an indirect characterisation of synthesisability of a transition system to a Petri net. This characterisation relies on the solvability of linear inequalities systems, which corresponds to fulfilling of two properties called *separation properties*.

The first separation property (*states separation property*) requires each pair of two distinct states to be distinguishable by some valid region of the lts. The second one (*event-state separation property*) for every pair of a state and a label of the lts such that the label is not activated at the state, needs a region which restricts the label at the state to be present. More formally, in terms of the theory of regions, a labelled transition system $(S, T, \to, s_0)$ satisfies

- the *state separation property* iff for all $s, s' \in [s_0\rangle$ such that $s \neq s'$ there exists a region $\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ with $\mathbb{R}(s) \neq \mathbb{R}(s')$,

- the *event-state separation property* iff for all $s \in [s_0\rangle$ and for all $t \in T$, if $\neg s[t\rangle$ then there exists a region $\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ such that $\mathbb{R}(s) < \mathbb{B}(t)$.

In the synthesis of Petri nets, the regions of a labelled transition system are transformed into places of the sought Petri net. For a given region $\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ of some lts $(S, T, \rightarrow, s_0)$ a place p can be constructed by defining the values of the flow function

$$F(p,t) = \mathbb{B}(t) \qquad \text{and} \qquad F(t,p) = \mathbb{F}(t)$$

and the initial marking

$$M_0(p) = \mathbb{R}(s_0).$$

Then, all the markings of p which are reachable from M_0 through the executions of the sequences enabled in the lts are given by

$$M_s(p) = \mathbb{R}(s) \qquad \text{for} \qquad s \in S.$$

The two separation properties become two *separation problems*, which are to be solved in order to perform the synthesis procedure. The first of them, *states separation problem* (SSP for short), requires that for every two distinct states of a transition system, a place having unequal markings at these states is present. The second one, *event-state separation problem* (ESSP for short), for every pair of a state and a transition which is not enabled at this state, needs a place which marking at this state forbids firing of this transition. If the latter problem is solvable for a given pair of a state and a transition, we shall say that this transition is *separable* at this state.

Solvability of these two separation problems SSP and $ESSP$ is necessary and sufficient for the lts being generated by a Petri net in the following sense (see e. g. [BD14a] or [BD98] for the details)

Theorem 1. [BD14a] *A (finite, totally reachable, deterministic) lts is the reachability graph of a Petri net if and only if it satisfies states separation property and event-state separation property.*

Example 8. *For the transition system TS from Fig. 2.6 the following instances of separation problems can be listed*

SSP :	$ESSP$:
all pairs of distinct states:	*b is not activated at s_0:* $\neg s_0[b\rangle$,
$s_0 \neq s_1,\ s_0 \neq s_2,\ s_0 \neq s_3,$	*a is not activated at s_2:* $\neg s_2[a\rangle$,
$s_1 \neq s_2,\ s_1 \neq s_3,\ s_2 \neq s_3$	*b is not activated at s_3:* $\neg s_3[b\rangle$

Each of these problems can be translated into the system of linear inequalities, and the solvability of the system means the solvability of the original separation problem. For example, in order to construct a region which handles the instance $\neg s_2[a\rangle$ of $ESSP$ one has to solve the following system of inequalities:

$$\begin{cases} \mathbb{R}(s_0) \geq \mathbb{B}(a) & (1) \\ \mathbb{R}(s_1) = \mathbb{R}(s_0) - \mathbb{B}(a) + \mathbb{F}(a) & (2) \\ \mathbb{R}(s_1) \geq \mathbb{B}(a) & (3) \\ \mathbb{R}(s_1) \geq \mathbb{B}(b) & (4) \\ \mathbb{R}(s_2) = \mathbb{R}(s_1) - \mathbb{B}(a) + \mathbb{F}(a) & (5) \\ \mathbb{R}(s_2) \geq \mathbb{B}(b) & (6) \\ \mathbb{R}(s_3) = \mathbb{R}(s_1) - \mathbb{B}(b) + \mathbb{F}(b) & (7) \\ \mathbb{R}(s_3) \geq \mathbb{B}(a) & (8) \\ \mathbb{R}(s_0) = \mathbb{R}(s_0) - 2 \cdot \mathbb{B}(a) + 2 \cdot \mathbb{F}(a) - \mathbb{B}(b) + \mathbb{F}(b) & (9) \\ \mathbb{R}(s_2) < \mathbb{B}(a) & (10) \end{cases} \tag{2.2}$$

The lines (1)-(9) of (2.2) repeat the system (2.1) which guarantees that each solution of the system is a valid region of TS. The newly added inequality

$$\mathbb{R}(s_2) < \mathbb{B}(a)$$

ensures that if a solution (in the form of a region of TS) of the extended system (2.2) is found, the place obtained from this region disables transition a at the marking corresponding to state s_2.

Figure 2.7: The place p obtained from the region ρ. The place q obtained from the region θ.

The region $\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ *with*

$$\begin{array}{lll} \mathbb{R}: & s_0 \to 2, \quad s_2 \to 0 & \mathbb{B}: \quad a \to 1 \qquad \mathbb{F}: \quad a \to 0 \\ & s_1 \to 1, \quad s_3 \to 3 & \qquad\quad b \to 0 \qquad\quad b \to 2 \end{array}$$

defined in Example 7 is a solution of (2.2). Indeed, from Example 7 we know that is satisfies equations (1)-(9) of (2.2). Due to

$$\mathbb{R}(s_2) = 0 < 1 = \mathbb{B}(a),$$

equation (10) of (2.2) is also satisfied. Hence, region ρ separates a *at* s_2 *or in other words* a *is separable at* s_2*.*

For the region ρ *we can construct the corresponding place* p *of a net* N *with transitions* a *and* b *as follows: The function* $\mathbb{B}$ *defines how many tokens the transition consumes from the place* p *by each its firing, and the function* $\mathbb{F}$ *defines how many tokens it produces by each firing. The marking of the place* p *is determined by the function* $\mathbb{R}$*. Since we are looking for the Petri net with the reachability graph isomorphic to* TS*,* $\mathbb{R}$ *exactly defines the marking of* p *for each node of the reachability graph (i.e. for each marking reachable in* N*). Finally, we obtain a place* p *(see in the left of Fig. 2.7) with the initial marking and the arc weights as follows:*

$$M_0(p) = 2$$
$$F(p,a) = 1,\ F(p,b) = 0,\ F(a,p) = 0,\ F(b,p) = 2.$$

For the event/state separation problem $\neg s_0[b\rangle$ *one can construct the following*

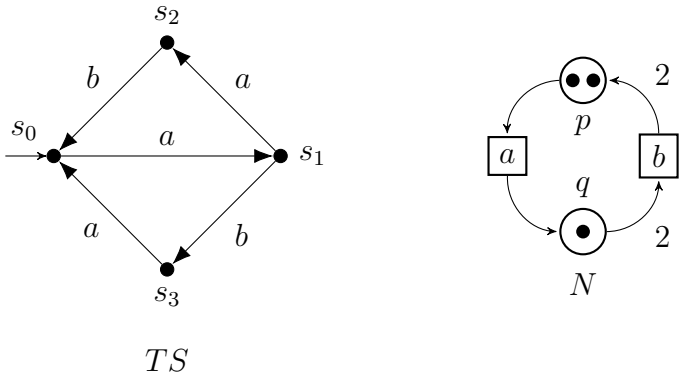


Figure 2.8: Transition system TS. Petri net N with $RG(N)$ isomorphic to TS.

region $\theta = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ *with*

$$\begin{array}{lll} \mathbb{R}: \quad s_0 \to 1, \quad s_2 \to 3 & \mathbb{B}: \quad a \to 0 & \mathbb{F}: \quad a \to 1 \\ \qquad s_1 \to 2, \quad s_3 \to 0 & \qquad b \to 2 & \qquad b \to 0 \end{array} \tag{2.3}$$

Since the system (2.1) is satisfied for θ*, it is a valid region of* TS *as well. Due to*

$$\mathbb{R}(s_0) = 1 < 2 = \mathbb{B}(b),$$

the instance $\neg s_0[b\rangle$ *of ESSP is solved by* θ*. The region* θ *is translated into the place* q *(see in the right of Fig. 2.7) with*

$$M_0(q) = 1$$
$$F(q,a) = 0,\ F(q,b) = 2,\ F(a,q) = 1,\ F(b,q) = 0.$$

Let us also note that the region θ *also solves the instance* $\neg s_3[b\rangle$ *of the event-state separation problem. This means that there is no need to construct any additional region (and a place, respectively) for this instance of ESSP. Moreover, since for* θ *(as well as for* ρ*) the values of the function* $\mathbb{R}$ *are different for all the states of* TS*, all the SSP issues are also solved by any of these regions. Hence, the net* N *in Fig. 2.8 with the places* p *and* q *solves* TS*.*

2.3 Motivating remarks

The following decision problem can be considered as an instance of a Petri net synthesis task:

Input: A finite, totally reachable, label-deterministic labelled transition system TS.
Question: Is there some bounded Petri net N which solves TS?

Finiteness of the input lts corresponds to the boundedness of the sought Petri net. Total reachability of the transition system is necessary due to the fact that the reachability graph of a Petri net is always a totally reachable transition system. As for any Petri net a firing of a transition uniquely determines the marking reachable through this firing, the input lts is required to be label-deterministic.

According to the theory of regions, in order to decide this problem we have to check whether TS satisfies both separation properties, that have been described in the previous section. Checking satisfiability of separation properties relies on constructing the regions of an lts. But, even for a finite lts the set of its regions is infinite. One can try to find a finite region basis [BDLM07], but constructing this basis is not an easy task. Moreover, even having this basis, in order to check whether states separation property is satisfied, it is required to examine all pairs of distinct states of the lts, and in order to check satisfiability of event-state separation property – all pairs of label/state. This procedure may happen to be not efficient, although there are tools such as `APT` [S$^+$13] or `synet` [Cai02] which are capable to produce a very good result of the Peri net synthesis. Furthermore, for some special classes of Petri nets more precise statements can be made [BD14b].

What is more important in our consideration, starting from the shape of an arbitrary labelled transition system, it looks rather difficult to answer if there is a Petri net solving it, and even harder to predict how such Petri net (if there exists one) can look like. While synthesising a Petri net from an lts, only the set of transitions is fixed, being the set of labels of the lts. The set of places can differ significantly, as in the number of tokens in the initial marking, so also in their connections to the transitions. Even in case of a successful synthesis procedure, the output is not guaranteed to be unique, and sometimes it is not

even irredundant. For instance, the output of synthesis with the tool `synet` (as well as with the tool `APT`) differs, depending on the particular way an lts was presented to the tool (state numbering or ordering of transitions). Nevertheless, the obtained Petri nets have isomorphic reachability graphs, which is guaranteed by the correctness-by-design of the output of synthesis procedure. Also, some places that were created at the earlier stages of the synthesis procedure may become redundant after adding some newly created places at later stages. Although the tool `APT` exploits some heuristics in order to exclude redundant places (if there are any) and to minimise the total number of produced places, the output Petri net can still differ for isomorphic labelled transition systems as inputs. All this gives us an intuition that the structure of a reachability graph has no strong relation to the structure of some Petri net it can originate from.

Example 9. *For the labelled transition system TS in Fig. 2.9 the result of the synthesis of a Petri net with a reachability graph isomorphic to TS varies depending on the tool which is used. For instance,* `synet` *produced the Petri net N_1 in the right of Fig. 2.9, which happens to be pure (without side-conditions) and has 4 places. The same transition system being provided as an input to* `APT`*, results the Petri net N_2 (see Fig. 2.10) which has 7 places. On the other hand,* `APT` *allows to initiate an optimisation of the synthesis with the key* `minimise`*. The result of such synthesis from the same input transition system is N_3 in Fig. 2.11, having only 2 places.*

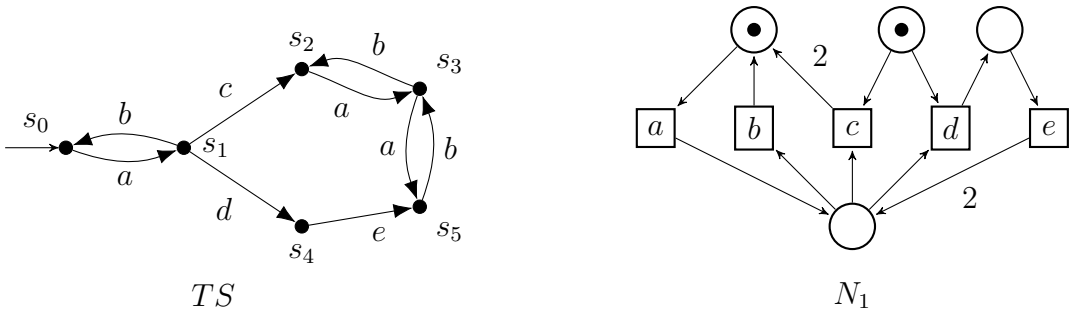


Figure 2.9: TS is isomorphic to the reachability graph of the Petri net N_1 which is synthesised by `synet`.

Albeit, there are results establishing structural properties of reachability graphs,

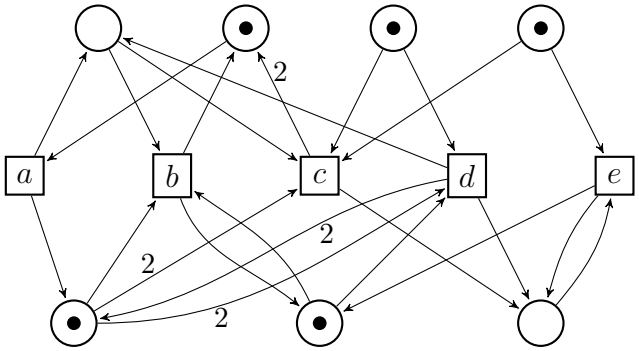


Figure 2.10: N_2 is synthesised by APT from TS (see Fig. 2.9).

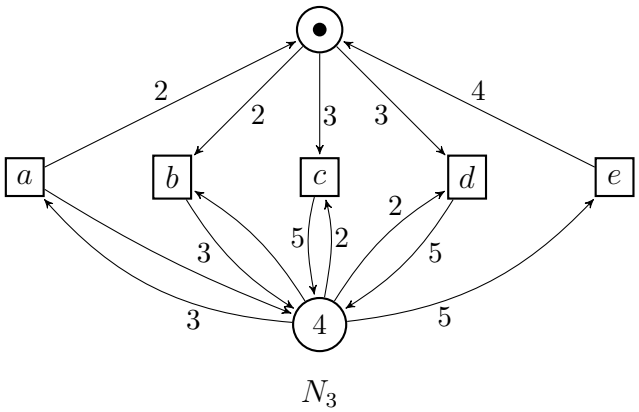


Figure 2.11: N_3 is produced from TS (see Fig. 2.9) by APT with minimisation.

obtained from Petri nets satisfying certain restrictions [LR78, KCK+95], only some of them (for instance [BD14b]) provide a characterisation of state spaces, which can be implemented with Petri nets (of a particular class). In this work, we are focused on the problem of characterising syntehsisable state spaces, and we investigate how such a characterisation can contribute to the field of Petri net synthesis.

2.4 Separation problems for linear lts

In the present section we restrict our attention to one of the simplest shapes of labelled transition systems – finite lines, which can be associated with finite sequences of symbols (*words*) over some finite alphabet in a natural way.

Definition 14. *A* word over alphabet T *is a finite sequence* $w \in T^*$. *A word is called* binary *if* $|T| = 2$. *The* empty word*, i.e., the word that has no letters in it, is denoted by* ϵ. *A word* $w' \in T^*$ *is called a* subword *(or* factor*) of* $w \in T^*$ *if* $w = u_1 w' u_2$ *for some* $u_1, u_2 \in T^*$ *(i.e. factor of* w *is a contiguous segment of* w*).*

The following definition determines a mapping which uniquely assigns a finite transition system to each finite sequence over a given alphabet T.

Definition 15. *A sequence* $w = t_1 t_2 \dots t_n \in T^*$ *of length* $n \in \mathbb{N}$ *uniquely corresponds to a finite transition system* $TS(w) = (S, T, \to, s_0)$*, where*

- $S = \{s_0, \dots, s_n\}$ *– the set of states of the transition system,*
- T *– the set of labels which coincides with the alphabet,*
- $\to = \{(i-1, t_i, i) \mid 0 < i \leq n \wedge t_i \in T\}$ *– the set of arrows,*
- $s_0 \in S$ *– the initial state.*

This correspondence allows us to extend the notion of solvability of labelled transition systems with Petri nets to finite sequences. We will say that Petri net N with the set of transitions T solves a word $w \in T^*$ if N solves $TS(w)$.

Definition 16. *The word* w *is called* solvable *(or Petri net solvable, or PN-solvable) if* $TS(w)$ *is solvable by some Petri net* N*, otherwise* unsolvable*.*

Example 10. *The sequence $w = abcb$ over $T = \{a, b, c\}$ has, for instance, abc or bc as its contiguous fragments, hence subwords, while acb is not a subword of w. The transition system $TS = TS(w)$ is depicted on the left of Fig. 2.12. This transition system is isomorphic to the reachability graph of the Petri net N in the right of Fig. 2.12. Thus, TS is solvable, implying that the sequence w is solvable as well.*

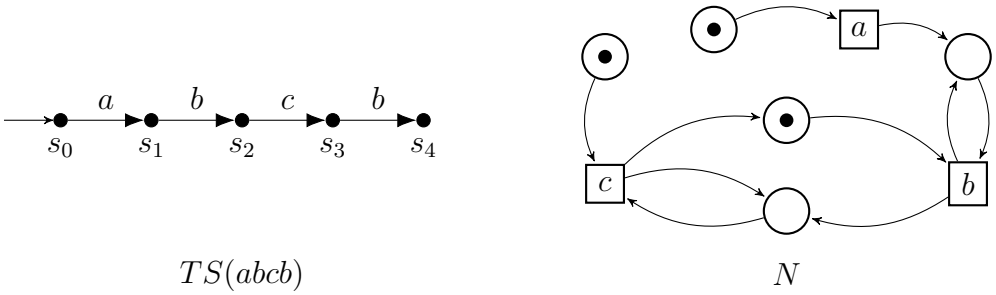


Figure 2.12: Transition system $TS = TS(abcb)$, and net N with $RG(N) \cong TS$.

The notion of a *Parikh vector* of a sequence of symbols is widely used in the literature, introducing some sort of measuring for sequences and allowing to reason about their 'similarities' or internal structure.

Definition 17. *For a finite sequence $\sigma \in T^*$ of labels, the* Parikh vector $\Psi(\sigma)$ *is a T-vector (i.e., a vector of integer numbers with index set T), where $\Psi(\sigma)(t)$ denotes the number of occurrences of t in σ, i.e. $\Psi(\sigma)(t) = \#_t(\sigma)$. For states s, p, in sequence $|_{s_0}u|_s v|_p w \in T^*$, the notation $\Psi(p - s) = \Psi(uv) - \Psi(u)$ is used for the Parikh vector of the path v from s to p. We use $\Psi(s) = \Psi(s - s_0) = \Psi(u)$ to denote the Parikh vector of the path u from the initial state s_0 to s.*

We have earlier considered the two kinds of separation problems, SSP and ESSP, which according to the theory of regions [BD98] represent the necessary and sufficient condition for synthesis of a Petri net from a given transition system. The first of them, states separation problem, requires for every pair of unequal states of the initial lts the existence of a region which distinguishes them from each other. The other one, event-state separation problem, needs for every pair of a transition and a state at which this transition is not enabled a region preventing

the transition from occurrence at the state. We shall now examine these conditions for the special class of labelled transition systems – the ones that are derived from finite sequences using the mapping TS (see Definition 15).

Let T be a finite nonempty alphabet, and $w \in T^*$ be a finite sequence of length $|w| = n$ over T. We can construct a finite labelled transition system $TS(w) = (S, T, \rightarrow, s_0)$ from w. In order to synthesise – if possible – a Petri net with the reachability graph isomorphic to $TS(w)$, T must, of course (since we do not consider any transition labels), be used directly as the set of transitions.

There are $|S| = n + 1$ states in $TS(w)$, and to satisfy the state separation property one has to solve $\frac{1}{2}\cdot(|S|\cdot(|S|-1)) = \frac{1}{2}\cdot(n+1)\cdot n$ state separation problems – one problem for each pair of distinct states. It turns out that all such problems are solvable if the transition system originates from a sequence (word). Moreover, it is always possible to construct a region which solves all state separation problems of such an lts. For instance, we can define the region $\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ as follows:

$$\mathbb{R}: \ \ s_i \rightarrow i, \ \ \forall 0 \leq i \leq n, \qquad \mathbb{B}: \ \ t \rightarrow 0, \qquad \mathbb{F}: \ \ t \rightarrow 1, \ \ \forall t \in T \tag{2.4}$$

which serves as a solution for all instances of SSP of $TS(w)$. Indeed, $\mathbb{R}$ has unique value i for each state $s_i \in S$ (in fact, this region just counts the number of letters that have been read up to the state), hence $TS(w)$ satisfies states separation property.

Example 11. *For the $TS(abcb)$ (see Fig. 2.12) the region ρ as in (2.4) is defined as follows:*

$$\begin{array}{lll} \mathbb{R}: \ \ s_0 \rightarrow 0, \ \ s_1 \rightarrow 1, & \mathbb{B}: \ \ a \rightarrow 0, \ \ b \rightarrow 0, & \mathbb{F}: \ \ a \rightarrow 1, \ \ b \rightarrow 1, \\ \quad s_2 \rightarrow 2, \ \ s_3 \rightarrow 3, & \quad c \rightarrow 0 & \quad c \rightarrow 1 \\ \quad s_4 \rightarrow 4 & & \end{array}$$

Since the values are distinct for distinct states:

$$\mathbb{R}(s_i) \neq \mathbb{R}(s_j), \quad \textit{for } \ i \neq j, \ 0 \leq i, j \leq 4,$$

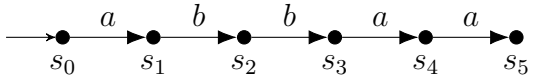
this region ρ solves all the instances of SSP for transition system $TS(abcb)$.

The situation is different when talking about event-state separation property.

In order to satisfy this property, for the transition system $TS(w)$, $|S| \cdot |T| - |S| + 1$ event-state separation problems need to be solved (we have $|T| - 1$ instances of ESSP for each of the $|S| - 1$ first states in $TS(w)$, and $|T|$ instances of ESSP for the last state). Unlike SSP, this kind of problems in general does not have the same solution for transition systems from the class under consideration. Furthermore, some ESSP instances may not be solvable at all for transition systems derived from words (see e.g. [BBE+16] for details). For example, when considering the word

$$ab|_{s_2}baa,$$

and the labelled transition system $TS(abbaa)$ (see Fig. 2.13) obtained from it, the instance $\neg s_2[a\rangle$ of ESSP has to be solved. In other words, we have to construct a region $\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ which restricts a at state s_2. For ρ, we have to guarantee $\mathbb{R}(s_2) < \mathbb{B}(a)$.



$TS(abbaa)$

Figure 2.13: Transition system derived from *abbaa*.

Assume that there exists ρ which solves the instance $\neg s_2[a\rangle$ of ESSP. Then it satisfies the following inequalities:

(0) $\mathbb{B}(a) \leq \mathbb{R}(s_0)$, since state s_0 activates a;

(4) $\mathbb{B}(a) \leq \mathbb{R}(s_0) + \mathbb{E}(abba)$, since state s_4 activates a;

(2) $\mathbb{R}(s_0) + \mathbb{E}(ab) < \mathbb{B}(a)$, due to the assumption about ρ.

This set of inequalities cannot be solved in the natural numbers. Combine (0) and (2) to obtain $0 < -\mathbb{E}(ab)$; combine (4) and (2) to obtain $0 < \mathbb{E}(abba) - \mathbb{E}(ab) = \mathbb{E}(ab)$; contradiction. This demonstrates that even simple sequences can be unsolvable with Petri nets. Although, the theory of regions suggests us a way to check a sequence (or lts) for its solvability, we see that it can be quite time

consuming, which suggest us a reasonable question: how can we figure out if the given labelled transition system is (un-)solvable in some uncomplicated way? To be more precise, we would like to give a characterisation for some class(es) of (un)solvable labelled transition systems. We will gradually achieve this aim throughout the following sections and chapters. More particularly, we will present a language-theoretical criterion for solvability of binary sequences with Petri nets, together with a synthesis algorithm based on this criterion. With the use of the criterion, a classification of unsolvable sequences will be described, which gives rise to an efficient procedure for checking whether a given binary sequence is not solvable. Moreover, in the last chapter we extend our view to finite sets of sequences (languages), and present a characterisation of the solvability of such sets with Petri nets.

2.5 Structural properties of (un)solvable words

In this section we shall consider some necessary and sufficient properties of solvable words step by step. Among other things, we will investigate how the shape of a word can affect its solvability. E.g., the unsolvability of a factor of a sequence, implies the unsolvability of the whole sequence, and the same for a relatively big difference between lengths of blocks of the same letter inside the sequence. We will also show that some kinds of consecutive modifications can or cannot preserve solvability of separation problems. For instance, prepending of the initial letter to a solvable sequence retains the solvability. Moreover, a presence of a factor of some pseudo-regular form will be proved as a sufficient condition for the unsolvability. A notion of a minimal unsolvable word will be introduced as a convenient tool for focusing on the crucial properties implying the (un)solvability.

We think that the first natural step in the investigation would be to 'localise' the part which is problematic for the solvability. It turns out that if a sequence w is solvable, then of all its subwords w' are. To see this, let the Petri net solving w be executed up to the state before w', take this as the new initial marking, and add a pre-place with $\#_a(w')$ tokens to a and a pre-place with $\#_b(w')$ tokens to b. Thus, the unsolvability of any proper subword of w entails the unsolvability of w.

Proposition 1. (solvability of subwords) *If $w = xvy$ with $x, y \in \{a, b\}$ is a*

solvable finite sequence, then xv and vy are also solvable.

Thanks to the Proposition 1, it is convenient to introduce the notion of a *minimal unsolvable word*, namely,

Definition 18. *A word (finite sequence) w is called a* minimal unsolvable word *(muw) if it is an unsolvable word, and all of its proper subwords are solvable.*

A complete list of minimal unsolvable words over $\{a, b\}$ up to length 110 can be found, amongst some other lists, in [P$^+$15] (see also Appendix for the list of minimal unsolvable binary words up to the length 20). Since we have such a list, we begin with an attentive watch of it in order to parse some possible regularities of these words. Observe, for instance, that in this list, every word starts and ends with the same letter. This is a consequence of the next proposition.

Proposition 2. (solvability of aw and wb implies solvability of awb) *If both aw and wb are solvable, then awb is also solvable.*

Proof. Assume that aw and wb are PN-solvable words over $\{a, b\}$. If $w = b^k$ (or $w = a^k$) for $k \in \mathbb{N}$ then $awb = ab^{k+1}$ ($awb = a^{k+1}b$, respectively) is obviously solvable (see, for instance, Figure 2.14). Hence we assume that w contains at least

Figure 2.14: Petri nets solving sequences ab^{k+1} (l.h.s) and $a^{k+1}b$ (r.h.s.) for $k \geq 0$.

one a and one b. Let $N_1 = (P_1, \{a, b\}, F_1, M_{01})$ and $N_2 = (P_2, \{a, b\}, F_2, M_{02})$ be Petri nets such that N_1 solves aw and N_2 solves wb. We can assume that N_1 and N_2 are disjoint, except for their transitions a and b. Forming the union of N_1 and N_2 by synchronisation at a and b gives a net which allows for execution all (and only) sequences allowed by both N_1 and N_2. Before forming such a union, we modify N_1 and N_2 as follows:

(i) Since N_1 solves the sequence aw, we have to modify it in such a way that it allows one last b after executing aw and stops computations having awb executed. In order to reach this goal, in N_1, for each place p in ${}^\bullet b \cap P_1$, we have to add another $F_1(p, b)$ tokens in the initial marking $M_{01}(p)$. This

additional tokens enable transition b after the sequence aw. It may be the case that before the modification such place p has been used to restrict firings of transition a, and the added surplus tokens break this function of p. To prevent this, if p in ${}^\bullet a \cap P_1$, then increase both weights $F_1(p,a)$ and $F_1(a,p)$ by the quantity $F_1(p,b)$; otherwise, keep the arc weights unchanged. These new arc weights keep additional tokens on p untouched while executing aw, and does not influence the firing of b afterwards. Since the last b in awb could have enabled a at the final state, we construct a counting place q_a which is an input place for transition a with a unit arc weight $F(q_a,a)=1$, and which has $\#_a(aw)$ tokens on it initially. Thus, a remains disabled while executing awb exactly at states in which it was disabled before the modification and becomes permanently disabled after aw.

(ii) As N_2 solves the word wb, we need to transfigure it to allow awb for execution. To achieve this, for each place q in ${}^\bullet a \cap P_2$ we supplement another $F_2(q,a)$ tokens to its initial marking $M_{02}(p)$. These tokens are enough to enable the additional a initially. Further, for each place p in $a^\bullet \cap P_2 \cap {}^\bullet b$, increase the both arc weights $F_2(p,b)$ and $F_2(b,p)$ by the quantity $F_2(a,p)$. The new arc weights lead to the same effect of b on p but prevent premature occurrences of b in the part wb (which could have been allowed by adding the tokens in front of b after the firing of the initial a in awb). Moreover, if there is a place p in ${}^\bullet a \cap {}^\bullet b \cap P_2$, b could have been allowed at the very beginning of awb by changing the initial marking $M_{02}(p)$. To prevent this, construct a new place p' in N_2, such that $F_2(a,p') = F_2(b,p') = F_2(p',b) = 1$ and $F_2(p',a) = M_0(p') = 0$. This place prevents firing of transition b at the beginning of awb, and it does not influence the behaviour of N_2 after the first a has occurred.

Define N as the union of the two nets thus modified (see Example 12 for details). In general, N solves awb in the following way: the initial a is allowed in N_1 by definition and in N_2 by the additional tokens. The subsequent w is allowed in both nets, and hence in their synchronisation. The final b is allowed in N_2 by definition and in N_1 by the additional tokens. No premature b is allowed by the arc weight increase, and no additional a is allowed after executing of awb because of a counting place q_a constructed in N_1. All intermediate occurrences of a are regulated by the

modification of N_1, and the same of b – by the modification of N_2. □

The following example illustrates the construction suggested in the proof of Proposition 2.

Example 12. *For demonstrating the transformations described in Proposition 2, consider the case $w = bab$. Both sequences $aw = abab$ and $wb = babb$ are solvable, (for instance, by Petri nets N_1 and N_2 from Fig. 2.15 and 2.16, respectively). First, we modify the net N_1 according to the steps described in item (i) of the proof. The result of this modification is the net N_1' in Fig. 2.15. The tokens added during the transformations are drawn as hollow circles and newly constructed places and adjacent arcs are dashed. Let us notice, although N_1' can execute the sequence $awb = ababb$, it also allows some more behaviour (see its reachability graph $RG(N_1')$ in Fig. 2.15). For this reason it is not enough to use only one transformed net. The modification N_2' (see Fig. 2.16) of the net N_2 is obtained by the item (ii) of the proposition proof. After the synchronisation of the two nets N_1' and N_2' by transitions, we derive the net N (see Fig. 2.16), whose reachability graph is isomorphic to $TS(awb)$. Hence, N solves the sequence $awb = ababb$. Let us also notice that the resulting net N is not necessarily an optimal solution (in the sense of the size of the net). It is easy to see, that some places of the net N are redundant: for instance, places q_1, q_a and q_2 have the same function – restricting the total number of firings of transition a. Similarly, places r_1 and r_2 also duplicate each other.*

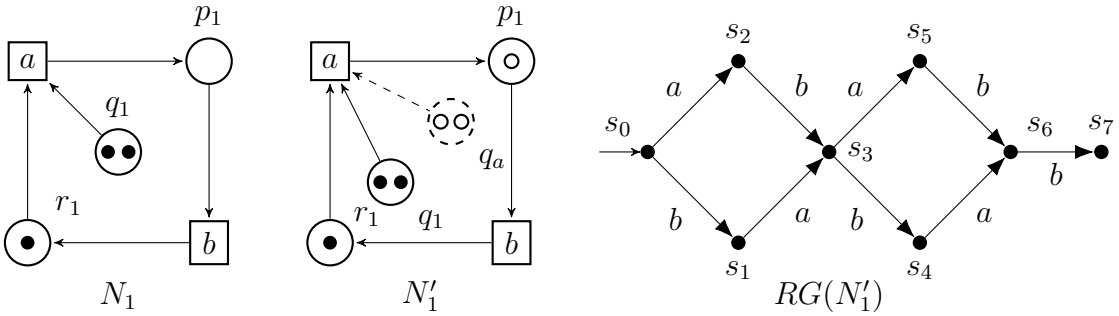


Figure 2.15: N_1 (l.h.s.) solves *abab*; N_1' (derived from from N_1) allows *ababb*.

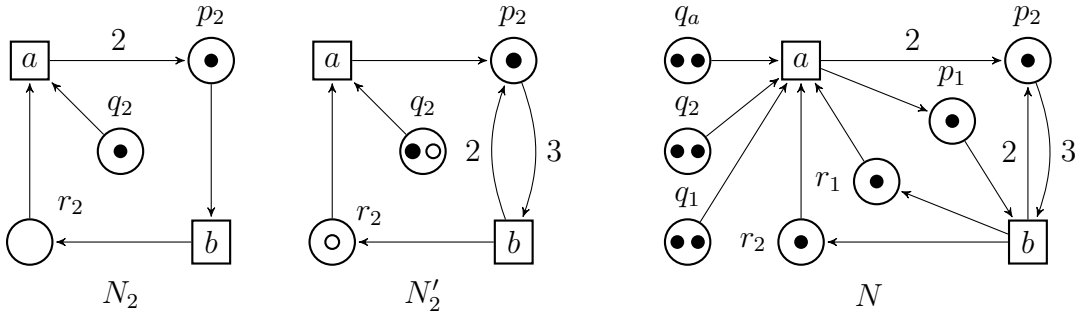


Figure 2.16: N_2 solves *babb*; N_2' (obtained from N_2) allows *ababb* for firing; net N solves *ababb*, N is derived from N_1' (see Fig. 2.15) and N_2' by synchronising at transitions.

The next observation from the list [P$^+$15] is that for any minimal unsolvable word, if it starts with a (or with b) then the next letter is always b (a, respectively). This observation is confirmed by the following proposition, establishing that prepending of the first letter to a solvable word does not violate its solvability.

Proposition 3. [BBE$^+$16](solvable words starting with a can be prefixed by a) *If a word av is PN-solvable then aav is, too.*

Proof. Let $N = (P, \{a, b\}, F, M_0)$ be a net solving av. We shall construct a net which solves aav. The idea is to obtain such a net by 'unfiring' a once from the initial marking of N. Since this may lead to a non-semipositive marking of some places from P, which we would like to avoid, we will first normalise and modify the net N, obtaining another solution N' of av, and then construct a solution N'' for aav as a modification of N'.

(i) For normalisation, we assume that there are two places p_b and q_a; the first prevents b explicitly in the initial phase, and the second prevents firing of a after the last of its occurrences in av. They are defined by $M_0(p_b) = 1$, $F(a, p_b) = 1$, $F(b, p_b) = \ell{+}1 = F(p_b, b)$, where $\ell \geq 1$ is the number of a before the first b in av, and $M_0(q_a) = k$, $F(q_a, a) = 1$, where $k \geq 1$ is the number of a in av (i.e. $k = \#_a(av)$). All other undefined values of F for p_b and q_a are supposed to be equal 0.

(ii) Let $NUF(a) = \{p \in a^\bullet \mid M_0(p) < F(a, p)\}$ be the set of places which do not allow the 'unfiring' of a at the initial marking M_0. Note that neither

p_b nor q_a are in $NUF(a)$. Note also that for every place $p \in NUF(a)$, $F(p,a) \leq M_0(p) < F(a,p)$ – the first inequality holds because a is initially enabled, the second – by $p \in NUF(a)$. That is, while firing a produces on such places more tokens than consumes, i.e. a has a positive effect on every $p \in NUF(a)$. Without loss of generality, b has a negative effect on every place from $NUF(a)$. Indeed, if, by contraposition, for some $p \in NUF(a)$ the effect of b on p is non-negative, then this place p does not prevent transition b from firing, once b has been enabled. On the other hand, all premature occurrences (the ones before the first b in av) of b are already restricted by the place p_b. Hence, such place p could be deleted without changing the behaviour of N.

(iii) For every place $p \in NUF(a)$ we add the quantity $F(a,p)$ uniformly to its initial marking $M_0(p)$, and to $F(p,b)$, and to $F(b,p)$, eventually obtaining the modified net $N' = (P', \{a,b\}, F', M_0')$. We shall show that N' also solves av, as N. First, both $M_0[a\rangle \wedge \neg M_0[b\rangle$ and $M_0'[a\rangle \wedge \neg M_0'[b\rangle$ (the former holds by the definition of N, the latter – by the construction). For an inductive proof, suppose that $M_0[a\rangle M_1[\tau\rangle M$ and $M_0'[a\rangle M_1'[\tau\rangle M'$. We have $M[b\rangle$ iff $M'[b\rangle$ by the construction. If $M[a\rangle$, then also $M'[a\rangle$, since $M \leq M'$. Next, suppose that $\neg M[a\rangle$; then there is some place q such that $M(q) < F(q,a)$. We show that, without loss of generality, $q \notin NUF(a)$, so that q also disables a at M' in N'. If M disables a after the last a in av, we can take $q = q_a \notin NUF(a)$. If M disables a before its last occurrence in av, then q cannot be in $NUF(a)$, since b acts negatively on such places.

Now, we construct a net $N'' = (P', \{a,b\}, F', M_0'')$ from N' by defining $M_0''(p) = M_0'(p) - F'(a,p) + F'(p,a)$ for every place $p \in P'$. By the construction, aav is a firing sequence of N''. Furthermore, M_0'' does not enable b because of p_b. □

The following example explains the construction described in Proposition 3.

Example 13. *Consider the word $av = abab$. This word is solvable, and a possible solution is the net N (see l.h.s. of Fig. 2.17). Since the net N does not possess the states q_a and p_b, they are constructed additionally (in l.h.s. of Fig. 2.17 they are drawn dashed). These new places do not influence the behaviour of the net. After this normalisation we can construct the set $NUF(a) = \{p_1\}$ which is a singleton for*

this case. Modification of the arc weights and the initial marking of p_1 yields the net N' (in the middle of Fig. 2.17). From the initial marking of net N' we are able to 'unfire' (or 'back-fire') transition a, which leads to the new initial marking in the net N'' (r.h.s. of Fig. 2.17). The reachability graph of the net N'' is isomorphic to the transition system $TS(aabab) = TS(aav)$, i.e. N'' solves $aav = aabab$.

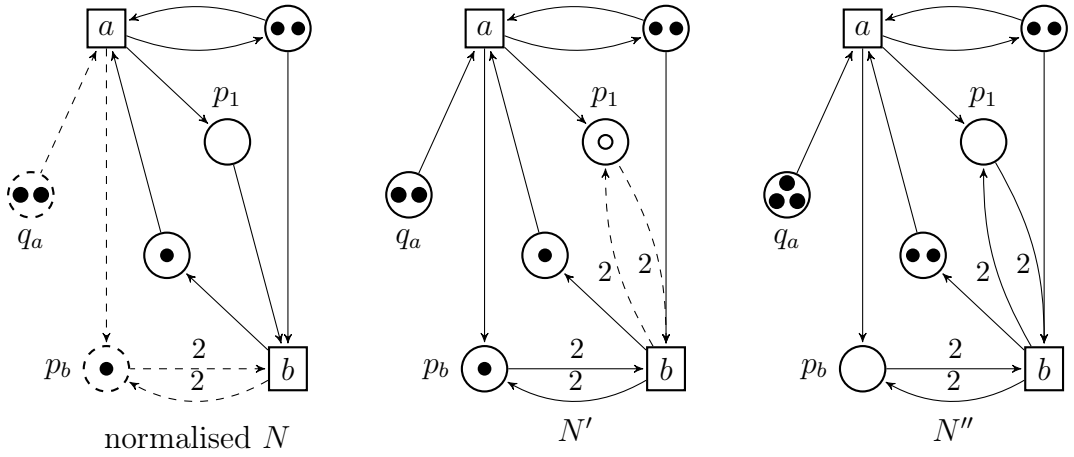


Figure 2.17: N (the solid part of in l.h.s.) solves *abab*; normalisation of N (l.h.s.) does not change the behaviour; N' obtained from normalised N; N'' (r.h.s.) solves *aabab*.

Hence, if a word av is minimal unsolvable, then, as a consequence of Proposition 3, v definitely starts with a letter b. That is, no minimal unsolvable word can start with aa (nor with bb, for that matter).

Studying the list [P+15] further, it can also be observed that all words starting and ending with b are just symmetrical images of those starting and ending with a under swapping letters. More interestingly, all minimal unsolvable words starting and ending with the letter a happen to satisfy the following general pattern:

$$\boxed{(\ a\ b\ \alpha\ )\ b^*\ (\ b\ a\ \alpha\ )^+\ a\ , \qquad \text{with } \alpha \in \{a,b\}^*} \tag{2.5}$$

with a not being separated (at least) at the state between the b^* and the second bracket (and thus, before the first b in the second bracket, which exists because the bracket contains at least one instance of $ba\alpha$). For example, *abbaa* satisfies (2.5) with $\alpha = \epsilon$, the star * being repeated zero times, and the plus $^+$ being repeated

once. Another example is *abbbaa*, where $\alpha = \epsilon$, the star * is repeated only once, and the plus $^+$ is repeated once as well. The word *ababaabaaa* can be decomposed as $(aba)(baa)(baa)a$, from where we see that it satisfies (2.5) with $\alpha = a$, the star * being repeated zero times, and the plus $^+$ being repeated twice.

The following proposition establishes that all words which contain a subword of form (2.5) are generally PN-unsolvable:

Proposition 4. [BBE$^+$15](sufficient condition for the unsolvability of a word) *If a word over $\{a, b\}$ has a subword of the form*

$$(\ a \ b \ \alpha \ ) \ b^* \ (\ b \ a \ \alpha \ )^+ \ a \ , \qquad \textit{with } \alpha \in \{a, b\}^*,$$

then it is not PN-solvable.

Proof. Let s_0 be the state before the first a, s the state before the first b in the second bracket, s' the state after this b, and r the state before the final a:

$$(\ |_{s_0} \ a \ b \ \alpha \ ) \ b^* \ (\ |_s \ b \ |_{s'} \ a \ \alpha \ )^+ \ |_r \ a$$

For a word w having a subword of this form, we prove that such a subword cannot be solved (implying that w cannot be solved either). Because $ba\alpha$ occurs at least once in the second bracket, $s \neq r$, b is enabled at state s, and a is not enabled at s. Suppose that some place q of a general form as in Fig. 2.18 with the initial marking $M_0(q) = m$ exists, which prevents a from firing at state s. Abbreviate $\mathbb{E}_q(ab\alpha)$ to E and $\mathbb{E}_q(b)$ to E_b. For place q, we have the following inequalities (for the corresponding states):

(s_0)	a_-	$\leq$	m	
(s')	a_-	$\leq$	$m + E + k{\cdot}E_b + E_b$	for some fixed $k \geq 0$
(r)	a_-	$\leq$	$m + E + k{\cdot}E_b + \ell{\cdot}E$	for the same k and some fixed $\ell > 0$
(s)	0	$\leq$	$-m - E - k{\cdot}E_b + a_- - 1$	for the same k

(s_0) is true because a is enabled at s_0. (s') is true because a is enabled at s'. (r) is true because a is enabled at r; and $\ell > 0$ because the second bracket repeats at least once. Finally, (s) is true because, by the assumption, q disables a at state s. Adding (s')+(s) gives $1 \leq E_b$. Adding (s_0)+(s) gives $1 \leq -E - k{\cdot}E_b$, and using

also $1 \leq E_b$ and $k \geq 0$ gives $1 \leq -E - k{\cdot}E_b \leq -E$. Adding (r)+(s) gives $1 \leq \ell{\cdot}E$, contradicting $1 \leq -E$ because of $\ell > 0$. The system cannot be solved, and no place q preventing a from firing at state s exists. □

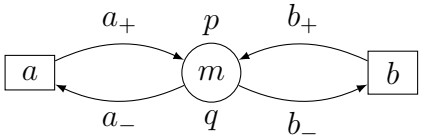


Figure 2.18: A general form of a place of Petri net with transitions a and b. The place has its initial marking $m \geq 0$, and four non-negative arc weights a_-, a_+, b_-, b_+. It is named p if used for preventing b and named q if used for preventing a.

Proposition 4 gives us a first intuition that, besides known region-theoretical approach, solvability of special classes of transition systems, like sequences, with Petri nets can also be characterised more directly, i.e. language-theoretically. The following proposition continues this idea, and establishes a next sufficient condition for the unsolvability of finite sequences. The condition relies on the fractions of letters within factors of a sequence.

Proposition 5. [BBE$^+$16](another sufficient condition for unsolvability) *Suppose $\alpha, \beta \in \{a, b\}^+$ and $w = \alpha\beta a$, where α starts with a, β starts with b, and*

$$\boxed{\#_a(\beta){\cdot}\#_b(\alpha) \;\geq\; \#_a(\alpha){\cdot}\#_b(\beta)} \tag{2.6}$$

Then w is unsolvable.

Proof. Let s_0 be the state before α, s the state before β, and r the state before the final a:

$$w \;=\; |_{s_0}\, \alpha\, |_s\, \beta\, |_r\, a.$$

Assume that there is a place q (see Fig. 2.18) which prevents firing of a at state s and has initial marking m at s_0. Let $E_a = \mathbb{E}_q(a)$, $E_b = \mathbb{E}_q(b)$, $E_\alpha = \mathbb{E}_q(\alpha)$ and $E_\beta = \mathbb{E}_q(\beta)$. Then for $E_\alpha = \#_a(\alpha){\cdot}E_a + \#_b(\alpha){\cdot}E_b$ and $E_\beta = \#_a(\beta){\cdot}E_a + \#_b(\beta){\cdot}E_b$

we have:

$$\begin{array}{lllll} (s_0) & a_- & \leq & m & \text{(since } \alpha \text{ starts with } a) \\ (r) & a_- & \leq & m + E_\alpha + E_\beta & \text{(since } r \text{ enables } a) \\ (s) & 0 & \leq & -m - E_\alpha + a_- - 1 & \text{(since } \neg s[a\rangle) \end{array}$$

Adding (s_0)+(s) yields $1 \leq -E_\alpha$, hence (A): $-(\#_a(\alpha)\cdot E_a + \#_b(\alpha)\cdot E_b) \geq 1$.
Adding (r)+(s) yields $1 \leq E_\beta$, hence (B): $(\#_a(\beta)\cdot E_a + \#_b(\beta)\cdot E_b) \geq 1$.
Also, $E_b \geq 1$ because q prevents a at s, but a becomes enabled after one or more firings of b. Then,

$$\begin{array}{lll} -\#_a(\beta) & \geq & \#_a(\beta)\cdot\#_a(\alpha)\cdot E_a + \#_a(\beta)\cdot\#_b(\alpha)\cdot E_b \quad \text{(multiplying (A) by } \#_a(\beta)) \\ & \geq & \#_a(\beta)\cdot\#_a(\alpha)\cdot E_a + \#_a(\alpha)\cdot\#_b(\beta)\cdot E_b \quad \text{(using (2.6) and } E_b \geq 1) \\ & \geq & \#_a(\alpha) \quad \text{(multiplying (B) by } \#_a(\alpha)) \end{array}$$

However, $-\#_a(\beta) \geq \#_a(\alpha)$ implies $\#_a(\beta) = \#_a(\alpha) = 0$, and this is a contradiction since α contains at least one a. Thus, such a place q does not exist. □

For sequence $abbaa$, which is known to be unsolvable, equation (2.6) can be checked with the following decomposition:

$$ab \mid_s baa \;=\; \alpha \mid_s \beta a, \quad \text{where } \alpha = ab,\ \beta = ba.$$

Then we have

$$\#_a(\beta)\cdot\#_b(\alpha) = 1 \;\geq\; 1 = \#_a(\alpha)\cdot\#_b(\beta),$$

i.e. (2.6) is satisfied (being an equality in this case). If we check (2.6) for another unsolvable sequence $abbbaa$, then for the decomposition

$$abb \mid_s baa \;=\; \alpha \mid_s \beta a, \quad \text{where } \alpha = abb,\ \beta = ba$$

the equation reads as

$$\#_a(\beta)\cdot\#_b(\alpha) = 2 \;\geq\; 1 = \#_a(\alpha)\cdot\#_b(\beta)$$

which holds true. On the other hand, for a solvable sequence *ababba* (Petri net N in Fig. 2.19 is its possible solution) there are three possible decompositions such that α starts with a and β starts with b:

$$\begin{aligned} a \mid_s babba &= \alpha \mid_s \beta a, \quad \text{where } \alpha = a,\ \beta = babb, \\ aba \mid_s bba &= \alpha \mid_s \beta a, \quad \text{where } \alpha = aba,\ \beta = bb, \\ abab \mid_s ba &= \alpha \mid_s \beta a, \quad \text{where } \alpha = abab,\ \beta = b. \end{aligned}$$

It can be directly checked that (2.6) is not satisfied for all of these decompositions.

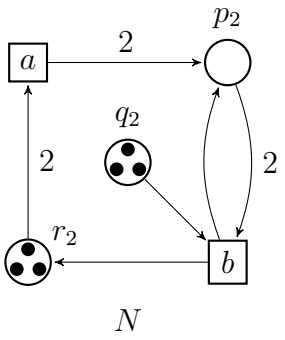


Figure 2.19: Net N solves *ababba*.

Some more structural (language-theoretical) properties about the shapes of (un-)solvable sequences can be obtained. It is easy to see from the list [P$^+$15] that there can be factors *aa* or *bb* inside a minimal unsolvable word. However, the next proposition (together with the previous proposition) implies that we cannot have both – unless one of them is at the very end of the word, as in *abbaa*.

Proposition 6. [BBE$^+$15](no aa and bb inside a minimal unsolvable word) *If a minimal unsolvable word is of the form $u = a\alpha a$, then either α does not contain the factor aa or α does not contain the factor bb.*

Proof. By contraposition. Assume that α contains a factor aa and a factor bb. Two cases are possible:

Case 1: There is a block of a's which follows a block of b's. Let a^m and b^n be such blocks, assume that b^n is followed by a^m and that there are no blocks of a's

or of b's between them. Then u is of the following form

$$|_{s_0} \ \ldots \ |_q \ ab^n(ab)^k a^m \ |_r \ \ldots$$

where $n, m \geq 2, \ k \geq 0$. Recombine the letters in u to the following form:

$$|_{s_0} \ \ldots \ |_q \ (ab)b^{n-2}(ba)^{k+1}aa^{m-2} \ |_r \ \ldots$$

Since u ends with a, and the block a^m is a factor of α, $u' = (ab)b^{n-2}(ba)^{k+1}a$ is a proper subword of u. On the other hand, u' has the form $(abw)b^*(baw)^+a$, with $w = \epsilon$, which implies unsolvability of u' by Proposition 4, contradicting the minimality of u.

Case 2: All groups of a precede all groups of b. In this case u is of the form

$$aa^{x_0}ba^{x_1} \ldots ba^{x_n}b^{y_0}ab^{y_1}ab^{y_2} \ldots ab^{y_m}a$$

where at least one of x_i and one of y_j is greater than 1. Consider $\ell = \max\{i \mid x_i > 1\}$. If $\ell = 0$, we get a contradiction to Proposition 3. Hence, $\ell > 0$. Let $t = \min\{j \mid y_j > 1\}$. Then u has the form

$$|_{s_0} \ a \ldots \ |_q \ ba^{x_\ell}(ba)^{n-\ell}(ba)^t b^{y_t} \ |_r \ \ldots a$$

Recombine the letters in u to the form

$$|_{s_0} \ a \ldots \ |_q \ (ba)a^{x_\ell - 2}(ab)^{n-\ell+t+1}bb^{y_t-2} \ |_r \ \ldots a$$

Notice that $n-\ell+t+1 \geq 1$. Hence, u has a proper subword $(ba)a^{x_\ell-2}(ab)^{n-\ell+t+1}b$, which is of the form $(baw)a^*(abw)^+b$ with $w = \epsilon$, implying its unsolvability, due to Proposition 4 with inverted a and b. This again contradicts the minimality of u. □

Proposition 6 establishes that factors aa and bb cannot occur inside minimal unsolvable words simultaneously. We shall now see that solvable words may contain both aa and bb as subwords, but only if one of these subwords appears at the beginning of the word, created by the prefixing mechanism of Proposition 3. This is indeed the case:

Proposition 7. [BESW16](never aa and bb in solvable words after initial a^+) *Let $w \in \{a,b\}^*$ be a solvable word, decomposable into $w = a^n b\alpha$ with $n \geq 1$ and $\alpha \in \{a,b\}^*$. Then, $b\alpha$ does not contain the factor aa or it does not contain the factor bb.*

Proof. Assume w contains both factors aa and bb in $b\alpha$. Select 'neighbouring' factors aa and bb, such there is no other factor aa or bb in between. Since neither chosen factor is at the start of the word, w can be decomposed into $w = \beta a b^i \underline{bb} (ab)^j \underline{aa} \gamma$ or $w = \beta b a^i \underline{aa} (ba)^j \underline{bb} \gamma$ with $\beta, \gamma \in \{a,b\}^*$ and $i, j \geq 0$. The neighbours aa and bb have been underlined. W.l.o.g. let us assume the latter form.

Let $N = (P, T, F, M_0)$ be a Petri net solving w and select states s, s', and s'' such that

$$w = \beta|_{s'} b a^i a|_s a (ba)^j b|_{s''} b\gamma.$$

Since b cannot fire at s, there must be a place $p \in P$ with $M_s(p) < b_-$ (compare Fig. 2.18). At s' and s'' the transition b can fire, so $M_{s'}(p) \geq b_- \leq M_{s''}(p)$ holds. As firing a enables b again after s, a must produce tokens on p and $\mathbb{E}_p(a) > 0$. Since b does not remain enabled from s' to s, it has to consume tokens from p, so $\mathbb{E}_p(b) < 0$. Computing the token differences on p between our chosen states we then obtain

$$0 > M_s(p) - M_{s'}(p) = (i+1) \cdot \mathbb{E}_p(a) + \mathbb{E}_p(b) \quad \text{and}$$

$$0 < M_{s''}(p) - M_s(p) = (j+1) \cdot \mathbb{E}_p(a) + (j+1) \cdot \mathbb{E}_p(b).$$

Comparing the lines gives $\mathbb{E}_p(a) > -\mathbb{E}_p(b) > (i+1) \cdot \mathbb{E}_p(a)$, which is a contradiction to $\mathbb{E}_p(a) > 0$, i.e. w is not solvable. □

This fact reduces the potentially solvable words to the regular expression

$$a^* b^+ (ab^+)^* (a|\epsilon) \quad | \quad b^* a^+ (ba^+)^* (b|\epsilon) \quad | \quad \epsilon,$$

where in the first subexpression aa may only occur at the beginning of the word and in the second one the roles of a and b are switched. The following results are shown for the first expression only, but hold for both.

If we compare two different blocks of the form ab^+ in the regular expression we find that their lengths must be nearly equal.

Lemma 1. [BESW16](block lengths differ by at most 1)
Let $w \in a^*b^+(ab^+)^*(a|\epsilon)$ be a word that contains both bab^ia and abb^ib with $i \geq 1$ as subwords. Then, w is not solvable.

Proof. Two cases are possible.

Case 1: Consider first the situation where bab^ia precedes abb^ib, i.e.

$$w = \alpha|_s bab^i|_{s'}(abb^i)^k abb^i|_{s''} b\beta$$

with $\alpha, \beta \in \{a, b\}^*$. If there are more or less than $i+1$ b's in any of the intermediate $k \geq 0$ blocks we can choose factors bab^ia and abb^ib that are closer together (possibly even having an a in common). Assume p to be a place of a Petri net solving w with $M_{s'}(p) < b_- \leq M_s(p)$, i.e. $\mathbb{E}_p(bab^i) = M_{s'}(p) - M_s(p) < 0$. Due to Parikh equivalence, $\mathbb{E}_p(bab^i) = \mathbb{E}_p(abb^i)$, we know

$$M_{s''}(p) = M_{s'}(p) + (k+1) \cdot \mathbb{E}_p(abb^i) < M_{s'}(p) < b_-,$$

which is a contradiction to b being enabled at s''.

Case 2: The next possibility,

$$w = \alpha|_s abb^ib^j(bab^i)^k|_{s'} bab^i|_{s''} a\beta$$

with $\alpha, \beta \in \{a, b\}^*$ and $j, k \geq 0$, we also obtain by choosing the factors – first abb^ib, then bab^ia this time – as close together as possible. Assume p a place with $M_{s'}(p) < a_- \leq M_{s''}(p)$, then with $M_{s''}(p) - M_{s'}(p) = \mathbb{E}_p(bab^i) = \mathbb{E}_p(abb^i) > 0$ and $\mathbb{E}_p(b) > 0$ (since firing b at s' enables a), we obtain $M_s(p) = M_{s'}(p) - k \cdot \mathbb{E}_p(bab^i) - j \cdot \mathbb{E}_p(b) - \mathbb{E}_p(abb^i) < M_{s'}(p) < a_-$. This contradicts a being enabled at s. □

Solvable words must then fulfill a kind of balancing property where the blocks of b's almost all have almost the same length.

Proposition 8. (balancing property) *Let $w = a^kb^{x_1}ab^{x_2} \dots ab^{x_n}$ with $k \geq 0$, $n \geq 2$, $x_1, \dots, x_n \geq 1$. Then, the following hold:*

1. *w solvable $\Rightarrow x_j - 1 \leq x_i$ for $2 \leq i \leq n-1$, $2 \leq j \leq n$.*

2. *wa solvable* $\Rightarrow x_j - 1 \leq x_i$ *for* $2 \leq i, j \leq n$.

3. *If* $k > 0$ *the above implications also hold for* $j = 1$.

Proof. Assume there are i and j such that one of the above implications does not hold. Then, w (or wa) contains the subwords $bab^{x_i}a$ (since $i \geq 2$) as well as $abb^{x_i}b$ as a (possibly trivial) prefix of ab^{x_j}. Lemma 1 shows that the word is not solvable, yielding a contradiction. □

For solvable words the first block of b's can have arbitrary length. For instance, the words

$$abab^9ab^9ab^9a \qquad \text{and} \qquad b^9abbabbabba$$

both are solvable, though the length of the first block is less than the lenghts of the inner blocks for the former word and bigger than the lengths of the inner blocks for the latter word. The last block of b's cannot be longer, but it can be much shorter than the average b-block if no final a follows, i.e.

$$ab^9ab^9ab^9ab$$

is solvable while

$$ababab^9$$

is not. In the former case, we may even append some more b's, which is confirmed by the following lemma.

Lemma 2. [BESW16](prolonging the last b-block) *Let* $w = a^kb^{x_1}ab^{x_2}a \dots ab^{x_n}$ *be a solvable word with* $k \geq 0$ *and* $x_i - 1 > x_n$ *for all* $1 \leq i < n$. *Then,* $w' = a^kb^{x_1}ab^{x_2}a \dots ab^{x_n+1}$ *is solvable.*

Proof. Consider the case $k \leq 1$ first. Assume $N = (P, T, F, M_0)$ to be a Petri net solving

$$w = a^kb^{x_1}ab^{x_2} \dots ab^{x_{i-1}}|_{s'}ab^{x_i-1}|_{s''}ba \dots b|_sab^{x_n}|_f$$

with a place p that prevents b at some s' before s. Then, $M_{s'}(p) < b_-$ and $M_{s''}(p) = M_{s'}(p) + \mathbb{E}_p(ab^{x_i-1}) \geq b_-$, i.e. $\mathbb{E}_p(ab^{x_i-1}) > 0$. With $M_s(p) \geq b_+$ (b fires directly before s) and $\mathbb{E}_p(b) < 0$ (b fires directly before s'), we conclude

$$M_f(p) = M_s(p) + \mathbb{E}_p(ab^{x_n}) \geq b_+ + \mathbb{E}_p(ab^{x_n}) = b_+ + \mathbb{E}_p(ab^{x_i-1}) - \mathbb{E}_p(b^{x_i-1-x_n}) > \\ > b_+ - (x_i - 1 - x_n) \cdot \mathbb{E}_p(b) \geq b_+ - \mathbb{E}_p(b) = b_-$$

Therefore, a place p preventing b at such s' cannot prevent b at the end of w. At s, b can be prevented by a new place p' with $b_- = 1$, $b_+ = 0$, $a_- = 0$, $a_+ = \min\{x_1, \ldots, x_{n-1}\}$, and an initial token count of

$$\sum_{i=1}^{n-1} x_i - (n - 2 + k) \cdot \min\{x_1, \ldots, x_{n-1}\}$$

which is non-negative. Then, $M_s(p') = 0$ and $M_f(p') = M_s(p') + a_+ - x_n > 0$. A place q preventing a (except after the last a) must have $\mathbb{E}_q(b) > 0$, so it cannot prevent b at the end either. After the last a, a new place with $\#_a(w)$ initial tokens, $a_- = 1$, and $a_+ = 0$ can disable any further a. With these modifications, a place preventing b at the end of the word w is not needed to prevent any other occurrence of a or b any more. We can now delete all places preventing b at the end of w from N and create a new place with $1 + \sum_{i=1}^{n} x_i$ tokens, $b_- = 1$, and $b_+ = 0$, to prevent b after w' is complete. The modified Petri net solves w'.

In case $k > 1$, we cut off all leading a's but one, apply the above proof, and then reprepend the missing a's using Proposition 3. □

Let us now define morphic mappings for the sequences over the same alphabets.

Definition 19. *For two alphabets Σ_1 and Σ_2, a mapping $\phi : \Sigma_1^* \to \Sigma_2^*$ is called a* morphism *if we have $\phi(u \cdot v) = \phi(u) \cdot \phi(v)$ for every $u, v \in \Sigma_1^*$, whenever all operations are defined.*

A morphism ϕ is uniquely determined by its values on the alphabet. Moreover, ϕ maps the neutral element (the empty sequence) of Σ_1^* into the neutral element of Σ_2^*. It turns out that some morphic modifications of sequences preserve their solvability. The following lemma shows that deleting one b from each block of b's will also not turn a word unsolvable.

Lemma 3. [BESW16](length reduction of b-blocks)
Let $w = a^k b^{x_1} a b^{x_2} a \dots a b^{x_n} a^j$ with $j \in \{0, 1\}$, $k \geq 0$, and $x_1, \dots, x_n \geq 2$ be a solvable word. Then, also $w' = a^k b^{x_1-1} a b^{x_2-1}\ a \dots a b^{x_n-1} a^j$ is solvable.

Proof. For $k > 1$, cut off all leading a's but one, apply the following proof for $k = 1$, and reprepend the missing a's using Proposition 3. So, let now $k \leq 1$. In case $j = 1$, we apply the proof to the word wb [and w'], which by Lemma 2 and Proposition 1 is solvable if and only if w is. If $k = 0$ we use the words w and bw', where $k = 0$ and $j = 1$ are, of course, combinable, and w' is solvable if bw' is. After applying the above modifications, note that with the homomorphism $h(a) = ab$ and $h(b) = b$ holds $h(w') = w$.

Let N be a Petri net solving w. For each place p with arc weights a_+, a_-, b_+, and b_- let $i_p = \max\{0, -a_+ - \mathbb{E}_p(b)\}$ and define a place p' for a new Petri net N' with $M_0'(p') = M_0(p) + i_p$, $b'_- = b_- + i_p$, $b'_+ = b_+ + i_p$, $a'_- = a_- + i_p$, and $a'_+ = a_+ + \mathbb{E}_p(b) + i_p$. In all cases, $a'_+ - a'_- = \mathbb{E}_p(ab)$ and $b'_+ - b'_- = \mathbb{E}_p(b)$ and all new arc weights (especially a'_+) are non-negative. By induction over the length of prefixes of w', the state reached in N' after some prefix v of w' is the state reached in N after the according prefix $h(v)$ of w plus the additional i_p. We conclude that w' and only w' can fire in N', i.e. N' solves w'. □

The following example shows how some homomorphic mappings of sequences can preserve solvability.

Example 14. *Let us consider the sequence*

$$w = abbbabbabbabbbabba$$

which is solvable, for instance, with the Petri net N in Fig. 2.20. According to the construction from the proof of Lemma 3, in order to obtain a net solving the sequence $w' = abbababbabba$ where each block of b is 1 letter shorter than the corresponding block of w, we have to append one more b at the end of w. Notice, that $h(w') = wb$, where $h(a) = ab$ and $h(b) = b$. We shall construct places p'_0, p'_1 and p'_2 by modifying initial markings and arc weights of places p_0, p_1, p_2, respectively:

$$
\begin{aligned}
i_p &= \max\{0, -a_+ - \mathbb{E}_p(b)\} \\
M_0(p') &= M_0(p) + i_p \\
b'_- &= b_- + i_p \\
b'_+ &= b_+ + i_p \\
a'_- &= a_- + i_p \\
a'_+ &= a_+ + \mathbb{E}(b) + i_p
\end{aligned}
$$

The resulting net N' with places p'_0, p'_1, p'_2 is shown in Fig. 2.20.

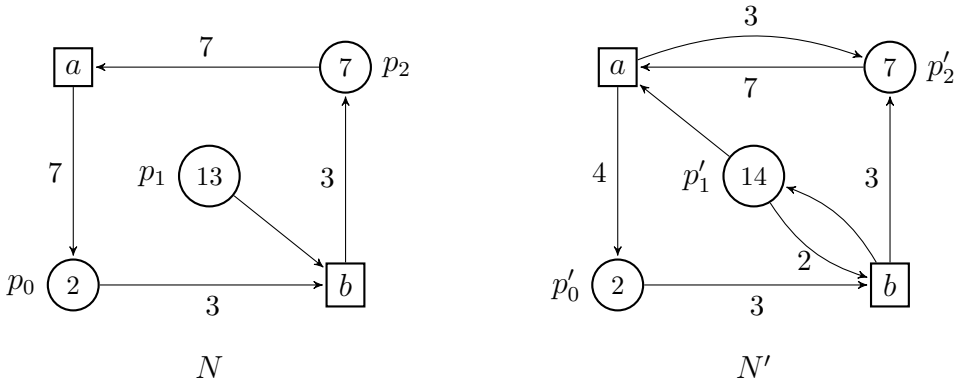


Figure 2.20: N solves sequence *abbbabbabbabbbabba*, N' solves sequence *abbababababbaba* with reduced lengths of b-blocks.

In this section we have seen that solvability of an lts from the considered class (the ones originated from finite binary sequences) is indeed connected with the structural characteristics (language-theoretical properties) of the lts in many various ways. Among such a characteristics we have seen the involving of a pattern (2.5) as a subsequence, the fraction (2.6) of the letters inside the sequence, the lengths of the blocks made of the same letter (see Lemma 1 and Proposition 8). Moreover, we have seen how some of the structural properties of the lts can be simulated in the structure of a Petri net solving it: like prepending of the initial letter to the solvable sequence, or reducing the length of the blocks by the same number.

The set of conditions for the synthesisability of a word with a Petri net described above can be summarised into one criterion. This criterion sets a number of

restrictions for the shape of a binary word, which are necessary conditions for its synthesisability. These conditions all together can be checked in a linear (in the length of a given word) time.

Theorem 2. [BESW16](linear time necessary criterion)
If a word $w \in \{a, b\}^*$ *is solvable, it is the empty word* $w = \epsilon$ *or it has the form* $w = a^k b^{x_1} a b^{x_2} a \dots a b^{x_n} a^j$ *or* $w = b^k a^{x_1} b a^{x_2} b \dots b a^{x_n} b^j$*, where* $j, k, n, x_1, \dots, x_n \in \mathbb{N}$ *with* $j \leq 1$*,* $n \geq 0$ *and there is some* $x \in \mathbb{N}$ *such that* $x_2, \dots, x_{n-1} \in \{x, x+1\}$ *and* $x_n \leq x+1$*. Furthermore, if* $j > 0$ *also* $x_n \in \{x, x+1\}$*, and if* $k > 0$ *also* $x_1 \leq x+1$*.*

The criterion is in linear time as we can detect the structure of a word w by going over it once from left to right. Remembering the block lengths that occurred so far allows us to check if the next block also has a valid length.

What we do not know so far is when a block may have length $x+1$ in the criterion, and when only length x is allowed. For instance

abababbabba,
ababbababba,
ababbabbaba,
abbababbaba,
abbababbba

are solvable sequences while

abbabbababa

is unsolvable. One could suspect that the high number of early b's makes the latter word unsolvable. But the sequence

abbabbabbaba

is solvable, though its early b-blocks are relatively (within the sequence) long. Hence, the dependency between the order of short and long blocks within the given sequence and the solvability of the sequence is not as straightforward as the first intuition says. This dependency will be made precise in the following section.

2.6 A necessary and sufficient condition for separability

While the previous section establishes some sufficient and some necessary conditions for synthesisability and non-synthesisability of an entire word, this section is dedicated to the investigation of the separability of a single transition at some state, or, in terms of the theory of regions, solvability of a single instance of ESSP. Although the theory of regions outsources this question to the solvability of a system of linear inequalities, we shall find a more straightforward condition for the separability of a transition. This condition relies on the shape of a given word, and can be checked without involving any intermediate means, i.e. it can be described language-theoretically.

The notion of the separability of a label at a given state of a labelled transition system can be naturally translated in the separability of a letter within the given sequence as follows

Definition 20. *For a decomposition $w = u|_s xv$ of a sequence $w \in \{a,b\}^*$ with $x \in \{a,b\}$ and $u, v \in \{a,b\}^+$, we call $y \in \{a,b\}$ with $y \neq x$ separable at state s iff we can construct a Petri net N with transitions a and b and one place p such that w can be fired completely in N and y is not enabled at s.*

Existence of a place which separates a letter at some state does not imply the solvability of the whole sequence. It does not even imply that this letter is separated at the other states when necessary. Similarly to the solvability of lts, for the solvability of a given sequence w, one has to provide a solution for each instance of ESSP in $TS(w)$.

Example 15. *As an instance consider sequence $w = abba|_{s_1} babb|_{s_5}$, and check separability of a at state s_1. A possible Petri net with a place q separating a at s_1 is depicted in Fig. 2.21. Although this net allows the sequence w for firing, it is nevertheless not a solution of w, i.e. $RG(N) \ncong TS(w)$. Indeed, besides word w, N also allows for firing for example sequence $abbb$. Place q does not solve the instance $\neg s_5[a\rangle$, i.e. in N transition a can fire after w. Moreover, N allows infinite sequences for firing.*

The following lemma supports our earlier intuition that the inequality (2.6),

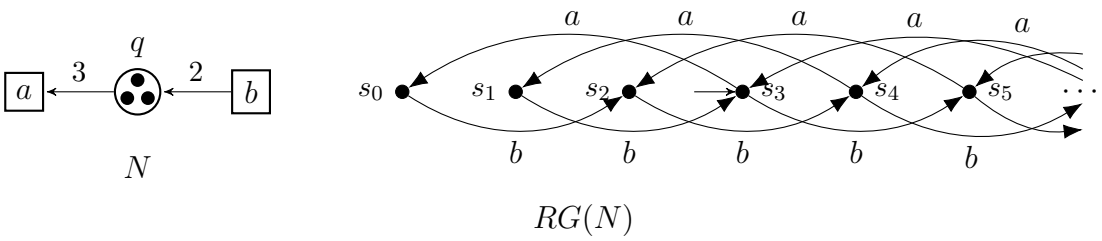


Figure 2.21: a is separable at state s_1 in $abba|_{s_1}babb$.

which has been proved to be a sufficient condition for the unsolvability of a sequence, can be extended to a characterisation of solvability of binary sequences. More precisely, it gives rise to a criterion for the separability of a label at a state. In the lemma (the negation of) (2.6) is proved to be a necessary and sufficient condition for the solvability of an instance of ESSP. The sufficiency of the criterion is proved constructively, i.e. a procedure for creating a place of the sought Petri net is presented.

Lemma 4. [BESW16](characterisation of separable states) *For a word $w \in \{a,b\}^*$ let $w = u|_s xv$ be an arbitrary decomposition with $x \in \{a,b\}$. Let $y \in \{a,b\}$ with $y \neq x$ be the other letter in our alphabet. Then, y is separable at s if and only if*

$$\forall \alpha, \beta, \gamma, \delta : (w = \alpha y \beta|_s x \gamma y \delta \;\Rightarrow\; \#_y(y\beta) \cdot \#_x(x\gamma) > \#_x(y\beta) \cdot \#_y(x\gamma)).$$

Proof. W.l.o.g. we assume that $\#_x(w) \cdot \#_y(w) > 0$.

"⇒": Let p be a place (of some Petri net) enabling y at s' and s'' but not at s in a decomposition

$$w = \alpha|_{s'} y \beta|_s x \gamma|_{s''} y \delta.$$

Since p disables y at s but not at s'', the number of tokens on p must increase from s to s'', and also from s to the first y after s, where only letters x are present. Thus, x effectively increases the token count on p, i.e. $\mathbb{E}_p(x) > 0$.

Assume firing y would not lower the token count on p. Since y is enabled at s', it will also be enabled at every state afterwards, even at s. So, p would not disable y at s. We conclude that y effectively removes tokens from p, i.e. $\mathbb{E}_p(y) < 0$.

Since y can fire at s' but not at s, tokens are consumed by $y\beta$, i.e.

$$\#_y(y\beta) \cdot (-\mathbb{E}_p(y)) > \#_x(y\beta) \cdot \mathbb{E}_p(x).$$

From s to s'', for analogous reasons, tokens are produced on p, so

$$\#_x(x\gamma) \cdot \mathbb{E}_p(x) > \#_y(x\gamma) \cdot (-\mathbb{E}_p(y)).$$

Multiply the first inequation by $\#_x(x\gamma)$ and the second one by $\#_x(y\beta)$, then divide both by $-\mathbb{E}_p(y)$ to make them comparable:

$$\#_y(y\beta) \cdot \#_x(x\gamma) > \#_x(y\beta) \cdot \frac{\mathbb{E}_p(x)}{-\mathbb{E}_p(y)} \cdot \#_x(x\gamma) > \#_x(y\beta) \cdot \#_y(x\gamma).$$

"$\Leftarrow$": Let

$$S' = \{s' \mid \exists \alpha, \beta : \ w = \alpha|_{s'} y\beta|_s xv\} \quad \text{and} \quad S'' = \{s'' \mid \exists \gamma, \delta : \ w = u|_s x\gamma|_{s''} y\delta\}.$$

Denoting by $\#_x(s', s)$ the number of occurrences of x between states s' and s (and analogously for y and for pairs of states (s, s'')), let us define ratios of y and x in $\mathbb{Q} \cup \{-\infty, \infty\}$ via

$$r_{max}(s) = \min_{s' \in S'} \left\{ \frac{\#_y(s', s)}{\#_x(s', s)} \right\} \quad \text{and} \quad r_{min}(s) = \max_{s'' \in S''} \left\{ \frac{\#_y(s, s'')}{\#_x(s, s'')} \right\}.$$

In case $S' = \emptyset$ we assume the minimum $r_{max}(s)$ to be ∞ as a default value, if $S'' = \emptyset$ the maximum $r_{min}(s)$ will be $-\infty$. $\#_x(s, s'')$ and $\#_y(s', s)$ cannot both be zero (as there is an x directly after s and at least one y after s' or s''), so no ambiguous fraction $\frac{0}{0}$ can occur. If $\#_x(s', s)$ is zero, we assume the default value of ∞ for this fraction. See Fig. 2.22 for a visualisation.

We show now that $r_{max}(s) > r_{min}(s)$. This is trivial in case one of the two assumes its default value ∞ or $-\infty$. Otherwise, for all decompositions

$$w = \alpha|_{s'} y\beta|_s x\gamma|_{s''} y\delta \ \text{ with } s' \in S' \ \text{ and } \ s'' \in S''$$

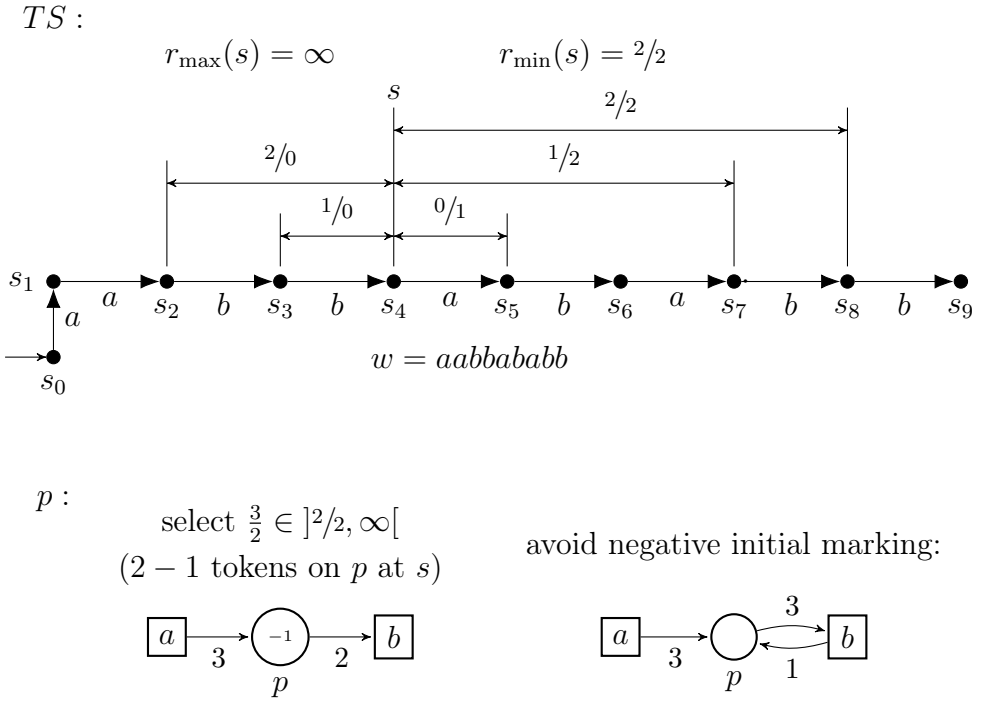


Figure 2.22: TS corresponding to *aabbababb* with a state s at which b must not occur (here in our notations $x = a$, $y = b$ and $s = s_4$). We compute maximal/minimal b/a-ratios $r_{min}(s)/r_{max}(s)$ for words starting with b ending at s and starting at s ending in front of a b, respectively. The production/consumption ratio for a place p in a Petri net prohibiting b at s must fall into the open interval $]r_{min}(s), r_{max}(s)[$. A loop around b can be added to prevent a negative initial marking.

the following holds

$$\#_y(y\beta) \cdot \#_x(x\gamma) > \#_x(y\beta) \cdot \#_y(x\gamma).$$

We now select those $s' \in S'$ and $s'' \in S''$ that yield the ratio values $r_{max}(s)$ and $r_{min}(s)$ in the above definitions, respectively. For these two states we obtain:

$$r_{max}(s) = \frac{\#_y(s', s)}{\#_x(s', s)} = \frac{\#_y(y\beta)}{\#_x(y\beta)} > \frac{\#_y(x\gamma)}{\#_x(x\gamma)} = \frac{\#_y(s, s'')}{\#_x(s, s'')} = r_{min}(s).$$

We now create a Petri net with two transitions x and y and a single place p that will disable y at s but not at any other state in $S' \cup S''$, and also will not disable x at all. Such a Petri net (among others) allows w for firing. In a first

step, let us choose arc weights $y_- \in \mathbb{N}^+$ (from p to y) and $x_+ \in \mathbb{N}^+$ (from x to p) such that

$$r_{max}(s) > \frac{x_+}{y_-} > r_{min}(s),$$

which is possible, compare Fig. 2.22. Furthermore, let us assume there are $y_- - 1$ tokens on p at state s, so p disables y at s. Choose any state $s' \in S'$, then

$$\frac{\#_y(s', s)}{\#_x(s', s)} \geq r_{max}(s) > \frac{x_+}{y_-}.$$

In case $\#_x(s', s) > 0$, we can multiply with this value and with y_- to obtain

$$\#_y(s', s) \cdot y_- > \#_x(s', s) \cdot x_+.$$

In case $\#_x(s', s) = 0$ the above inequality is true, since s' is immediately followed by a y. The inequality shows that there are more tokens on p in s' than in s. Due to our choice of $y_- - 1$ tokens for s, y is not disabled at s' by p.

Accordingly, for a state $s'' \in S''$, we have

$$\frac{x_+}{y_-} > r_{min}(s) \geq \frac{\#_y(s, s'')}{\#_x(s, s'')}$$

and by multiplying with the non-zero denominators we get

$$\#_x(s, s'') \cdot x_+ > \#_y(s, s'') \cdot y_-.$$

So, at s'' there are more tokens on p than at s, and p cannot disable y at s''.

It remains to be shown that there are always at least zero tokens on p at any possible state. This is already known for all states from $S' \cup S''$ (having at least y_- tokens) and for all states $\hat{s}$ immediately following a state from $S' \cup S''$ (only y_- tokens are consumed). Since from such an $\hat{s}$ until the next state in $S' \cup S''$ only x occurs in the word w, the number of tokens will only be increased. So, all states beginning with the first state from $S' \cup S''$ in the word w are covered. Before this first state, only letters x occur in w, so it suffices to check if the initial state of the Petri net has at least zero tokens on p.

If the initial state s_0 is in $S' \cup S''$, we are done. Otherwise, we compute the

initial number of tokens via $M_s(p) = y_- - 1$ in w:

$$n = y_- - 1 + \#_y(s_0, s) \cdot y_- - \#_x(s_0, s) \cdot x_+.$$

Only in case of an initial marking $n < 0$ we have a problem. This can be easily solved, though, by creating an arc from y to p with weight $F(y, p) = -n$ and replacing the values for the reverse arc weight and the initial marking by $F(p, y) = y_- - n$ and $M_0(p) = n - n = 0$. The additional $-n$ tokens are never used up but are always needed for y, so they will neither allow any additional firing of y nor prevent any required one. □

We have now formulated an exact criterion for separability of a transition at some state, which is based exclusively on the shape of the word under consideration, i.e. we have provided a language-theoretical (or graph-theoretical, when talking about labelled transition systems) condition for solvability of an instance of ESSP without involving regions. The next step is to gather the results of the previous investigations for establishing a solvability criterion of the sequences.

The following theorem, being a natural continuation of Lemma 4, presents a criterion for the synthesisability of a given binary word, which is based on the graph-theoretical properties of the word and does not involve the region mechanism.

Theorem 3. [BESW16](characterisation of solvable words)
A word $w \in \{a, b\}^*$ is Petri net solvable if and only if the following formula holds for $x = a \wedge y = b$ as well as for $x = b \wedge y = a$:

$$\forall \alpha, \beta, \gamma, \delta : (w = \alpha y \beta x \gamma y \delta \;\Rightarrow\; \#_y(y\beta) \cdot \#_x(x\gamma) > \#_x(y\beta) \cdot \#_y(x\gamma)).$$

The theorem serves as a theoretical characterisation of the class of solvable finite sequences over 2-letters alphabet. Based on the theorem, in the following section we shall present an algorithm for Petri net synthesis from such sequences.

2.7 A letter-counting based synthesis algorithm

The criterion for the synthesisability of binary words provides an algorithm that takes a word as an input, and produces a Petri net which solves this word as an

output. This algorithm solves all necessary separability problems consecutively by constructing a place for each such problem. Moreover, a modified version of the algorithm allows to combine separability problems into groups and construct a single common solution for all problems of a group. Beside the theoretical estimation of the complexity of algorithm, some experimental results of the described algorithm are provided. These results compare the new algorithm with a general linear-algebraic approach from the theory of regions, demonstrating the relative promptness of the new algorithm for constructing a Petri net as well as for quickly checking for the synthesisability of a word in general, without producing a net.

As we have seen in Section 2.2 the same region, and hence the same place of a Petri net can solve several separation problems, i.e. it is possible to disable for several transitions their firings at some marking of a Petri net, or to disable the firings of some transition at several distinct markings (states of an lts, respectively) with some place of the net. The following proposition establishes the condition when the same place of a Petri net can be used for the separation of a transition (letter) at distinct states of a given binary word.

Proposition 9. (shared separating places) *Let $w = u|_s xv|_{\hat{s}} xz$ be a solvable word with $x \in \{a, b\}$ and with two states s, $\hat{s}$ after which the same letter x occurs. Then, for s and $\hat{s}$, we can use the same place for the separation if and only if the open intervals $]r_{min}(s), r_{max}(s)[$ and $]r_{min}(\hat{s}), r_{max}(\hat{s})[$ (from the proof of Lemma 4) have a non-empty intersection.*

Proof. The first direction of the proof of Lemma 4 shows that for state s ($\hat{s}$) the arc weight ratio $\frac{x_+}{y_-}$ of the occurring letter x compared to the separation letter y must lie inside the open interval $]r_{min}(s), r_{max}(s)[$ ($]r_{min}(\hat{s}), r_{max}(\hat{s})[$, respectively). If one separation place is enough for both states, the arc weight ratio must fall into both open intervals. Similarly, if the intervals have a non-empty intersection, the arc weight ratios in the second part of the proof of Lemma 4 can be chosen identical, so the same place is generated for both states. The different separation states may require a different number of loops at y to prevent a negative initial marking. In this case, the higher number of loops will always suffice. □

The algorithm 1 – `ABSolve` – for the Petri net synthesis for a finite word w has runtime in $O(|w|^2)$ because the first part of the algorithm (with the **for**-loops) is quadratic, and the **while**-loop is executed up to $|w|$ times. For the choice of I,

Algorithm 1 ABSolve

Input: $w \in \{a,b\}^*$
Output: A Petri net $N = (P, \{a,b\}, F, M_0)$ solving w if it exists

$P \leftarrow \emptyset$, $F \leftarrow \emptyset$, $M_0 \leftarrow \emptyset$
for $i = 0$ **to** $|w| - 1$ **do** {separation point s}
$r_{min}[i] \leftarrow -\infty$, $r_{max}[i] \leftarrow \infty$ {defaults}
$N[0] \leftarrow 0$, $N[1] \leftarrow 0$ {for counting a's and b's}
if $w[i] = 'a'$ **then** $R \leftarrow 1$ **else** $R \leftarrow 0$ {fraction selector}
for $j = i - 1$ **down to** 0 **do** {compute r_{max}}
if $w[j] = 'a'$ **then** $N[0] \leftarrow N[0] + 1$ **else** $N[1] \leftarrow N[1] + 1$
if $w[j] \neq w[i]$ **and** $r_{max}[i] > \frac{N[R]}{N[1-R]}$ **then** $r_{max}[i] \leftarrow \frac{N[R]}{N[1-R]}$
endfor
$N[0] \leftarrow 0$, $N[1] \leftarrow 0$
for $j = i + 1$ **to** $|w| - 1$ **do** {compute r_{min}}
if $w[j-1] = 'a'$ **then** $N[0] \leftarrow N[0] + 1$ **else** $N[1] \leftarrow N[1] + 1$
if $w[j] \neq w[i]$ **and** $r_{min}[i] < \frac{N[R]}{N[1-R]}$ **then** $r_{min}[i] \leftarrow \frac{N[R]}{N[1-R]}$
endfor
if $r_{min}[i] \geq r_{max}[i]$ **then return** {unsolvable}
endfor
$S \leftarrow \{0, \ldots, |w| - 1\}$ {unprocessed intervals}
while $S \neq \emptyset$ **do**
choose $I \subseteq S$ with $|\{w[i] | i \in I\}| = 1$ and $\bigcap_{i \in I}]r_{min}[i], r_{max}[i][\neq \emptyset$
$S \leftarrow S \backslash I$
choose $\frac{m}{n} \in \bigcap_{i \in I}]r_{min}[i], r_{max}[i][$
$P \leftarrow P \cup \{p_I\}$
$\ell \leftarrow w[\min I]$ {doesn't matter which $i \in I$}
if $\ell = 'a'$ **then** $F(a, p_I) \leftarrow m$, $F(p_I, b) \leftarrow n$ **else** $F(b, p_I) \leftarrow m$, $F(p_I, a) \leftarrow n$
compute the minimal $M_0(p_I) \in \mathbb{Z}$ for $i \in I$ from $M(p_I) = n - 1$
{via backward firing $M_0[w[0] \ldots w[i-1]\rangle M$}
if $M_0(p_I) < 0$ **and** $\ell = 'a'$ **then**
$F(b, p_I) \leftarrow F(b, p_I) - M_0(p_I)$, $F(p_I, b) \leftarrow F(p_I, b) - M_0(p_I)$, $M_0(p_I) \leftarrow 0$
if $M_0(p_I) < 0$ **and** $\ell = 'b'$ **then**
$F(a, p_I) \leftarrow F(a, p_I) - M_0(p_I)$, $F(p_I, a) \leftarrow F(p_I, a) - M_0(p_I)$, $M_0(p_I) \leftarrow 0$
endwhile
return $(P, \{a,b\}, F, M_0)$

select one interval and intersect consecutively with any other interval unless the intersection would become empty, resulting in $O(|w|)$ time. The choice of $\frac{m}{n}$ can be done in constant time unless some 'optimal' value is sought. The computation of $M_0(p_I)$ by backward firing is in $O(|w|)$. So, overall the **while**-loop is in $O(|w|^2)$ in the worst case.

2.7.1 Experimental results of ABSolve algorithm

In an implementation using the region-based approach, usually (as in `APT` [S$^+$13]) one of the freely available ILP solvers is used, and these normally (e.g. see [LPK00, nlA06]) implement the simplex algorithm. The simplex algorithm can have exponential run times, but typically solves a system of equations in linear time, which is much better than Karmarkar's [Kar84] worst case complexity.

To see how our algorithm fares compared to the region-based approach, we used the tools `Synet` [Cai02] and `APT` [S$^+$13], both of which can synthesise Petri nets, and let all three run on the same computer. `APT` and our algorithm `ABSolve` have been implemented in Java while `Synet` was written in OCaml, which is known to produce efficient code. For each single test that was done we randomly generated 4000 words meeting certain criteria and fed them to all tools (including a pseudo-tool "no-op" doing virtually nothing, in order to see the time spent only for reading words), trying to synthesise the whole set of words. From the composite result we computed the average run time per word.

We made tests for words in $(a|b)^l$ with a fixed word length $l \in \{1, \ldots, 700\}$, i.e. 700 tests times 4000 words per tool, and again for words from $(ab|abb)^l$, where we expected a higher probability for solvable words. (For $l = 1$ randomisation means we tested the words a and b each about 2000 times, but e.g. at $l = 50$ it is extremely unlikely that we tested the same word twice.) Fig. 2.23 shows the results for $(a|b)^l$ on a logarithmic time scale. `ABSolve` is about a factor 10^3 faster than `APT` and `Synet`, and by the same factor slower than "no-op". In Fig. 2.24 we normalised all curves by dividing all values of each curve by its value at $l = 350$ in order to see the linear time scales. We can see that `ABSolve` and `APT` both seem to have run time $O(n)$ while `Synet` shows a clearly parabolic curve, i.e. $O(n^k)$ with $k > 1$. Fig. 2.25 shows the results for $(ab|abb)^l$. The times are higher than for $(a|b)^l$, but this seems to be mostly due to the increased word length.

We then tried to compare random sets of *solvable* words with sets from all words. From about $l = 40$ upwards it takes a lot of time to randomly generate solvable words (by randomly creating words and then picking the solvable ones) as solvable words become scarce. For an enumeration of all solvable words (ordered by length) without synthesising Petri nets we need to remember all solvable words of the same length and their $r_{max}[i]$-values (in a breadth-first manner). If we append a letter x to some word w, all comparisons of r_{min} and r_{max} for wx have already been done when we inspected w, except (possibly) for the comparison of $r_{max}[i]$ with the ratio of the subword from position i to $|wx| - 1$, for each i. Starting with $i = |wx| - 1$ and counting down, these comparisons can be done in linear time. So our enumeration takes at most $O(|w|)$ time per solvable word w.

Fig. 2.26 shows that solvable words take distinctly more time with `APT` or `ABSolve` than arbitrary ones. This is the result of quick fail strategies in both algorithms (we stop checking at the first unsolvable system of equations or the first empty open interval, respectively). It also explains the linear run time for `ABSolve` and `APT` (we expect at least quadratic for solvable words) and the visible hook at the beginning of the curve for `ABSolve` in the other three pictures. `Synet` has identical times for solvable and for arbitrary words and was thus left out of the picture. A likely reason for this is the missing quick fail mechanism in this tool.

2.7.2 Synthesis of binary words with bounded Petri nets

The algorithm `ABSolve` studied earlier can be adapted for a special case of synthesis – synthesis with k-bounded Petri nets (where in every reachable marking every place has at most k tokens). This class of Petri nets is often used in applications (e.g. in process mining [vdAG07]).

The first point of modification is at the step

Algorithm 1 ABSolve

...
choose $\frac{m}{n} \in \bigcap_{i \in I}]r_{min}[i], r_{max}[i][$
...

When choosing $\frac{m}{n}$ in a run of the algorithm for synthesis of a k-bounded net, both

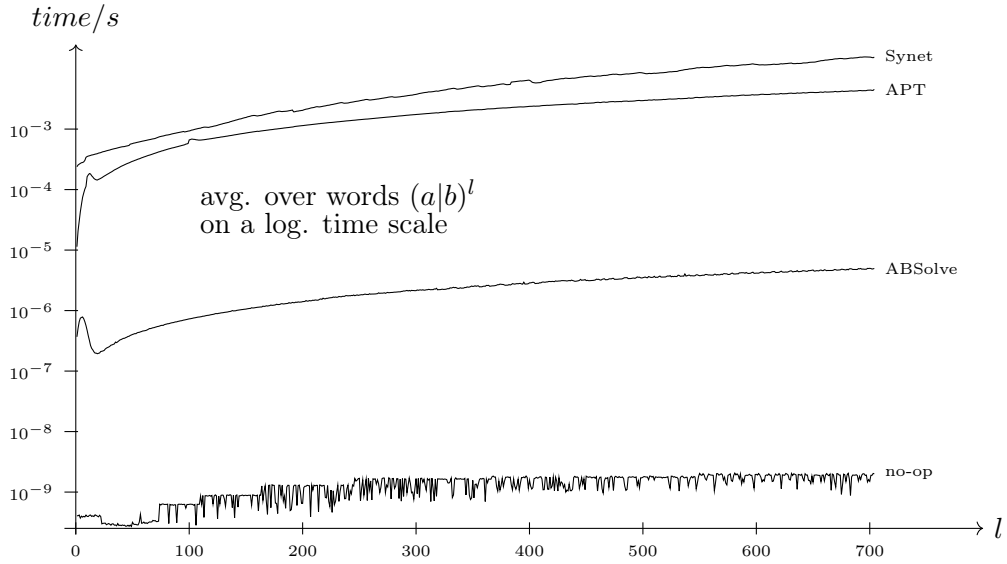


Figure 2.23: Tests were done for random sets of 4000 words for each l with $1 \leq l \leq 700$ where words stem from $(a|b)^l$. Time scale is logarithmic.

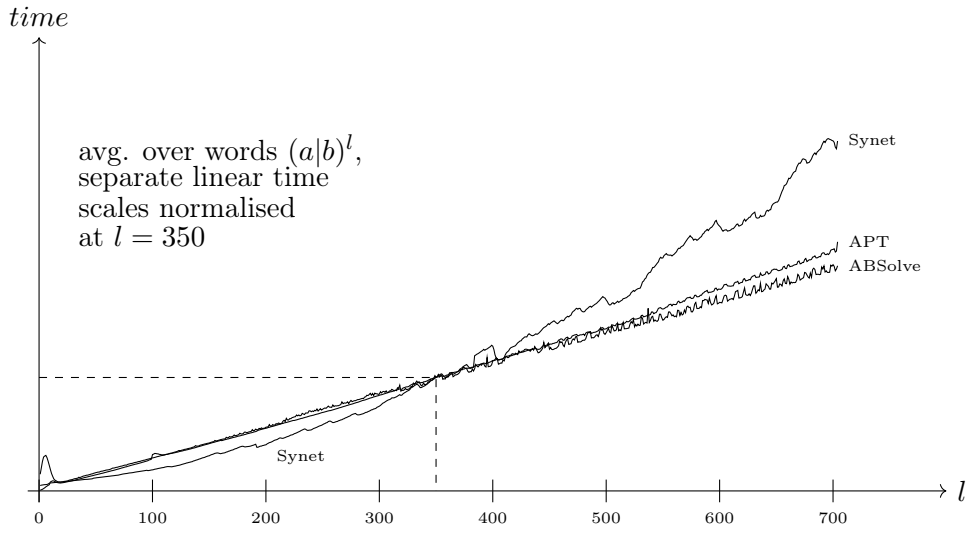


Figure 2.24: All curves from Fig. 2.23 are normalised by dividing all values of each curve by its value at $l = 350$.

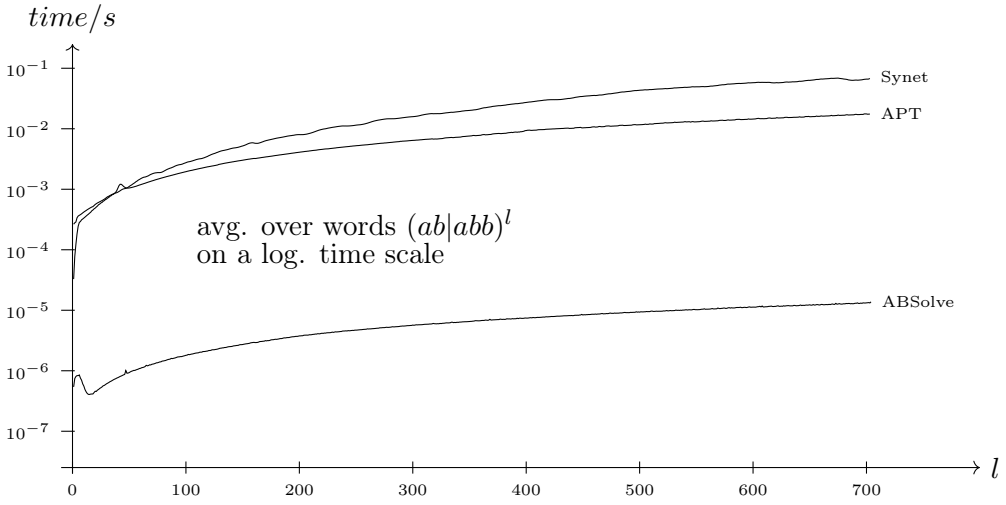


Figure 2.25: Tests were done for random sets of 4000 words for each l with $1 \leq l \leq 700$ where words stem from $(ab|abb)^l$. Time scale is logarithmic.

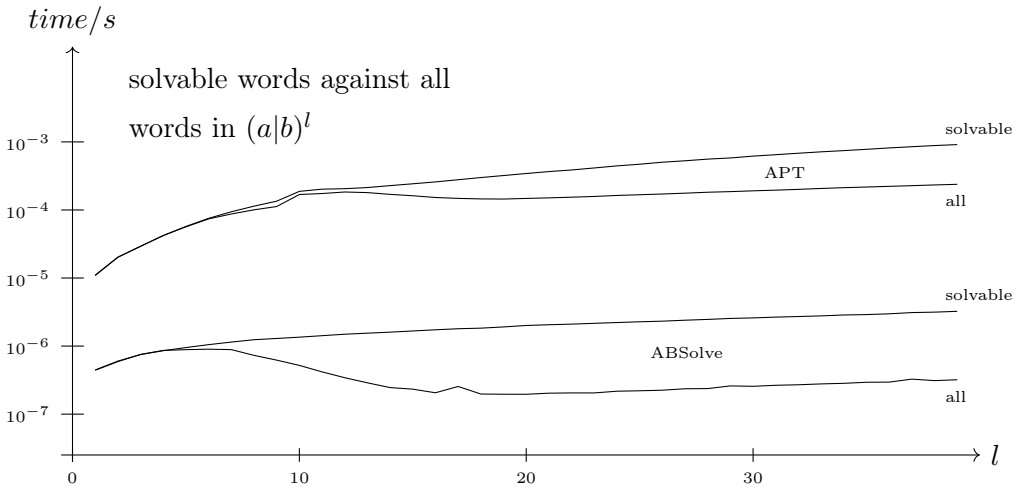


Figure 2.26: Comparison of random sets of *solvable* words with sets from all words.

$m \leq k$ and $n \leq k$ must hold, since these values are then used as arc weights of a constructed place in the sought Petri net, precisely at the following step

Algorithm 1 ABSolve

...
if $\ell = 'a'$ **then** $F(a, p_I) \leftarrow m$, $F(p_I, b) \leftarrow n$ **else** $F(b, p_I) \leftarrow m$, $F(p_I, a) \leftarrow n$
...

This implies that the number of possible values for m and n does not depend on $|w|$. We need to check, though, if the created place could have more than k tokens on it in some reachable marking of the net. This can be done in linear time for fixed m, n, by 'firing' the word and computing the maximal token difference. Unluckily, it is possible that the intersection of intervals of the form $]r_{min}(s), r_{max}(s)[$ does not allow for a valid choice of m and n while there are valid choices for each interval separately. This means that the criterion from Theorem 3 is no longer a characterisation for the synthesisability with k-bounded Petri nets. While the necessary condition of the criterion remains unchanged, the sufficient part of the criterion is not true for this restricted class of Petri nets. As an instance, consider the word $w = a \mid_s ab$ which satisfies the criterion. Assume there is a safe (1-bounded) Petri net $N_{aab} = (P, T, F, M_0)$ solving w, then a place p which separates b at state s must be present. This implies $F(p, b) > 0$. Due to the safety of N_{aab}, we have $F(p, b) = 1$. Since p forbids the firing of b at s, then p has no tokens on it at the marking M corresponding to state s. On the other hand, b has to fire in N_{aab} after the execution of aa, implying that a increases the marking of p while firing, i.e. $F(a, p) > 0$. This yields that the marking of p after the execution of aa starting from the initial marking is at least 2, which contradicts the safety of N_{aab}. Hence, w cannot be solved with any safe Petri net, although it can be solved in general case (see e.g. net N in Fig. 2.27, as a possible solution).

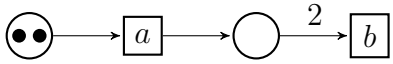


Figure 2.27: Petri net N solving aab.

The modified version of the algorithm has the changed complexity, determined by the parameter k for the bound. The first half remains unchanged. The second part was in $O(|w|^2)$ before the modification. Since both m and n in the fraction $\frac{m}{n}$ have to be non-negative integers not exceeding k now, we have to check at worst k^2 possibilities for the fraction. As we have already mentioned, validation of the place (its k-boundedness) can be done in linear time for each possible choice of the nominator and the denominator of the fraction. Hence, if we create one place for each interval we could do the second half of `ABSolve` in $O(k^2 \cdot |w|^2)$. But an optimal solution with as few places as possible is much harder to gain.

2.8 Cyclic lts over a binary alphabet

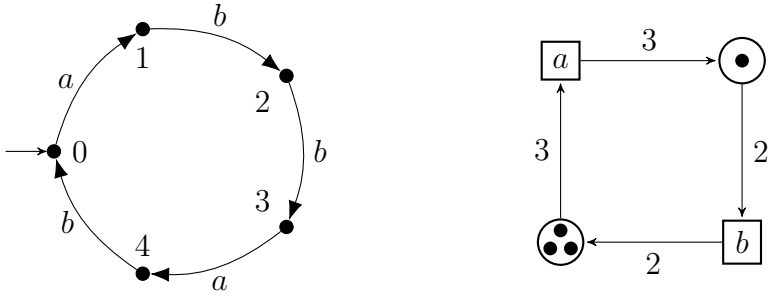
This section is devoted to another simple form of transition systems – *cycles*. For state s and s' of a transition system, a path $s \xrightarrow{\sigma} s'$ is a cycle if and only if $s = s'$. We will treat necessary and sufficient conditions for synthesisability of transition systems which are cycles or almost cycles (in the sense that beside a loop, a transition system has a linear part which leads to the loop, i.e. it has a form of lasso).

Definition 21. *A word $w = t_1 \dots t_n$ (with $t_i \in T$) is* cyclically solvable *if the transition system*

$$TS_{cyc}(w) = (\{0, \dots, n\}, T, \{(i-1, t_i, i) \mid 0 < i < n \wedge t_i \in T\} \cup \{(n, t_n, 0)\}, 0)$$

is solvable. $TS_{cyc}(w)$ represents the 'infinite word' w^ω. Word w is called minimal cyclically solvable *if it is cyclically solvable and there is no shorter word v, $|v| < |w|$, with $w^\omega = v^\omega$.*

If a word w is cyclically solvable, then a Petri net solving $TS_{cyc}(w)$ reproduces its initial marking by firing w and allows for the (infinite) firing of w^ω. For example, word *abbab* is cyclically solvable, which can be seen in Fig. 2.28: on the left, transition system $TS_{cyc}(abbab)$ is depicted which is isomorphic to the reachability graph of net N on the right, hence solvable. The net N reproduces its initial marking after every firing of sequence *abbab*.



$TS_{cyc}(abbab)$ N

Figure 2.28: An lts $TS_{cyc}(abbab)$ and Petri net N with $RG(N) \cong TS_{cyc}(abbab)$.

2.8.1 A synthesisability criterion for cyclic lts

Similarly as for a linear-shaped transition system, we provide a criterion for separability of a transition in a transition system having the shape of a cycle. This criterion, as we show, can be extended to a synthesisability criterion of this class of transition systems.

Theorem 4. [BESW16](characterisation of cyclically solvable binary words) *A word $w \in \{a,b\}^+$ is cyclically solvable if and only if $\forall x, y \in \{a,b\} \forall \alpha, \beta, \gamma, u, v$:*

$$(x \neq y \ \wedge \ w = uv \ \wedge \ vu = x\alpha y\beta) \Rightarrow \#_x(x\alpha) \cdot \#_y(w) > \#_y(x\alpha) \cdot \#_x(w).$$

Proof. "⇒": Let N be the Petri net solution for $TS_{cyc}(w)$. Due to the reproduction of the initial marking, w can be fired indefinitely in N. For $ww = uvuv$ we can investigate the decomposition of

$$vuvu = x\alpha|_{s'} y\beta|_s x\alpha|_{s''} y\beta.$$

Looking at the subword from s' to s'', by Lemma 4 we know

$$\#_y(y\beta) \cdot \#_x(x\alpha) > \#_x(y\beta) \cdot \#_y(x\alpha).$$

Since

$$\#_y(w) = \#_y(x\alpha) + \#_y(y\beta) \quad \text{and} \quad \#_x(w) = \#_x(x\alpha) + \#_x(y\beta),$$

the ratio of y by x ($\in \mathbb{Q} \cup \{\infty\}$) in w must lie between those in $x\alpha$ and $y\beta$:

$$\frac{\#_y(y\beta)}{\#_x(y\beta)} > \frac{\#_y(w)}{\#_x(w)} > \frac{\#_y(x\alpha)}{\#_x(x\alpha)}.$$

The latter inequality completes this direction of the proof.

"$\Leftarrow$": Consider a decomposition of the 'rolled out' version w^ω of w

$$\ldots|_{s'}\hat{w}^i|_{\hat{s}'}y\beta|_s x\alpha|_{\hat{s}''}\tilde{w}^j|_{s''}y\ldots$$

where $\hat{w} = y\beta\gamma$ and $\tilde{w} = \delta x\alpha$ (with some $\gamma, \delta \in \{a, b\}^*$) have the same Parikh vector as w and $i, j \geq 0$. Note that $x\alpha$ and $y\beta$ may each have a length up to $|w|-1$, so they might not concatenate up to w. We will now show that all possible finite subwords from some s' to s'' around our separation point s fulfill the condition of Lemma 4, the lemma is applicable with the result of y being separable at s.

If $x\alpha$ is a factor in w, we know

$$\#_x(x\alpha) \cdot \#_y(w) > \#_y(x\alpha) \cdot \#_x(w).$$

If $x\alpha = uv$ is distributed such that $w = vy\gamma u$, we come to the same conclusion by using the rolled version $uvy\gamma$ in the precondition. For $y\beta$, consider the rolled version $x\gamma y\beta$ of w (with γ chosen accordingly). We then know

$$\#_x(x\gamma) \cdot \#_y(w) > \#_y(x\gamma) \cdot \#_x(w)$$

and conclude that the ratio of x by y in w must be between those in $x\gamma$ and $y\beta$, i.e.

$$\frac{\#_y(y\beta)}{\#_x(y\beta)} > \frac{\#_y(w)}{\#_x(w)} > \frac{\#_y(x\gamma)}{\#_x(x\gamma)}.$$

Overall, we get

$$\frac{\#_y(y\beta)}{\#_x(y\beta)} > \frac{\#_y(w)}{\#_x(w)} > \frac{\#_y(x\alpha)}{\#_x(x\alpha)},$$

which is the precondition for Lemma 4 at $\hat{s}'$ and $\hat{s}''$. We can now argue that

$$\frac{\#_y(y\beta)}{\#_x(y\beta)} > \frac{\#_y(\hat{w}y\beta)}{\#_x(\hat{w}y\beta)} > \frac{\#_y(w)}{\#_x(w)}$$

(and analogously for $x\alpha$ and $x\alpha\tilde{w}$). But we have begun with the decomposition

$$\ldots|_{s'}\hat{w}^i|_{\hat{s}'}y\beta|_s x\alpha|_{\hat{s}''}\tilde{w}^j|_{s''}y\ldots,$$

where $\hat{w}$ and w have the same Parikh vector, so the same number of x and y are in it. The argument can be applied repeatedly until $\hat{w}^i$ and $\tilde{w}^j$ are reached, and we get

$$\frac{\#_y(\hat{w}^i y\beta)}{\#_x(\hat{w}^i y\beta)} > \frac{\#_y(w)}{\#_x(w)} > \frac{\#_y(x\alpha\tilde{w}^j)}{\#_x(x\alpha\tilde{w}^j)}.$$

So, the precondition for Lemma 4 is fulfilled for arbitrary s' and s'' that are followed by y, and by arbitrary s followed by x. Lemma 4 is applicable and y is separable at s. Since s is chosen arbitrarily (according to the decomposition $w = uv \wedge vu = x\alpha y\beta$), the infinite word w^ω is solvable and thus w is cyclic solvable. This concludes the proof. □

The theorem establishes a necessary and sufficient condition for a labelled transition system having the form of a cycle to be solvable by a Petri net. As in the case of finite sequences, exploiting the similarities of the characterisations, we shall present a Petri net synthesis algorithm originating from the condition in the theorem.

2.8.2 A synthesis algorithm for cycles

Relying on the criterion from the previous paragraph, we present a synthesis algorithm for transition systems having the form of a cycle. A theoretical estimation of the complexity of the algorithm is provided as well, being better than the one known for the general synthesis algorithm.

With increasing i and j the ratios of y and x in the words $\hat{w}^i y\beta$ and $x\alpha\tilde{w}^j$ converge against $\frac{\#_y(w)}{\#_x(w)}$ (without ever reaching it). Thus, the open interval in Lemma 4 from which we can choose the arc weight ratio for the place p to be

created turns into a single point $\frac{\#_y(w)}{\#_x(w)}$ – independent of the separation point, as long as we prevent the same transition y. We conclude:

Proposition 10. (nets for cyclically solvable words) *If $w \in \{a,b\}^+$ is cyclically solvable, there is a Petri net solving it that has at most two places. The arc weights of these places are determined by the ratios $\frac{\#_a(w)}{\#_b(w)}$ and $\frac{\#_b(w)}{\#_a(w)}$, respectively.*

In case of a finite sequence, in order to define the initial marking for each place of the sought net in algorithm `ABSolve` we applied the 'unfiring procedure'. Since we are now dealing with loops, it is unclear how can we define the termination point for such an unfiring. The following proposition gives us an understanding for choosing of the initial marking of the synthesised Petri net. Particularly, the proposition establishes that for a cyclically solvable word w, the total number of tokens on both place of the Petri net from Proposition 10 (which solves $TS_{cyc}(w)$) is a constant for any reachable marking of the net. This constant is determined by the length of w. Since the reachability graph of the net has the shape of a cycle, each of $|w|-1$ markings is revisited indefinitely (in a cycle) while the token game. Hence, the initial marking can be defined by rotation of w and the possible reachable markings.

Proposition 11. (token count for cyclic solvable words) *Let $w \in \{a,b\}^+$ be minimal cyclically solvable. There is a Petri net $N = (\{p_1,p_2\}, \{a,b\}, F, M_0)$ solving w^ω such that for all $M \in [M_0\rangle$, $M(p_1) + M(p_2) = |w| - 1$.*

Proof. From Proposition 10 we have a Petri net solution with two places and two transitions and know that we may choose arc weights $F(p_1,a) = F(a,p_2) = \#_b(w)$ and $F(p_2,b) = F(b,p_1) = \#_a(w)$. Thus,

$$\forall M \in [M_0\rangle\colon\ M(p_1) + M(p_2) = M_0(p_1) + M_0(p_2).$$

Let $w[i]$ be the ith letter of w and let M_i markings with $M_{i-1}[w[i]\rangle M_i$ for $1 \leq i \leq |w|$. Then, $M_0 = M_{|w|}$ and due to the minimal cyclic solvability of w, $M_i \neq M_j$ for $0 \leq i < j < |w|$ (otherwise for some rotation v of $w[i{+}1]\dots w[j]$ we have $v^\omega = w^\omega$). We conclude $|[M_0\rangle| = |w|$. Since $|P| = 2$, there are at most $M_0(p_1) + M_0(p_2) + 1$ reachable states in N, i.e. $M_0(p_1) + M_0(p_2) \geq |w| - 1$. Assume that for $n = M_0(p_1) + M_0(p_2)$, $n \geq |w|$ is satisfied. Then, markings $(\#_b(w) + k, \#_a(w) + \ell)$

with $k, \ell \geq 0$ must be unreachable as they allow firings of both transitions. The remaining possible markings

$$(0,n),\ \ldots,\ (\#_b(w){-}1, n{-}\#_b(w){+}1) \quad \text{and} \quad (n,0),\ \ldots,\ (n{-}\#_a(w){+}1, \#_a(w){-}1)$$

are exactly the $|w| = \#_a(w) + \#_b(w)$ markings reachable in N. Now, $(0,n)\ [b\rangle\ (\#_b(w), n - \#_b(w))$ would reach an unreachable marking, a contradiction. Thus, $M_0(p_1) + M_0(p_2) = |w| - 1$. □

The proposition allows us to define the initial marking for each place of the sought net in the case of synthesis from a cycle.

Algorithm 2 – `ABCycSolve` – for cyclic solving of a word w is in $O(|w|^2)$, since the outer **for**-loop runs through the all possible rotations of w, and the inner **for**-loop checks the ratio of the letters for all prefixes of the rotations.

Algorithm 2 ABCycSolve

Input: $w \in \{a,b\}^+$
Output: Petri net solving w^ω if it exists

```
compute W[0] ← #_a(w), W[1] ← #_b(w)
m_0 ← 0, m ← 0 {tokens available for a}
for i = 0 to |w| − 1 do {rotations of w}
  v ← w[i] ... w[|w| − 1]w[0] ... w[i − 1]
  if v[0] = 'a' then R ← 0 else R ← 1 {fraction selector}
  N[0] ← 0, N[1] ← 0 {for counting a's and b's}
  for j = 0 to |v| − 1 do {prefixes of v}
    if v[0] ≠ v[j] and N[R] * W[1 − R] ≤ W[R] * N[1 − R]
      then return {unsolvable}
    if v[j] = 'a' then N[0] ← N[0] + 1 else N[1] ← N[1] + 1
  endfor
  if v[0] = 'a' then m ← m − W[1] else m ← m + W[0] {fire}
  if m < 0 then m_0 ← m_0 − m, m ← 0
endfor
F(p_1, a) ← #_b(w), F(a, p_2) ← #_b(w), F(p_2, b) ← #_a(w), F(b, p_1) ← #_a(w)
return ({p_1, p_2}, {a, b}, F, {p_1 → m_0, p_2 → |w| − 1 − m_0})
```

So far we have presented the characterisation for the class of finite sequences and loops solvable with Petri nets. In the following section we shall combine these two shapes, concluding with a general theorem.

2.9 Special cyclic forms of synthesisable lts

A specific kind of transition systems havig the shape of a lasso, which we shall discuss in the next lemmata, leads to a general theorem characterising the synthesisability of a class of transition systems with at most two transitions and having uncomplicated shape, precisely those which consist of a finite prefix and a cyclic remainder.

Checking words in which the cyclic part contains only one letter, we have:

Lemma 5. [BESW16](solvable binary words of the form va^ω) *Let $v \in \{a,b\}^+$. The infinite word va^ω is solvable if and only if $v \in b^*a^*$.*

Proof. "$\Leftarrow$": If $\#_b(v) > 0$, the Petri net $N = (\{p_1, p_2\}, \{a, b\}, F, M_0)$ with $F(p_1, a) = F(a, p_1) = \#_b(v)$, $F(p_2, b) = 1 = F(b, p_1)$, $M_0(p_2) = \#_b(v)$, and $M_0(p_1) = 0$ solves va^ω. With $\#_b(v) = 0$, the Petri net $(\emptyset, \{a\}, \emptyset, \emptyset)$ is a solution.

"$\Rightarrow$": Assume $v \notin b^*a^*$, then a decomposition $va^\omega = u|_s a|_{s'} b|_{s''} a^\omega$ exists. A place q preventing a at s' exists with $M_{s'}(q) - M_s(q) = \mathbb{E}_q(a) < 0$. Thus, a cannot fire infinitely often at s''. □

A criterion for solvability of the words where the cyclic remainder contains both letters at least once is established in the next lemma.

Lemma 6. [BESW16](solvable binary words of the form vw^ω) *Let $v \in \{a,b\}^+$ and $w \in \{a,b\}^+ \backslash (a^+ \cup b^+)$. The infinite word vw^ω is solvable if and only if w is cyclically solvable and v is a postfix of w^i for some $i \geq 1$.*

Proof. "$\Rightarrow$": For arbitrary late parts of vw^ω (those which are contained by w^ω), Lemma 4 results in the same conditions as for w^ω, i.e. if vw^ω is solvable by $N = (\{p_1, p_2\}, \{a, b\}, F, M_0)$, so is w^ω (possibly with a different initial marking). W.l.o.g. let w be minimal cyclic solvable (otherwise rewrite vw^ω accordingly). If v is not a postfix of w^i (with i such that $|v| \leq |w^i|$), we find u, x, y with (w.l.o.g.) $v = xau$ and $w^i = ybu$, and $M_0[xa\rangle M[uyb\rangle M$ for some marking M. By the condition of the lemma, w contains an a and a b. W.l.o.g. p_1 receives tokens from b and delivers to a, and p_2 covers the other direction. Then,

$$M(p_1) + M(p_2) \geq F(b, p_1) + F(a, p_2) = \#_a(w) + \#_b(w) = |w|,$$

contradicting Proposition 11.

"$\Leftarrow$": If v is a postfix of w^i we can rewrite vw^ω as u^ω with u and w being rotations of each other, and thus having the same Petri net solving them by Theorem 4, differing only at the initial marking. □

So, words vw^ω with $w \in \{a,b\}^+ \backslash (a^+ \cup b^+)$ are solvable only if they can be rewritten as u^ω.

Summing up these lemmas, we obtain the following theorem for words that consist of a finite prefix and a cyclic remainder.

Theorem 5. [BESW16](solvable cyclic binary words with a prefix)
A word vw^ω with $v, w \in \{a,b\}^*$ is solvable if and only if $w = \epsilon$ or vw^ω can be rewritten as a cyclically solvable word u^ω or it has the form a^+b^ω or b^+a^ω.

Since we can encode both a finite sequence and a loop by the regular expression vw^ω, the theorem gives a characterisation for both of these classes, referring for the details to Theorems 3 and 4, respectively.

2.10 Synthesis of words by special classes of Petri nets

As we have already seen, there are various synthesis algorithms, allowing to construct a Petri net implementation from the given specification. In some application the implementation may be sought only in particular class of Petri nets, for instance safe (1-bounded) ones (e.g. in asynchronous circuits design [YK98]), or pure ones (e.g. in control theory [GRX02]). In such cases, taking into account that the synthesis procedure can be time-consuming, it may be of use to know in advance if the synthesis is possible, without initiating the synthesis itself. To have such a possibility one would need some necessary conditions for synthesisability with various classes of Petri nets, that can be checked on the given specification relatively quickly, i.e. the conditions need to be formulated graph-theoretically. In this section we will have a look at such possible conditions for various classes of Petri nets.

2.10.1 Synthesis with output-non-branching Petri nets

The property of output-non-branching (i.e. a place cannot have more than one output transitions) is of a big importance for the cases when one needs to eliminate the situation of conflict of different agents for the same resource. The next proposition demonstrates that in case of sequential behaviour of a Petri net with two transitions, we can always exclude such situations.

Proposition 12. *If $w \in \{a, b\}^*$ is solvable, it is solvable by an ON-net.*

Proof. Let $N = (P, T, F, M_0)$ be a net solving w. If N is output-non-branching then we are done. If N is not an ON-net, we then modify N in order to obtain an ON-net with the same reachability graph. For each place p (of a general form as in Fig. 2.29) which is not ON, depending on the effects $\mathbb{E}_a = \mathbb{E}(p)(a)$ and $\mathbb{E}_b = \mathbb{E}(p)(b)$ of transitions a and b for this place, respectively, we can apply the following transformation:

- $\mathbb{E}_a \geq 0,\ \mathbb{E}_b \geq 0$
  If w starts with a, p never separates a at any state s. Hence, we can modify p as follows: $m' = m$, $a'_- = 0$, $a'_+ = \mathbb{E}_a$, $b'_- = b_-$, $b'_+ = b_+$. If w starts with b then p never separates b at any state s. The modification of p in this case: $m' = m$, $a'_- = a_-$, $a'_+ = a_+$, $b'_- = 0$, $b'_+ = \mathbb{E}_b$.

- $\mathbb{E}_a < 0,\ \mathbb{E}_b < 0$
  In this case p can separate a and b only after their very last occurrences. Therefore, it can be substituted with two places p_a and p_b, such that $M_0(p_a) = \#_a(w)$, $F(p_a, a) = 1$, $F(p_a, b) = F(p_a, b) = F(p_a, b) = 0$ and $M_0(p_b) = \#_b(w)$, $F(p_b, b) = 1$, $F(p_b, b) = F(p_b, b) = F(p_b, a) = 0$, correspondingly.

- $\mathbb{E}_a \geq 0,\ \mathbb{E}_b < 0$ ($\mathbb{E}_a < 0,\ \mathbb{E}_b \geq 0$ is symmetrical)
  If a is initially disabled, it never occurs. Then, w starts with a, and p can only separate a after its very last occurrence. Then, modify p: $m' = m$, $a'_- = 0$, $a'_+ = \mathbb{E}_a$, $b'_- = b_-$, $b'_+ = b_+$; and add an additional place p_a: $M_0(p_a) = \#_a(w)$, $F(p_a, a) = 1$, $F(p_a, b) = F(p_a, b) = F(p_a, b) = 0$.

□

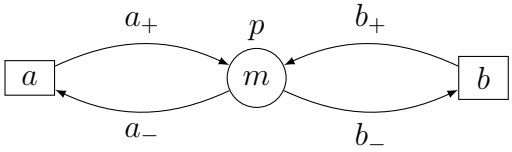


Figure 2.29: A place p with arc weights a_-, a_+, b_-, b_+ and initial marking m.

Proposition 12 establishes that for the class of linear transition systems over the binary set of labels solvability with ON-nets is equivalent to solvability in general. As we have seen in Section 2.7.2, this is not the case for the class of bounded Petri nets.

2.10.2 Synthesis with bounded Petri nets

The propositions below deal with boundedness of the sought Petri net, which is also often of interest in applications (see e.g. [vdA00], [MKMH86]).

Proposition 13. *If a finite sequence $w \in \{a, b\}^*$ has a factor u^{k+1}, where $u \in \{a, b\}^*$ and $k \in \mathbb{N}$, then w is not solvable with a k-bounded PN.*

Proof. By contradiction, assume, there is a k-bounded Petri net N that solves w. If the effect of u on some place p of N is $\mathbb{E}(p)(u) \neq 0$, then the marking of p after executing u^{k+1} is either $> k$ (in case $\mathbb{E}(p)(u) > 0$) or $< -k$ (in case of $\mathbb{E}(p)(u) < 0$), which contradicts the k-boundedness of p. If for every place p of N the effect of u on p is $\mathbb{E}(p)(u) = 0$, then N executes an infinite number of u in a row contradicting the finiteness of w. □

The condition from Proposition 13 can be used for the goal-oriented synthesis (when the solution is sought as a member of some special class of net, in particular k-bounded here) in order to detect quickly the unsolvability of the input lts (or sequence) without initiating the synthesis procedure itself. Another result of this kind is established in the next lemma.

Proposition 14. *Let $w \in \{a, b\}^*$. If $\#_a(w) \geq 3 \wedge \#_b(w) \geq 3$, then w is not 2-boundedly PN-solvable.*

Proof. Assume a contrary, i.e. there is 2-bounded net N that solves w. Then, for every place p of N, $\mathbb{E}(p)(a) \cdot \mathbb{E}(p)(b) < 0$. Indeed, if $\mathbb{E}(p)(a) = 0$, then, due

to $\#_b(w) \geq 3$ and 2-boundedness of N, $\mathbb{E}(p)(b) = 0$. Hence, p is useless. If $\mathbb{E}(p)(a) \cdot \mathbb{E}(p)(b) > 0$, then either both effects are positive or both are negative. This contradicts 2-boundedness of N, because of too many occurrences of both letters in w.

W.l.o.g. w ends with a. Let $P_1 = \{p \text{ is a place of } N \mid \mathbb{E}(p)(a) > 0\}$. If $\mathbb{E}(p)(a) = 2$ for all $p \in P_1$, b is enabled after w, which is a contradiction. Hence, there is $q \in P_1$ such that q contains 1 token after executing of w and $F(q, b) = 2$. This implies $F(a, q) = 1$ and $F(q, a) = 0$. Moreover, $F(b, q) = 0$, otherwise q contains 2 tokens after executing the last a. Thus, there are exactly two a's between different occurrences of b in w:

$$w = ...baabaab...$$

As w ends with a, it cannot start with a, because otherwise it has a subword $abaabaaba = (aba)^3$, i.e. it is not 2-boundedly solvable by Proposition 13. Hence, w starts with b. Since $baabaabaa = (baa)^3$, it cannot be solved with a 2-bounded PN. Therefore,

$$w = baabaaba$$

Consider an arbitrary place r in N such that $\mathbb{E}(r)(a) < 0$ (such r exists, since a is separated at the end of w). Due to the $\mathbb{E}(r)(a) \cdot \mathbb{E}(r)(b) < 0$, $\mathbb{E}(r)(b) > 0$. If $\mathbb{E}(r)(a) = -2$, then a is disabled after first ba. Then, $\mathbb{E}(r)(a) = -1$. For $\mathbb{E}(r)(b) = 1$, $baabaa$ is not executable with 2-bounded PN for any initial marking of r. Hence, $\mathbb{E}(r)(b) = 2$ and r initially has no tokens in it. Finally, r enables a after w. As r is arbitrary, we have a contradiction to 2-bounded PN-solvability of w. □

Since for each letter the number of its occurrences in a sequence can be found in a linear (in the length of the sequence) time, the condition from Proposition 14 suggests a linear procedure for a quick-fail check of the 2-bounded unsolvability of the input sequence, which is faster than the complexity of synthesis algorithms (e.g. `ABSolve`).

2.10.3 Synthesis with pure Petri nets

If a word $w = b^{x_1}a \dots ab^{x_n}$ can be solved, then side-places may be necessary to do it. For instance, *bbab* $|_s$ *bababab* is solvable (e.g. by net N_1 in the left of Fig. 2.30), but it cannot be solved side-place-freely. (More precisely: a side-place is needed in order to separate a at state s.) However, we will show that in the worst case, only some side-places q around a, preventing a at some state, are necessary. Also, such side-places are unnecessary if x_1 is small enough, in the sense that $x_1 \leq \min\{x_2, \dots, x_{n-1}\}$. For example, *babbababab* can be solved without any side-places (net N_2 in Fig. 2.30 is a possible pure solution). The 'smallness' of x_1 is sufficient but not necessary. For instance, *bbabbabab* has a side-place-free solution (e.g. net N_3 in the right of Fig. 2.30), even though $x_1 \not\leq \min\{x_2, \dots, x_{n-1}\}$.

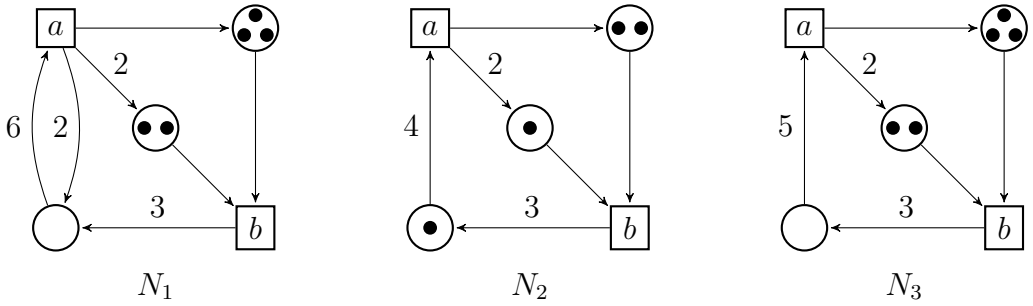


Figure 2.30: N_1 is a (non-pure) solution for *bbabbababab*; N_2 is a possible pure solution for *babbababab*; N_3 solves *bbabbabab* side-place-freely.

In the following, we assume w to be of the following form (2.7). The states s_i $(1 \leq i \leq n-1)$ denote the important states at which b has to be prevented, and the states r_k $(1 \leq k \leq n-1)$ denote the important states at which a has to be prevented. At or after the last group of b's, a can be prevented by a counting place, and at the final state, b can similarly be prevented by a counting place.

$$\boxed{w = b^{x_1-1} \,|_{r_1}\, b \,|_{s_1}\, a\, b^{x_2-1} \,|_{r_2}\, b \,|_{s_2}\, a \;\dots\; |_{s_{k-1}}\, a\, b^{x_k-1} \,|_{r_k}\, b \,|_{s_k}\, a \;\dots\; |_{s_{n-1}}\, a\, b^{x_n}} \quad (2.7)$$

Proposition 15. [BBE+16](side-place-free solvability with few initial b's) *If $w = b^{x_1}ab^{x_2}a \dots ab^{x_n}$ is solvable, then side-places are necessary, at worst, between a and q, where q is some place preventing a at one of the states r_k with $1 \leq k <$*

$n-1$. *If* $w = b^{x_1}ab^{x_2}a\dots ab^{x_n}$ *is solvable and* $x_1 \leq \min\{x_2,\dots,x_{n-1}\}$, *then* w *is solvable side-place-freely.*

Proof. The first claim follows from Lemmata 7 and 8 below. The second claim follows from Lemma 9. □

As usual, we utilise the general form of a place in Petri net with transitions a and b as in Fig. 2.31.

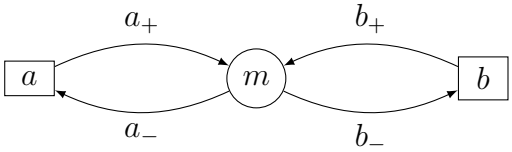


Figure 2.31: A place with arc weights a_-, a_+, b_-, b_+ and initial marking m.

Lemma 7. [BBE$^+$15](side-place-freeness around b**)** *If* $w = b^{x_1}a\dots ab^{x_n}$ *is solvable, then* w *is solvable without side-place around* b.

Proof. We show that side-places around b are necessary neither for preventing any b (cf. **(A)** below), nor for preventing any a (cf. **(B)** below).

(A): Suppose some place p prevents b at some state s_k, for $1 \leq k \leq n-1$. (The only other state at which b must be prevented is state s_n, but that can clearly be done by a non-side-place, e.g. by an incoming place of transition b that has $\#_b(w) = \sum_{i=1}^{n} x_i$ tokens initially.) Note that $b_- > b_+$, because place p allows b to be enabled at the state preceding s_k but not at s_k. Similarly, $a_- < a_+$, because b is not enabled at state s_k but at the immediately following state, which is reached after firing a. From the form (2.7) of w, we have

$$\begin{array}{lcl} b_+ & \leq & m + x_1(b_+ - b_-) \\ b_+ & \leq & m + (x_1+x_2)(b_+ - b_-) + (a_+ - a_-) \\ \dots & & \\ b_+ & \leq & m + (x_1+\dots+x_n)(b_+ - b_-) + (n-1)(a_+ - a_-) \\ 0 & \leq & -m - (x_1+\dots+x_k)(b_+ - b_-) - (k-1)(a_+ - a_-) + b_- - 1 \end{array} \tag{2.8}$$

The first n inequations assert the semipositivity of the marking of place p (more precisely, its boundedness from below by b_+, since p may be a side-place) at the n states $s_1, \ldots, s_n$. In our context, if these inequalities are fullfilled, then the marking is $\geq b_+$ at *all* states, as a consequence of $b_- \geq b_+$, $a_- \leq a_+$, and the special form of the word. The last inequality comes from $\neg(s_k[b\rangle)$.

We certainly have $0 \leq b_+ < b_- \leq m$, because of $b_- > b_+$ as noted above, and because b is initially enabled. If $b_+ = 0$, then p is not a side-place around b, and there is nothing more to prove (for p). If $b_+ \geq 1$, we consider the transformation

$$b'_+ = b_+ - 1 \text{ and } b'_- = b_- - 1 \text{ and } m' = m - 1$$

The relation $0 \leq b'_+ < b'_- \leq m'$ still holds for the new values. Also, all inequalities in (2.8) remain true for the new values: in the first n lines, 1 is subtracted on each side, and on the last line, the increase in $-m$ is offset by the decrease in b_-.

We have thus shown that subtracting one arc from b to p, one arc from p to b, and removing one initial token from p, leaves the region inequalities invariant. Thus, we get a solution preventing b with a 'smaller' side-place, and we can continue until eventually b_+ becomes zero. This finishes part **(A)** of the proof.

(B): A side-place around b might still be necessary to prevent a at some state. We show next that such side-places are also unnecessary. Suppose some place q as in Fig. 2.31 prevents a at state r_k, for $1 \leq k \leq n-1$. Symmetrically to the previous case, we have $b_+ > b_-$. This is true because, while q does not have enough tokens to enable a at state r_k, it must have enough tokens to enable a at the directly following state (which we may continue to call s_k). But we also have (w.l.o.g.) $a_+ < a_-$. For $k \geq 2$, this follows from the fact that if the previous a (enabled at the state s_{k-1} just after r_{k-1}) acts positively on q, then q also has sufficiently many tokens to enable a at state r_k. For $k = 1$, it is possible to argue that $a_+ < a_-$ is valid without loss of generality. For suppose that q disables a only at r_1 and nowhere else. (This is no loss of generality because for the other states r_k, $k \geq 2$, copies of q can be used.) Then we may consider q' which is an exact copy of q, except that $a_+ = a_- - 1$ for q'. This place q' also disables a at state r_1 (because it has the same marking as q). Moreover, it does not disable a at any other state after r_1 because it always has $\geq a_- - 1$ tokens, and after the next b, $\geq a_-$ tokens, since $b_+ > b_-$.

Because of $b_+ > b_-$ and $a_+ < a_-$, place q also prevents a at all prior states in the same group of b's. Moreover, in the last (i.e. nth) group of b's, a can easily be prevented side-place-freely. For place q with initial marking m, we have

$$\begin{array}{lcl} a_+ & \leq & m + x_1(b_+ - b_-) + (a_+ - a_-) \\ a_+ & \leq & m + (x_1 + x_2)(b_+ - b_-) + 2(a_+ - a_-) \\ \ldots & & \\ a_+ & \leq & m + (x_1 + \ldots + x_{n-1})(b_+ - b_-) + (n-1)(a_+ - a_-) \\ 0 & \leq & -m - (x_1 + \ldots + x_k - 1)(b_+ - b_-) - (k-1)(a_+ - a_-) + a_- - 1 \end{array} \tag{2.9}$$

The first $n-1$ inequations assert the semipositivity of the marking of place q (more precisely, its boundedness from below by a_+, since q may be a side-place of a) at the $n-1$ states just after the a's in (2.7). If they are fullfilled, then the marking is $\geq a_+$ at *all* states after the first a, as a consequence of $b_+ > b_-$ and the special form of the word. The last inequality asserts that place q prevents transition a at state r_k, hence effects the event-state separation of a at r_k.

If b_- is already zero, place q is not a side-place of b. Otherwise, we may perform the transformation

$$b'_+ = b_+ - 1 \text{ and } b'_- = b_- - 1 \text{ and } m' = m$$

because of $b_+ > b_-$ as noted above. The left-hand sides of the first $n-1$ inequalities in (2.9) do not decrease, and neither do the right-hand sides. The same is true for the last inequality. This finishes part **(B)** of the proof. □

Lemma 8. [BBE+15](side-place-freeness around a, preventing b)
Suppose $w = b^{x_1}ab^{x_2}a \ldots ab^{x_n}$. If w is solvable by a net in which some place p separates b, then we may w.l.o.g. assume that p is not a side-place around a.

Proof. The equation system (2.8) is invariant under the transformation

$$a'_+ = a_+ - 1 \text{ and } a'_- = a_- - 1 \text{ and } m' = m$$

as neither left-hand sides nor right-hand sides change their values. □

Lemma 9. [BBE+15](side-place-freeness around a, preventing a)
Suppose $w = b^{x_1}ab^{x_2}a \ldots ab^{x_n}$. If $x_1 \leq \min\{x_2, \ldots, x_{n-1}\}$ and if w is solvable by a

net in which some place q prevents transition a at state r_k with $1 \leq k \leq n$, then we may w.l.o.g. assume that q is not a side-place around a.

Proof. For preventing a at state r_n, we only need a place with no input and a single output transition a (weight 1) which has $n-1$ tokens initially.

Suppose q prevents a at state r_k, with $1 \leq k \leq n-1$. From previous considerations, we know $a_+ \leq a_-$ and $b_+ > b_-$, and we may assume, from Lemma 7, that q is not a side-place around b, i.e., that $b_- = 0$. The initial marking m of q and the remaining arc weights a_+, a_-, b_+ satisfy the following system of inequations (which is the same as (2.9), except that it is simplified by $b_- = 0$):

$$\begin{array}{lcl} a_+ & \leq & m + x_1(b_+) + (a_+ - a_-) \\ a_+ & \leq & m + (x_1 + x_2)(b_+) + 2(a_+ - a_-) \\ \cdots & & \\ a_+ & \leq & m + (x_1 + \ldots + x_{n-1})(b_+) + (n-1)(a_+ - a_-) \\ 0 & \leq & -m - (x_1 + \ldots + x_k - 1)(b_+) - (k-1)(a_+ - a_-) + a_- - 1 \end{array} \tag{2.10}$$

If $a_+ = 0$, then q is already of the required form. For $a_+ > 0$, we have two cases.

Case 1: $m > 0$ and $a_+ > 0$. Then consider the transformation

$$m' = m - 1 \text{ and } a'_+ = a_+ - 1 \text{ and } a'_- = a_- - 1$$

By $m > 0$ and $a_- \geq a_+ > 0$, we get new values $m', a'_+, a'_- \geq 0$. Moreover, (2.10) remains invariant under this transformation. So, q' serves the same purpose as q, and it has one incoming arc from a less than q. By repeating this procedure, we either get a place which serves the same purpose as q, or we hit Case 2.

Case 2: $m = 0$ and $a_+ > 0$. In this case, we consider the transformation

$$m' = m = 0 \text{ and } a'_+ = 0 \text{ and } a'_- = a_-$$

Such a transformation also guarantees $m', a'_+, a'_- \geq 0$. Also, the last line of (2.10) is clearly satisfied with these new values, since the value of its right-hand stays the same (for $k = 1$) or increases (for $k > 1$). To see that the first $n-1$ lines of (2.10) are also true with the new values, and that we can, therefore, replace q by q', we may argue as follows. At any marking $\widetilde{m}$ reached along the execution of w,

we have the following:

$$\widetilde{m}(q) \;\geq\; \widetilde{m}(q') \;\geq\; 0 \tag{2.11}$$

These inequalities imply that the new place q' prevents a at r_k, whenever the old one, q, does, and that, moreover, no occurrences of a are excluded by the place q' where they should not be prohibited.

The first of the inequalities (2.11) holds because it holds initially (when $\widetilde{m} = m$, then $\widetilde{m}(q) = m = m' = \widetilde{m}(q')$), and because the effect of a before the transformation is $(a_+ - a_-)$, and after the transformation, it is $(-a_-)$. In other words, a reduces the token count on q' more than it does so on q, while b has the same effect on q' as on q. To see the second inequality in (2.11), let $x = \min\{x_2, \ldots, x_{n-1}\}$. Then

$$a_- \;\leq\; x_1 \cdot b_+ \;\leq\; x \cdot b_+$$

The first inequality follows because $m = 0$ and q has enough tokens after the first x_1 occurrences of b in order to enable a. The second inequality follows from $x_1 \leq x$. But then, since a only removes a_- tokens from q' and the subsequent block of b's puts at least $x \cdot b_+$ tokens back on q', the marking on q' is always ≥ 0, up to and including the last block of b's. □

Solving a word of the form $w = b^{x_1}a \ldots ab^{x_n}$ side-place-freely allows us to draw some conclusion about prepending a letter a to it. In fact, we have:

Proposition 16. [BBE$^+$16](side-place-free solvability of $b^{x_1}ab^{x_2}a \ldots ab^{x_n}$) *Sequence $w = b^{x_1}ab^{x_2}a \ldots ab^{x_n}$ is solvable side-place-freely if and only if aw is solvable.*

Proof. Lemmata 10 and 11 for ($\Rightarrow$), and Lemma 12 for ($\Leftarrow$). □

Lemma 10. [BBE$^+$16](preventing a in aw) *Suppose $w = b^{x_1}ab^{x_2}a \ldots ab^{x_n}$ is solvable side-place-freely. Then in aw, all occurrences of a can be separated side-place-freely.*

Proof. Because a can be prevented side-place-freely in w at any state r_k, the system (2.9) has a solution with $a_+ = 0$ and $b_- = 0$ for any fixed $1 \leq k \leq n-1$. This

refers to a pure input place q of a, which may or may not be an output place of b. In order to prevent a in aw side-place-freely, we need to consider the states r_k as before (but shifted to the right by one index position, still just before the last b of the kth group of b's) and a correspondingly modified system as follows:

$$\boxed{\begin{array}{rcl} 0 & \leq & m' + (x_1 + \ldots + x_i) \cdot (b'_+) + (i+1) \cdot (-a'_-) \quad \text{for all } 0 \leq i \leq n-1 \\ 0 & \leq & -m' - (x_1 + \ldots + x_k - 1) \cdot (b'_+) - k \cdot (-a'_-) + a'_- - 1 \end{array}} \tag{2.12}$$

where m', b'_+ and a'_- refer to a new pure place q' preventing a at state r_k in aw. The line with $i = 0$ was added because a must be enabled initially. Consider the transformation

$$m' = m + a_- \text{ and } b'_+ = b_+ \text{ and } a'_- = a_-$$

These values satisfy (2.12), provided m, b_+ and a_- (together with $a_+ = 0$ and $b_- = 0$) satisfy (2.9). The line with $i = 0$ follows from $m' = m + a_- \geq 0$. The other lines corresponding to $i \geq 1$ reduce to the corresponding lines in (2.9), since the additional $(-a_-)$ at the end of each line is offset by the additional $(+a_-)$ at the beginning of the line. The last line (which belongs to state r_k at which a is separated) corresponds to the last line of (2.9), because the decrease by a_- at the beginning of the line is offset by an increase by a_- in the term $k \cdot (-a'_-)$ (compared with $(k-1) \cdot (-a_-)$ as in (2.9)). □

Let p' be a general new place which is supposed to prevent b at state s_k in aw. In order to check the general solvability of aw if w is side-place-freely solvable, we consider a general transformation

$$m' = m + \mu \;\;,\;\; b'_+ = b_+ + \beta_+ \;\;,\;\; b'_- = b_- + \beta_- \;\;,\;\; a'_+ = a_+ + \alpha_+ \;\;,\;\; a'_- = a_- + \alpha_-$$

where $\mu \geq -m$, $\beta_+ \geq -b_+$, $\beta_- \geq -b_-$, $\alpha_+ \geq -a_+$ and $\alpha_- \geq -a_-$, as well as a new inequality system:

$$\boxed{\begin{array}{rcl} b'_+ & \leq & m' + (x_1 + \ldots + x_i) \cdot (b'_+ - b'_-) + i \cdot (a'_+ - a'_-) \quad \text{for } 1 \leq i \leq n \\ 0 & \leq & -m' - (x_1 + \ldots + x_k) \cdot (b'_+ - b'_-) - k \cdot (a'_+ - a'_-) + b'_- - 1 \end{array}}$$

This system has to be compared with a restricted form of (2.8) (setting $b_+ = a_- = 0$, since the solution of w is pure). Doing this by line-wise comparison, we get the following inequality system for the new value differences:

$$\boxed{\begin{array}{lcl} \mu & \geq & -m,\ \beta_+ \geq -b_+,\ \beta_- \geq -b_-,\ \alpha_+ \geq -a_+,\ \alpha_- \geq -a_- \\ \beta_+ & \leq & \mu + (x_1 + \ldots + x_i) \cdot (\beta_+ - \beta_-) + i \cdot (\alpha_+ - \alpha_-) + a_+ \\ 0 & \leq & -\mu - (x_1 + \ldots + x_k) \cdot (\beta_+ - \beta_-) - k \cdot (\alpha_+ - \alpha_-) - a_+ + \beta_- \end{array}} \quad (2.13)$$

The lines with i must be solved simultaneously for every $1 \leq i \leq n$ while the line with k must be solved individually for every $1 \leq k \leq n-1$, in order to get a place preventing b at state s_k. This leads to the following lemma.

Lemma 11. [BBE$^+$16](solving aw from w) *Suppose $w = b^{x_1}ab^{x_2}a \ldots ab^{x_n}$ is solvable side-place-freely. Then aw is solvable.*

Proof. Suppose that a pure place p with parameters b_- (arc into b), a_+ (arc from a) and m (initial marking) is given and suppose it separates b from s_k in w. This place solves (2.8) for that particular k. We distinguish two cases:

Case 1: $a_+ \leq m$. In this case, the place p can essentially be re-used for the same purpose in the solution (that we construct in this way) for aw, since (2.13) is solved by putting

$$\mu = -a_+ \ ,\ \beta_+ = \beta_- = 0 \ ,\ \alpha_+ = \alpha_- = 0$$

Hence, a place p' which differs from p only by its initial marking ($m' = m - a_+$ instead of m) separates b at s_k in aw.

Case 2: $a_+ > m$. In this case, (2.13) can be solved by

$$\mu = -m \ ,\ \beta_+ = \beta_- = a_+ - m \ ,\ \alpha_+ = \alpha_- = 0$$

That is, we may replace p by a place p' with zero initial marking and adding uniformly the value $a_+ - m$ to the incoming and outgoing arcs of b, creating a side-place around b. □

Lemma 12. [BBE$^+$16](solving w side-place-freely from aw)
If w=$b^{x_1}ab^{x_2}a \ldots ab^{x_n}$ and aw can be solved, then w has a side-place-free solution.

Proof. Suppose that aw has a solution in which some place q', preventing a, is a side-place around a. Because q' prevents a, $a'_- > a'_+$ (unless it is the first a, but then we don't need q' in solving w). Because a is enabled initially, $m' \geq a'_-$. But then, the transformation $a''_- = a'_- - a'_+$, $a''_+ = 0$, $m'' = m' - a'_+$ yields another place q'' which is not a side-place around a but serves the same purpose as q'. The rest of the proof follows because the above transformations (removing side-places around b, or side-places around a which prevent b) do not introduce any new side-places around a. □

As the main result of this section, Proposition 16 establishes the characterisation for pure Petri net solvability of sequences of a special form $w = b^{x_1}ab^{x_2}a \ldots ab^{x_n}$ which relies on the (general) solvability of the sequence aw.

2.11 Summary

The question of characterising the solvability of labelled transition systems by Petri nets has been considered in the present chapter. More precisely, we have focused on the class of binary linear and cyclic transition systems which can be represented in the form of finite sequences. As the main results of our investigations, we can recall the letter-counting criterion (first appeared in [BBE+15] and [BBE+16], sufficiency proved in [BESW16]) and the synthesis algorithm `ABSolve` [BESW16] based on the criterion, which demonstrates better run times than the classical region-based approach, and which can be modified for synthesis of only bounded Petri net solution when the one exists. The criterion and the algorithm, initially obtained for linear transition systems, were generalised for cyclically solvable lts [BESW16] Besides the solvability of an lts in general, we discussed the problem of the goal-oriented synthesis, i.e. the case when the solution is required in the form of Petri net of some special class. Precisely, the classes of output-non-branching, bounded and pure [BBE+16] Petri nets were considered. The solvability of finite sequences (in case of a binary alphabet) with a special classes of nets can be equivalent to the solvability in general (as in the case of output-non-branching Petri nets, see Proposition 12), can rely on the solvability of a longer sequence containing the initial one (as for pure nets [BBE+15]), or can be not directly implied by the general solvability (as for bounded Petri nets). It was also shown that besides the

synthesis algorithms that can be derived from a graph-theoretical (or language-theoretical for sequences) characterisation of the solvability, one can hope to gain some methods for a quick-fail check if the synthesis is possible without initiating the synthesis procedure itself. In particular, we have seen that the presence of a subword which satisfies the factor $(ab\alpha)b^*(ba\alpha)^+a$ [BBE+15] in a given sequence implies the unsolvability of the sequence in general. And the presence of a repeating factor can yield the unsolvability of the sequence with a bounded Petri net (see Proposition 13).

Chapter 3

Characterisation of minimal unsolvable words

3.1 Minimality of non-synthesisable binary words

Throughout the previous chapter we have investigated graph-theoretical conditions for finite sequences to be (un-)solvable with Petri nets. We have also mentioned that sometimes it makes sense to run some kind of pre-synthesis check in order to detect whether the synthesis is possible (can be finished successfully). Among other things, the notion of a minimal unsolvable word has been introduced [BBE+15]. This notion plays a crucial role in the current chapter which is base on works [EBMP16, BMP+16] by the author and co-authors. Here we aim to provide a complete characterisation for the class of minimal unsolvable words over a binary alphabet. As the use of such a result we see the possibility to develop a technique for a fast pre-synthesis check of the unsolvability of the input sequence, without initiating the synthesis itself. Since the unsolvability of a (proper) subword implies the unsolvability of the whole word, so as to reach our aim, we will start with attentive investigation of the graph-theoretical characteristics of muws and construct their classification.

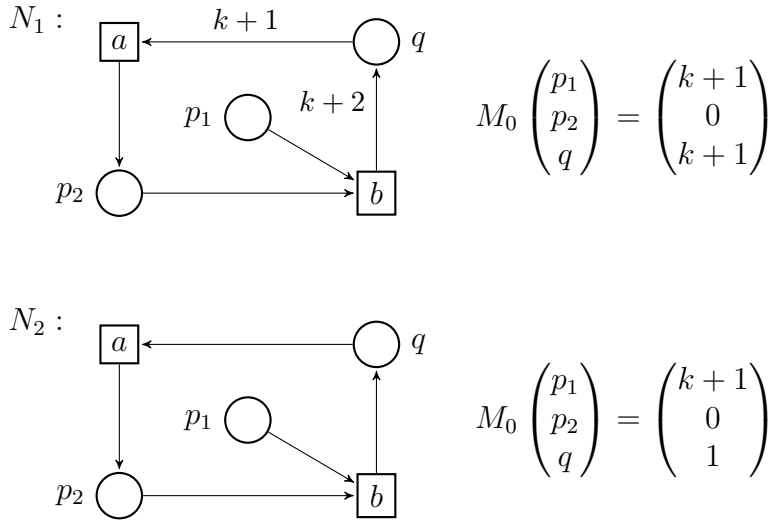


Figure 3.1: N_1 solves $ab(ab)^k aa$, N_2 solves $ab(ab)^k a$, for $k \geq 0$.

3.2 A classification of binary muws by shape

Let w be a minimal unsolvable binary word starting with a. By Proposition 2 we have two possible cases: either w ends with a single a, or it has many (more than one) a's at the end. So far we know

$$w = abua.$$

Proposition 6 implies that bu either does not contain the infix aa or does not contain the infix bb or both. Assume that bu has neither factors aa nor factors bb inside. Then the following two cases for a muw w are possible, depending on if u ends with a or with b:

$$ab(ab)^k aa \quad \text{or} \quad ab(ab)^k a, \quad \text{where } k \geq 0.$$

Petri nets N_1 and N_2 in Fig. 3.1 solve the first and the second of these forms, respectively. From Proposition 6 and this observation, we deduce that in a minimal unsolvable word of the form $a\alpha a$, α has either the factor aa or the factor bb, but never both.

Thus, w has one of the following forms, where $x_i > 0$ for $1 \leq i \leq n$ and $x_i > 1$ for some i:

1. $ab^{x_1}ab^{x_2}a \ldots ab^{x_n}a$: starts and ends with a, single a at the end, no aa;
2. $aba^{x_1}ba^{x_2}b \ldots ba^{x_n}ba$: starts and ends with a, single a at the end, no bb;
3. $ab^{x_1}ab^{x_2}a \ldots ab^{x_n}aa$: starts and ends with a, many a's at the end, no aa inside;
4. $aba^{x_1}ba^{x_2}b \ldots ba^{x_n}a$: starts and ends with a, many a's at the end, no bb.

All those patterns can be comprised into the following three general forms of a muw w :

$$ab^{x_1}ab^{x_2}a \ldots ab^{x_n}a \quad \text{with } x_i > 0 \text{ for } 1 \leq i \leq n \tag{3.1}$$

$$bab^{x_2}ab^{x_3}a \ldots ab^{x_n} \quad \text{with } x_i > 0 \text{ for } 2 \leq i \leq n \tag{3.2}$$

$$ab^{x_1}ab^{x_2}a \ldots ab^{x_n}aa \quad \text{with } x_i > 0 \text{ for } 1 \leq i \leq n \tag{3.3}$$

Form (3.1) represents item 1, form (3.2) represents items 2 and 4 (modulo swapping a/b and with bb appearing inside), and form (3.3) represents item 3. In the rest of this section we will analyse these forms more precisely.

Consider first some muw w of the form (3.1). In case of $n = 1$ we have

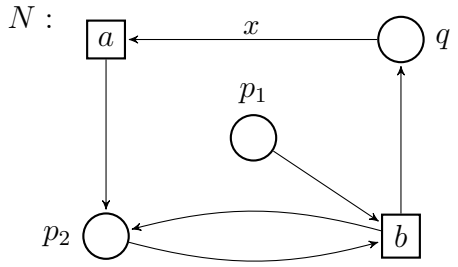
$$ab^x a, \quad \text{where} \quad x \geq 2.$$

The words of this form can be solved by Petri nets as in Fig. 3.2. Hence, the words corresponding to pattern (3.1) are solvable for $n = 1$.

Consider the special instance, $n = 2$, of pattern (3.1). The words of the following two classes

$$ab^{x+1}ab^x a \quad \text{or} \quad ab^{x-k}ab^x a \quad \text{with } 0 \leq k < x,$$

are solvable, and Petri nets N_1 and N_2 in Fig. 3.3 are possible solutions for words of the first and of the second of these forms, respectively. Thus, if $w = ab^{x_1}ab^{x_2}a$



$$M_0 \begin{pmatrix} p_1 \\ p_2 \\ q \end{pmatrix} = \begin{pmatrix} x \\ 0 \\ x \end{pmatrix}$$

Figure 3.2: N solves $ab^x a$.

is minimal unsolvable, then $x_1 - x_2 \geq 2$.

Consider an arbitrary minimal unsolvable word $w = ab^{x_1}ab^{x_2}a \ldots ab^{x_n}a$ of the form (3.1) with $n \geq 3$, $x_i > 0$ for $1 \leq i \leq n$. Let $x = \min\{x_i \mid 2 \leq i \leq n\}$. Due to Proposition 8, $x_i \in \{x, x+1\}$ for $2 \leq i \leq n$, and then $x_1 \leq x+2$. If $x_1 < x+1$, by Lemmata 9 and 11, the word w is solvable, contradicting the choice of w as a muw. Hence, $x+1 \leq x_1 \leq x+2$, and $\min\{x_i \mid 1 \leq i \leq n\} = x$. We now show $x_n = x$. Two cases are possible:
Case 1: $x_1 = x+2$. If $x_n = x+1$, then $x_j = x$ for some $1 < j < n$, which, due to Proposition 8, contradicts the minimality of w. Hence, $x_n = x$, and w follows the pattern $ab^{x+2}a(b^{x+1}a)^+b^x a$.
Case 2: $x_1 = x+1$. By contraposition, assume $x_n = x+1$. Then, $x_j = x$ for some $2 \leq j \leq n-1$. Let $j_1 = \max\{j \mid x_j = x\}$. Assume a is not separated from some state s_k in w.

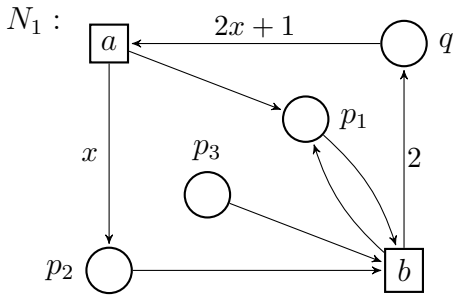
If $k < j_1$, then, by Lemma 4, for

$$w \;=\; \underbrace{a\, b^{x_1}\, a \ldots a\, b^{x_k - 1}}_{\alpha} |_{s_k} \overbrace{\underbrace{b\, a \ldots\, b^{x_{j_1}}}_{\beta}\, a \ldots\, b^{x_n}}^{\beta'}\, a$$

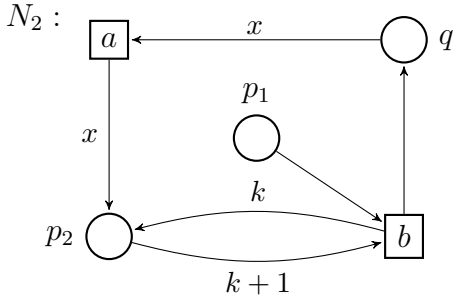
we have

$$\#_a(\beta) \cdot \#_b(\alpha) \geq \#_a(\alpha) \cdot \#_b(\beta).$$

Since $\#_a(\alpha) \neq 0$ by the form of w, and $\#_a(\beta) \neq 0$ due to $j_1 \leq n-1$, this inequality



$$M_0 \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ q \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 2x+1 \\ 2x+1 \end{pmatrix}$$



$$M_0 \begin{pmatrix} p_1 \\ p_2 \\ q \end{pmatrix} = \begin{pmatrix} 2x-k \\ 0 \\ x+k \end{pmatrix}$$

Figure 3.3: N_1 solves $ab^{x+1}ab^xa$, N_2 solves $ab^{x-k}ab^xa$.

is equivalent to

$$\frac{\#_b(\alpha)}{\#_a(\alpha)} \geq \frac{\#_b(\beta)}{\#_a(\beta)}.$$

From the choice of j_1 we have

$$\frac{\#_b(\beta)}{\#_a(\beta)} \geq \frac{\#_b(\beta')}{\#_a(\beta')},$$

implying

$$\frac{\#_b(\alpha)}{\#_a(\alpha)} \geq \frac{\#_b(\beta)}{\#_a(\beta)} \geq \frac{\#_b(\beta')}{\#_a(\beta')}.$$

The left and the right part of the last inequality can be rewritten as

$$\#_a(\beta') \cdot \#_b(\alpha) \geq \#_a(\alpha) \cdot \#_b(\beta').$$

According to Proposition 5, this entails the unsolvability of the proper subword $\alpha\beta' a$ of w, which contradicts the minimality of w.

Assume now that $k \geq j_1$. Then in

$$w \;=\; \underbrace{a\, b^{x_1}\, a\, \ldots a\, b^{x_{j_1}}\, a\, \ldots\, b^{x_k-1}}_{\alpha} |_{s_k}\; \underbrace{b\, a\, \ldots\, b^{x_n}}_{\beta}\; a,$$

by Lemma 4, we have

$$\#_a(\beta) \cdot \#_b(\alpha) \geq \#_a(\alpha) \cdot \#_b(\beta).$$

By the form of w, we have $\#_a(\alpha) \neq 0$. Transition a can be disabled from firing in the block b^{x_n} with a place p having $\#_b(w) \cdot n$ tokens on it initially, the weight of the arc from p to a is $\#_b(w)$, and the weight of the arc from b to p is 1. Hence, $\#_a(\beta) \neq 0$, and the inequality can be rewritten as

$$\frac{\#_b(\alpha)}{\#_a(\alpha)} \geq \frac{\#_b(\beta)}{\#_a(\beta)}.$$

On the other hand, thanks to the choice of x_{j_1}, we have

$$x+1 > \frac{\#_b(\alpha)}{\#_a(\alpha)} \quad \text{and} \quad \frac{\#_b(\beta)}{\#_a(\beta)} > x+1.$$

Together with the previous inequality this yields

$$x+1 > x+1,$$

which is a contradiction. Hence, $x_n = x$.

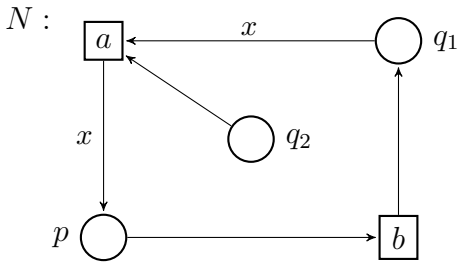
From the consideration above, we can deduce that all minimal unsolvable words of the form (3.1) match one of the following three refined patterns where $x > 0$:

$$\boxed{\begin{array}{l} ab^{x+k}ab^xa, \text{ with } k>2 \quad \text{or} \\ ab^{x+2}(ab^{x+1})^*ab^xa \quad \text{or} \\ ab^{x_1}ab^{x_2}a\ldots ab^{x_n}a, \text{ with } x_1 = x+1,\ x_n = x,\ x_i \in \{x, x+1\},\ n \geq 3 \end{array}} \tag{3.1′}$$

Let us now study pattern (3.2). In case of $n = 2$ we have

$$bab^x, \text{ where } x > 1.$$

The words of this form can be solved by Petri nets as in Fig. 3.4.



$$M_0 \begin{pmatrix} p \\ q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} 1 \\ x-1 \\ 1 \end{pmatrix}$$

Figure 3.4: N solves bab^x.

We now consider an arbitrary minimal unsolvable word

$$w = bab^{x_2}ab^{x_3}a\ldots ab^{x_n}$$

with $n \geq 3$ and $x_i > 0$ for $2 \leq i \leq n$ of the form (3.2). Let $x = \min\{x_i \mid 2 \leq i \leq n-1\}$. Due to Proposition 8, $x_i \in \{x, x+1\}$ for all $2 \leq i \leq n-1$, and then $x_n \leq x+2$. Assume $x_n \leq x$. Consider state s in w

$$w = \underbrace{b\, a\, b^{x_2}\, a \ldots\, a\, b^{x_k}}_{\alpha} \,|_s\, \overbrace{\underbrace{a\ \ldots\ b^{x_{n-1}-1}}_{\beta'}\ b\, a\, b^{x_n-1}}^{\beta}\, b,$$

from which b is not separated. Transition b can be separated from the state right after the first b with a place p having an arc from a to p with the weight equal to $\max\{x_i \mid 2 \leq i \leq n\}$, an arc from p to b with weight 1, and initially 1 token on it. Hence, $k \neq 1$. Transition b can easily be separated at the very end of w by an input place p of b, having $\#_b(w)$ tokens on p initially. Hence, $k \neq n$. If $k = n-1$, we have

$$\#_a(\alpha) \cdot \#_b(\beta) = (n-2) \cdot (x_n - 1)$$
$$\#_a(\beta) \cdot \#_b(\alpha) = 1 \cdot (1 + x_2 + \ldots + x_{n-1}).$$

The assumption that $x_n \leq x$ implies

$$\#_a(\alpha) \cdot \#_b(\beta) \leq (n-1) \cdot (x-1).$$

On the other hand, since $x_i \in \{x, x+1\}$ for all $2 \leq i \leq n-1$, then

$$\#_a(\beta) \cdot \#_b(\alpha) \geq 1 + (n-1) \cdot x.$$

Thus,

$$\#_a(\alpha) \cdot \#_b(\beta) < \#_a(\beta) \cdot \#_b(\alpha),$$

which, due to the minimal unsolvability of w, contradicts Lemma 4. Hence, $k < n-1$. From Lemma 4, because of the minimal unsolvability of w, we have

$$\#_a(\alpha) \cdot \#_b(\beta) \geq \#_a(\beta) \cdot \#_b(\alpha)$$

which is equivalent to

$$\frac{\#_b(\beta)}{\#_a(\beta)} \geq \frac{\#_b(\alpha)}{\#_a(\alpha)},$$

where $\#_a(\beta) \neq 0$ because of $k < n-1$, and $\#_a(\alpha) \neq 0$ due to $k > 1$. Since we assumed $x_n \leq x$,

$$\frac{\#_b(\beta')}{\#_a(\beta')} \geq \frac{\#_b(\beta)}{\#_a(\beta)}.$$

The last inequality can be rewritten as

$$\#_a(\alpha) \cdot \#_b(\beta') \geq \#_a(\beta') \cdot \#_b(\alpha),$$

which implies, due to Lemma 4, that $\alpha\beta' b$ is not solvable. Since it is a proper subword of w, we get a contradiction to the minimality of w. Thus $x+1 \leq x_n \leq x+2$.

We now demonstrate $x_2 = x$. Consider two possible cases:
Case 1: $x_n = x+2$. Take $j = \max\{i \mid x_i = x\}$. Then for the subword

$$u \;=\; \underbrace{b\,a\,b^{x_j}\,(a\,b^{x+1})^k}_{\alpha} \;|_s\; \underbrace{a\,b^{x_n-1}}_{\beta}\;b$$

of w with $k \geq 0$, the following inequality is satisfied:

$$\#_b(\beta) \cdot \#_a(\alpha) = (x+1)\cdot(k+1) \geq (1 + x + (x+1)\cdot k)\cdot 1 = \#_b(\alpha)\cdot\#_a(\beta).$$

This means that u is unsolvable. If $j > 2$, u is a proper subword of w, contradicting the minimality of w. Hence, in this case, $x_2 = x$ and $x_i = x+1$ for $2 < i < n$.
Case 2: $x_n = x+1$. Let $j_1 = \min\{i \mid x_i = x\}$. By the definition of x, $j_1 \neq n$. Assume $x_2 = x+1$. Consider a state s_k in w after the group b^{x_k}, such that b is not separated at s_k. If $k > j_1$, then, by Lemma 4, for

$$w \;=\; \underbrace{b\,a\,b^{x_2}\,a\,b^{x_3-1}\,\overbrace{b\,a\;\ldots\;b^{x_{j_1}}\;a\;\ldots\;b^{x_k}}^{\alpha'}}_{\alpha} \;|_{s_k}\; \underbrace{a\;\ldots\;b^{x_n-1}}_{\beta}\;b$$

the following inequality holds

$$\#_b(\beta) \cdot \#_a(\alpha) \geq \#_a(\beta) \cdot \#_b(\alpha).$$

This is equivalent to

$$\frac{\#_b(\beta)}{\#_a(\beta)} \geq \frac{\#_b(\alpha)}{\#_a(\alpha)},$$

where $\#_a(\beta) \neq 0$ by the choice of s_k, $\#_a(\alpha) \neq 0$ due to the fact that b can be separated from the state after the first b. As $x_2 = x + 1$ and $x_{j_1} = x$, we have

$$\frac{\#_b(\alpha)}{\#_a(\alpha)} \geq \frac{\#_b(\alpha')}{\#_a(\alpha')}.$$

Together with the previous inequality, we get

$$\frac{\#_b(\beta)}{\#_a(\beta)} \geq \frac{\#_b(\alpha')}{\#_a(\alpha')}$$

which is equivalent to

$$\#_b(\beta) \cdot \#_a(\alpha') \geq \#_a(\beta) \cdot \#_b(\alpha').$$

From this, according to Proposition 5, it follows that the proper subword $\alpha'\beta b$ of w is unsolvable, contradicting the minimality of w. Suppose that $k \leq j_1$. Then, by Lemma 4, for

$$w = \underbrace{b\, a\, b^{x_2}\, a\, \ldots\, b^{x_k}}_{\alpha} \,|_{s_k}\, \underbrace{a\, \ldots\, b^{x_{j_1}-1}\, b\, a\, \ldots\, b^{x_n-1}}_{\beta}\, b$$

the following inequality is satisfied

$$\#_b(\beta) \cdot \#_a(\alpha) \geq \#_a(\alpha) \cdot \#_b(\beta).$$

Since $\#_a(\beta) \neq 0$, thanks to the special form of the word, and $\#_a(\alpha) \neq 0$ due to

$k < n$, this inequality can be written as

$$\frac{\#_b(\beta)}{\#_a(\beta)} \geq \frac{\#_b(\alpha)}{\#_a(\alpha)}.$$

On the other hand, due to $x_n = x + 1$ and by the choice of j_1, we have

$$x + 1 > \frac{\#_b(\beta)}{\#_a(\beta)} \quad \text{and} \quad \frac{\#_b(\alpha)}{\#_a(\alpha)} \geq x + 1.$$

Hence, due to the previous inequality,

$$x + 1 > x + 1,$$

which is a contradiction. Thus, $x_2 = x$. From this we deduce the following refinement of pattern (3.2), where $x > 0$:

$$\boxed{\begin{array}{l} bab^x(ab^{x+1})^*ab^{x+2} \quad \text{or} \\ bab^{x_2}ab^{x_3}a \ldots ab^{x_n}, \text{ with } x_2 = x,\ x_n = x + 1,\ x_i \in \{x, x + 1\}, n \geq 3 \end{array}} \tag{3.2′}$$

The last pattern to be studied in detail is (3.3):

$$w = ab^{x_1}ab^{x_2}a \ldots ab^{x_n}aa \text{ with } x_i > 0 \text{ for } 1 \leq i \leq n.$$

Since w necessarily has bb as a factor, $x_i \geq 2$ for some $1 \leq i \leq n$. If $n = 1$ then $x_1 \geq 2$. We shall prove now that if $n > 1$ then $x_1 = 2$, $x_2 = \ldots = x_n = 1$. Let $j = \max\{1 \leq i \leq n \mid x_i \geq 2\}$. For the subword

$$v = \underbrace{ab^{x_j - 1}}_{\alpha} \mid \underbrace{ba \ldots aba}_{\beta} a$$

of w, where $x_j \geq 2$ and $x_{j+1} = \ldots = x_n = 1$, we have

$$\#_a(\beta) \cdot \#_b(\alpha) = (n - j + 1) \cdot (x_j - 1) \geq 1 \cdot (n - j + 1) = \#_a(\alpha) \cdot \#_b(\beta),$$

implying that v is not solvable, due to Proposition 5. If $j > 1$, v is a proper subword of w, which contradicts the minimal unsolvability of w. Hence, $x_i \leq 1$ for

$i > 1$. Thus, there are two possibilities for w of the form (3.3):

$$\boxed{ab^x aa, \text{ with } x > 2 \quad \text{or} \quad abb(ab)^k aa, \text{ with } k \geq 0} \tag{3.3′}$$

In sum, we have established that minimal unsolvable word must satisfy one of the following three forms:

$$\boxed{\begin{array}{l} ab^{x+k}ab^x a, \text{ with } x > 0,\ k > 2 \quad \text{or} \\ ab^{x+2}(ab^{x+1})^* ab^x a, \text{ with } x > 0 \quad \text{or} \\ ab^{x_1}ab^{x_2}a \ldots ab^{x_n}a, \text{ with } x_1 = x+1, x_n = x, x_i \in \{x, x+1\}, x > 0, n \geq 3 \end{array}} \tag{3.1′}$$

$$\boxed{\begin{array}{l} bab^x(ab^{x+1})^* ab^{x+2}, \text{ with } x > 0 \quad \text{or} \\ bab^{x_2}ab^{x_3}a \ldots ab^{x_n}, \text{ with } x_2 = x, x_n = x+1, x_i \in \{x, x+1\}, x > 0, n \geq 3 \end{array}} \tag{3.2′}$$

$$\boxed{ab^x aa, \text{ with } x > 2 \quad \text{or} \quad abb(ab)^k aa, \text{ with } k \geq 0} \tag{3.3′}$$

The sets of words generated by all patterns (3.1′)-(3.3′) are mutually disjoint. Indeed, the forms in (3.1′) are different since the lengths of b-blocks in the first form differ by $k > 2$, in the second form differ by exactly 2, and in the third form differ by 1. Similar argument can be applied to distinguish the forms in (3.2′) and in (3.3′). Every word satisfying the forms from (3.2′) begins with the same letter that occurs as long blocks inside the word (in (3.2′) this letter is b, but due to the symmetry we consider the forms up to swapping a and b). This is different from the words satisfying (3.1′) or (3.3′), hence (3.2′) is disjoint with (3.1′) and with (3.3′). Since all the words from (3.3′) end with the doubled letter, which is never the case for words satisfying (3.1′), the forms (3.1′) and (3.3′) are also disjoint. Thus, all the forms represent disjoint sets of muws.

With this refined classification of the possible forms of muws, in the following section we shall suggest a (generative) language-theoretical characterisation of them. To this aim, we divide all the muws into classes of extendable and non-

extendable words, and demonstrate how (some) muws can be derived from other muws.

3.3 Extension of muws

In this section we provide a complete characterisation of the class of minimal unsolvable binary words. The general idea is to split the whole set into two disjoint classes: *extendable* and *non-extendable*. The extendable words turn out to serve as origins for more complex minimal unsolvable words. For instance, the muw *abbbaba* can be extended to *abababababaabababaa* which is also minimal unsolvable. The extension is given by mapping the initial a into ab, the final a into aa, every other a into aab and every b into ab. This rule will be defined formally as an *extension operation* in this section. The non-extendable words might be also regarded as origins for more complex unsolvable, but not minimal, binary words. As an example of this class, consider the muw *abbbaa*. If we apply the same principle of the extension to this muw, we obtain *abababababaabaa* which is also unsolvable but not minimal, since its proper subword *abababaabaa* is unsolvable. In the class of extendable muws we distinguish a subclass of simplest extendable ones (*abbbaba* is one of them). The words of this subclass satisfy the pattern (2.5)

$$\boxed{(\ a\ b\ \alpha\ )\ b^*\ (\ b\ a\ \alpha\ )^+\ a\ , \qquad \text{with } \alpha \in \{a,b\}^*} \tag{2.5}$$

with the factor α in the form a^i or b^i. Such words are called *base extendable*. After introducing the class of base extendable words, we provide an *extension operation* based on simple extension morphisms. Morphisms which are reciprocal to the extension morphisms are used in the subsequent section, where we define the converse procedure, called *compression*.

3.3.1 Extendable and non-extendable muws

The following definitions should be understood modulo swapping a/b. As in the second part of the previous section, we focus on binary words not containing the infix aa.

We are now introducing the formal definitions of the (base) extendable and

non-extendable minimal unsolvable words.

Definition 22. (base extendable words) *A word* $u \in \{a,b\}^*$ *is called* base extendable *if it is of the form*

$$\begin{aligned} &abb^j(bab^j)^k a \;\; \textit{with} \;\; j > 0,\; k \geq 1, \quad \textit{or} \\ &bab^j(abb^j)^k b \;\; \textit{with} \;\; j \geq 0,\; k \geq 1. \end{aligned} \tag{3.4}$$

The class of base extendable words is denoted by $\mathcal{BE}$.

The word *abbbaba*, considered above, satisfies the first pattern of Definition 22, and hence it is a base extendable word. The simplest (shortest) word satisfying the second pattern of the definition is *baabb* (obtained for $j = 0$ and $k = 1$). It is easy to see that this word is essentially the shortest muw *abbaa* modulo swapping a and b.

Definition 23. (non-extendable words) *A word* $u \in \{a,b\}^*$ *is called* non-extendable *if it is of the form*

$$abb^j b^k bab^j a \;\; \textit{with} \;\; j \geq 0,\; k \geq 1.$$

The class of all non-extendable words is denoted by $\mathcal{NE}$.

The word *abbbaa* satisfies the definition, and hence is non-extendable.

We now establish that all words from classes $\mathcal{BE}$ and $\mathcal{NE}$ are minimal unsolvable.

Proposition 17. [EBMP16](minimal unsolvability of base extendable and non-extendable words) *If* w *belongs to the class* $\mathcal{BE}$ *or the class* $\mathcal{NE}$*, then it is unsolvable and minimal with that property.*

Proof. A word w is a muw if and only if w is unsolvable and both every proper prefix and every proper suffix of w are solvable. Every word w from $\mathcal{BE} \cup \mathcal{NE}$ is of the form (2.5), hence unsolvable. We shall prove the minimality of such a w by providing Petri nets solving its maximal proper prefix and suffix.

Case 1 (base extendable words):

(a) $w = abb^j(bab^j)^k a$, with $j \geq 0$ and $k \geq 1$. This form satisfies (2.5) with $\alpha = b^j$, the star * being repeated zero times, and the plus $^+$ being repeated k times. Due to Proposition 4, all binary words of this form are unsolvable.

The maximal proper prefix $abb^j(bab^j)^k$ of this word can be solved by the Petri net N_1 in Fig. 3.5. Place q in this net enables the initial a, and then disables it unless b has been fired $j+2$ times. After the execution of the block bb^jb there are $k-1$ tokens more than a needs to fire on place q. These surplus tokens allow a to be fired after each sequence b^jb, but not earlier. Place p has initially 1 token on it, which is necessary to execute the block bb^jb after the first a, and this place has only $j+1$ tokens after each next a, preventing b at states where a must occur. Place d prevents a premature occurrence of b at the very beginning of the prefix, and places c_a and c_b restrict the total number of firings of a and b, respectively.

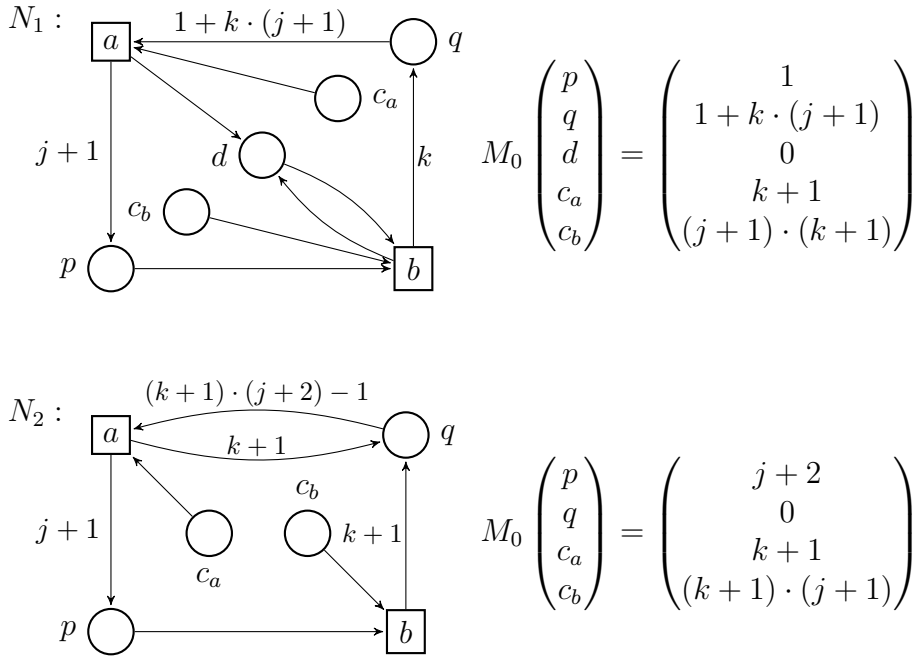


Figure 3.5: N_1 solves the prefix $abb^j(bab^j)^k$. N_2 solves the suffix $bb^j(bab^j)^ka$.

For the maximal proper suffix $bb^j(bab^j)^ka$ of w, one can consider the Petri net N_2 in Fig. 3.5 as a possible solution. Indeed, place q prevents premature occurrences of a in the first block bb^jb, and enables a only after this and each next block b^jb. Doing so, it collects one additional token after each b^jb, which allows this place to enable the very last a after sequence b^j. The initial marking allows to execute the sequence bb^jb at the beginning, and at most $j+1$ b's in a row after that, thanks to place p. Place c_b restricts the total number of b's allowing only

block b^j at the end.

Thus we deduce that any word of the form $abb^j(bab^j)^k a$ with $j > 0$ and $k \geq 1$ is a muw.

(b) $w = bab^j(abb^j)^k b$, with $j \geq 0$ and $k \geq 1$. The word w satisfies (2.5) with swapped a and b, $\alpha = b^j$, the star * being repeated zero times, and the plus $^+$ being repeated k times. Due to Proposition 4, all binary words of this form are unsolvable. Petri nets N_1 and N_2 in Fig. 3.6 are possible solutions for the maximal proper prefix and the maximal proper suffix of w, respectively.

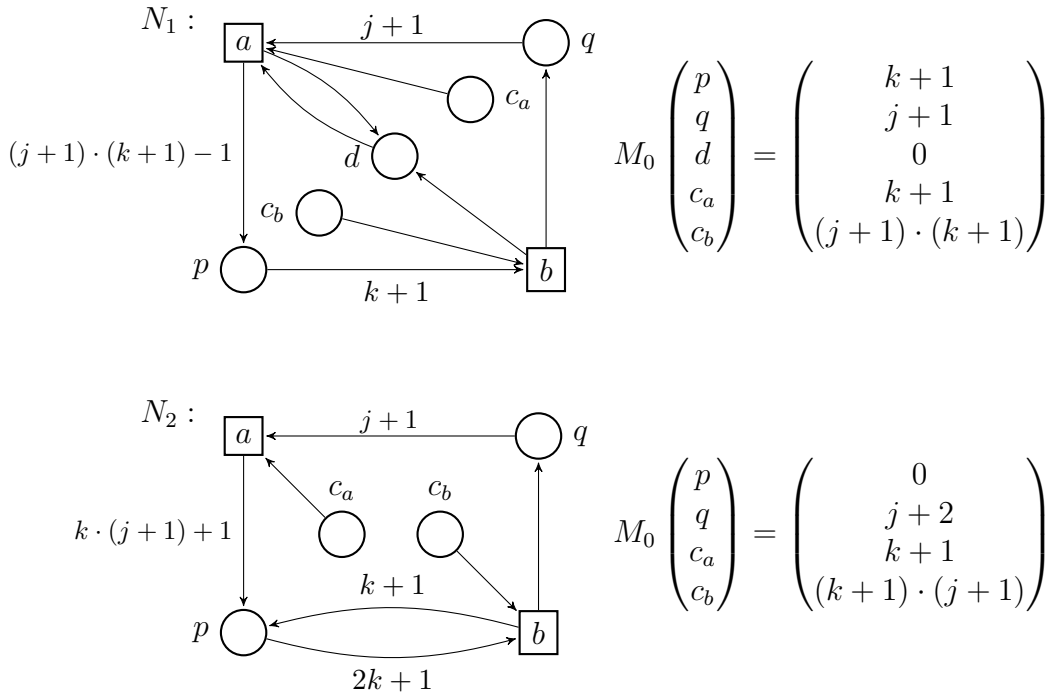
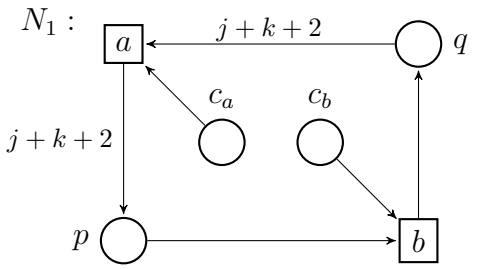


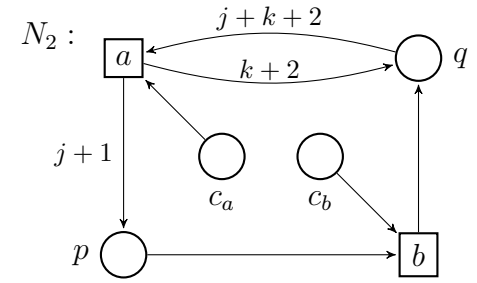
Figure 3.6: N_1 solves the prefix $bab^j(abb^j)^k$. N_2 solves the suffix $ab^j(abb^j)^k b$.

Remark (On the special structure of Petri nets solving prefixes and suffixes): The Petri net N_1 in Fig. 3.5, which solves the maximal proper prefix $abb^j(bab^j)^k$ of the word $w = abb^j(bab^j)^k a$ in the class $\mathcal{BE}$, has a special structure. Place d serves for preventing an undesirable b at the very beginning of w, and places c_a and c_b restrict the total number of a and b, correspondingly. So, the internal structure of the word, being executed by N_1, is completely determined by the two places p and q, which prevent b and a, respectively, exactly at those states where it is necessary.

For later purpose, we refer it places p, q and transitions a, b in Fig. 3.5 as *core parts* of these nets, and to the remaining places as *additional parts*.



$$M_0 \begin{pmatrix} p \\ q \\ c_a \\ c_b \end{pmatrix} = \begin{pmatrix} 0 \\ j+k+2 \\ 2 \\ 2\cdot(j+1)+k \end{pmatrix}$$



$$M_0 \begin{pmatrix} p \\ q \\ c_a \\ c_b \end{pmatrix} = \begin{pmatrix} j+k+2 \\ 0 \\ 2 \\ 2\cdot(j+1)+k \end{pmatrix}$$

Figure 3.7: N_1 solves the prefix $abb^jb^kbab^j$. N_2 solves the suffix $bb^jb^kbab^ja$.

Case 2 (non-extendable words):

We now demonstrate that any binary word of the form $w = abb^jb^kbab^ja$ with $j \geq 0$ and $k \geq 1$ in the class $\mathcal{NE}$ is minimal unsolvable. For w (2.5) is satisfied with $\alpha = b^j$, the star * being repeated k times, and the plus $^+$ being repeated only once. Due to Proposition 4, w is unsolvable. To show the minimality of w, we provide Petri nets N_1 and N_2 (see Fig. 3.7) solving its maximal proper prefix and its maximal proper suffix, respectively. □

Example 16. *Let us consider a word $w = abbbaba$, which is of the form (2.5), with $\alpha = b$, the star * being repeated zero times, and the plus $^+$ being repeated just once. By Definition 22, w is a base extendable word with $j = 1$ and $k = 1$. The word w is unsolvable (by Proposition 4) and minimal with that property. We show the minimality by introducing Petri nets solving a proper prefix abbbab and a proper suffix bbbaba of w. Those Petri nets, constructed on the basis of the proof of Proposition 17, are depicted in Fig 3.8.*

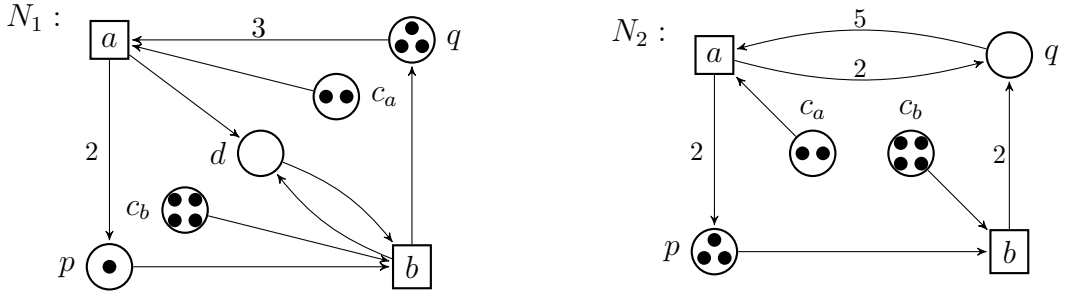


Figure 3.8: N_1 solves the prefix *abbbab*. N_2 solves the suffix *bbbaba*.

Notice that both Petri nets contain the core parts consisting of places p and q, which are responsible for the required behaviour of the nets, as well as auxiliary places – a delay place d and counter places c_a and c_b.

3.3.2 Extension morphisms and operations

Let us now explain how some minimal unsolvable words can be obtained from other minimal unsolvable words. For this purpose we use the following notion of *extension operation*:

Definition 24. (extension operation) *For a word $u = xwx$ with $w \in \{a,b\}^*$, $x \in \{a,b\}$, an* extension operation *E is defined as follows:*

$$\begin{aligned} E(awa) &= \bigcup_{i=1}^{\infty} \Big\{ abM_{a,i}(w)a^{i+1},\ aM_{b,i}(wa) \Big\}, \\ E(bwb) &= \bigcup_{i=1}^{\infty} \Big\{ baM_{b,i}(w)b^{i+1},\ bM_{a,i}(wb) \Big\}, \end{aligned}$$

where $M_{a,i}$ and $M_{b,i}$ are morphisms defined as follows

$$M_{a,i} = \begin{cases} a \mapsto a^{i+1}b \\ b \mapsto a^i b \end{cases} \qquad \text{and} \qquad M_{b,i} = \begin{cases} a \mapsto b^i a \\ b \mapsto b^{i+1}a \end{cases}.$$

From a non-extendable word this operation derives again unsolvable but not minimal words.

Example 17. *For the minimal unsolvable word abbbaba, which is in the class $\mathcal{BE}$,*

using the extension operation E we can obtain, for instance, the word

$$ababababaababaa = abM_{a,1}(bbbab)a^2 \in E(abbbaba).$$

This word is minimal unsolvable. If we apply the operation E to the muw abbbaa, which is in the class $\mathcal{NE}$, in the similar way, then we obtain

$$ababababaabaa = abM_{a,1}(bbba)a^2 \in E(abbbaa).$$

This word is unsolvable, but not minimal with this property, since its proper subword ababababaabaa is also unsolvable.

In what follows, for a given $w \in \{a, b\}^*$, we shall call $u \in E(w)$ an *extension* of w (when $E(w)$ is defined).

Proposition 18. [EBMP16](unsolvability of extensions of non-extendable words) *If $w \in \mathcal{NE}$, then any extension $u \in E(w)$ is unsolvable but not minimal unsolvable.*

Proof. Consider an arbitrary $w \in \mathcal{NE}$, where $w = abb^jb^kbab^ja$ with $j \geq 0$, $k \geq 1$. Depending on the particular morphism $M_{x,i}$ with $x = a$ or $x = b$ for some $i \geq 1$, the extension $u_x \in E(w)$ of $w = aw_1a$ can be constructed as

$$\begin{aligned} u_a = aM_{a,i}(w_1)a^{i+1} &= a\ a^ib\ (a^ib)^j\ (a^ib)^k\ a^ib\ a^{i+1}b\ (a^ib)^j\ a^{i+1} = \\ &= a\ (a^ib)^{k-1}\ a^{i-1}\ ab\ \underbrace{a^ib\ (a^ib)^j\ a^i}_{\alpha_a}\ |\ ba\ \underbrace{a^ib\ (a^ib)^j\ a^i}_{\alpha_a}\ a \end{aligned}$$

or

$$\begin{aligned} u_b = aM_{b,i}(w_1a) &= a\ b^{i+1}a\ (b^{i+1}a)^j\ (b^{i+1}a)^k\ b^{i+1}a\ b^ia\ (b^{i+1}a)^j\ b^ia = \\ &= (ab^{i+1})^k\ ab\ \underbrace{b^ia\ (b^{i+1}a)^j\ b^i}_{\alpha_b}\ |\ ba\ \underbrace{b^ia\ (b^{i+1}a)^j\ b^i}_{\alpha_b}\ a, \end{aligned}$$

respectively. By Proposition 4, the word $ab\alpha_bba\alpha_ba$ is unsolvable, which implies the unsolvability of u_b. Due to $k \geq 1$, $ab\alpha_bba\alpha_ba$ is a proper subword of u_b. Hence, u_b is not minimal unsolvable. Analogously, the unsolvability of $ab\alpha_aba\alpha_aa$ implies the non-minimal unsolvability of u_a.

Argumentation holds modulo swapping a and b. □

We shall show, in what follows, that the extension operation produces muws from the base extendable words and their extensions.

3.3.3 Minimality of extensions

The class of *extendable* minimal unsolvable words is defined recursively basing on the class $\mathcal{BE}$.

Definition 25. (extendable words) *For a word $w \in \{a, b, \}^*$*

1. *if $w \in E(v)$ for some base extendable v, then w is extendable,*
2. *if $w \in E(v)$ for some extendable v, then w is extendable,*
3. *there are no other extendable words.*

The class of all extendable words is denoted by $\mathcal{E}$.

The class of extendable words consists only of muws, which will be shown in the next two lemmata. What is more important: this class is disjoint with the other classes defined earlier, and it completes the classification of all possible muws over the binary alphabet, as we shall see in the next sections.

Lemma 13. [EBMP16](unsolvability of extendable words) *Let $u \in \{a, b\}^*$ be of the form $abv(bav)^k a$ or $bav(abv)^k b$ with $k > 0$. Then $E(u)$ is a set of PN-unsolvable words.*

Proof. Let $u = abv(bav)^k a$ $(k > 0)$. Then

$$\begin{aligned} E(u) &= \bigcup_{i\in\mathbb{N}} \Big\{ aba^i bM_{a,i}(v)\Big(a^i ba^{i+1} bM_{a,i}(v)\Big)^k a^{i+1}, \\ &\qquad ab^{i+1} aM_{b,i}(v)\Big(b^{i+1} ab^i aM_{b,i}(v)\Big)^k b^i a\Big\} \\ &= \bigcup_{i\in\mathbb{N}} \Big\{ ab(a^i bM_{a,i}(v)a^i)\Big(ba(a^i bM_{a,i}(v)a^i)\Big)^k a, \\ &\qquad ab(b^i aM_{b,i}(v)b^i)\Big(ba(b^i aM_{b,i}(v)b^i)\Big)^k a\Big\} \\ &= \bigcup_{i\in\mathbb{N}} \Big\{ abv_i^{(a)}\Big(bav_i^{(a)}\Big)^k a, abv_i^{(b)}\Big(bav_i^{(a)}\Big)^k a\Big\}. \end{aligned}$$

Therefore, by Proposition 4, $E(u)$ is a set of PN-unsolvable words. The case $u = bav(abv)^k b$ can be proved similarly. □

Knowing that members of $\mathcal{E}$ are unsolvable, we now continue with the investigation of their minimality.

Lemma 14. [EBMP16](minimality of extendable words) *If $w \in \mathcal{E}$, then w is minimal unsolvable.*

Proof. (Sketch) By Lemma 13, any extendable word is unsolvable. According to Definition 25, for every $w \in \mathcal{E}$ there is a sequence $w_0, w_1, \ldots, w_r$ such that $w_0 \in \mathcal{BE}$, $w_j \in \mathcal{E}$ and $w_j \in E(w_{j-1})$ for $1 \leq j \leq r$, and $w_r = w$. We will argue by induction on r, and check the existence of Petri nets solving the maximal proper prefix and suffix of w. Every base extendable word w_0 is minimal unsolvable, and there are Petri nets N_1^0 and N_2^0 with core parts and additional parts, which solve the maximal proper prefix and suffix of w_0. Suppose, for $1 \leq j \leq r-1$, there are Petri nets N_1^j and N_2^j doing similar job, for w_j, and which have been obtained from N_1^{j-1} and N_2^{j-1}, respectively, with the appropriate transformations of their core parts. The morphism $M_{x,i}$ with $x \in \{a, b\}$, that has been used to derive w_j from w_{j-1}, determines this transformation uniquely. The inductive step consists of proving that N_1^r and N_2^r obtained from N_1^{r-1} and N_2^{r-1}, respectively, solve the maximal proper prefix and the suffix of w_r. Having morphism $M_{x,i}$, the transformation and two core parts (new and old), it can be directly checked that place p disables/enables transition b in prefix of w_{r-1} as a place of (core part of) N_1^{r-1} if one only if it does the same as the place of (core part of) N_1^r at the corresponding state in prefix of w_r. Similar checking can be done for place q and transition a, and also for suffixes of w_{r-1} and w_r with nets N_2^{r-1} and N_2^r. The additional parts of nets N_1^r and N_2^r can be implemented with a place 'from initial to non-initial' transition, having zero tokens initially and 'sufficiently many' tokens after, and a place which is a counter for the (total) number of firings.

Example 18. *Consider again the word $w = abbbaba$. From the previous considerations (see Example 16) we know that this word is base extendable, and therefore it is a muw. By the application of the extension operation, using the morphism* $M_{a,1} = \begin{cases} a \mapsto aab \\ b \mapsto ab \end{cases}$, *we obtain the word*

$$w_{a,1} = ab\,ababa\,ba\,ababa\,a,$$

	$M_{a,i}$	$M_{b,i}$
$\widetilde{N}_1$	$a^+ \longmapsto a^+ + b^-$ $b^- \longmapsto b^- + i \cdot (a^+ + b^-)$ $b^+ \longmapsto b^+ + i \cdot (a^- + b^+)$ $a^- \longmapsto a^- + b^+$ $M(p) \longmapsto b^- + i \cdot (a^+ + b^-)$ $M(q) \longmapsto a^- + b^+$	$a^+ \longmapsto a^+ + i \cdot (a^+ + b^-)$ $b^- \longmapsto a^+ + b^-$ $b^+ \longmapsto a^- + b^+$ $a^- \longmapsto a^- + i \cdot (a^- + b^+)$ $M(p) \longmapsto a^+ + b^-$ $M(q) \longmapsto a^- + i \cdot (a^- + b^+)$
$\widetilde{N}_2$	$a^+ \longmapsto a^+ + b^-$ $b^- \longmapsto b^- + i \cdot (a^+ + b^-)$ $b^+ \longmapsto b^+ + i \cdot (a_0^- + b^+ - a_0^+)$ $a_0^- \longmapsto a_0^- + b^+$ $a_0^+ \longmapsto a_0^+$ $M(p) \longmapsto b^- + (i+1) \cdot (a^+ + b^-)$ $M(q) \longmapsto 0$	$a^+ \longmapsto a^+ + i \cdot (a^+ + b^-)$ $b^- \longmapsto a^+ + b^-$ $b^+ \longmapsto b^+ + a_0^- - a_0^+$ $a_0^- \longmapsto a_0^- + i \cdot (b^+ + a_0^- - a_0^+)$ $a_0^+ \longmapsto a_0^+$ $M(p) \longmapsto a^+ + (i+1) \cdot (a^+ + b^-)$ $M(q) \longmapsto 0$

Table 3.1: Correspondence between morphisms and transformations

which is of the form (2.5) with $\alpha = ababa$, the star * being repeated zero times, and the plus $^+$ being repeated just once, hence – by Proposition 4 – unsolvable. On the basis of the Petri nets of Fig. 3.8, and according to Table 3.1, we construct Petri nets (depicted in Fig. 3.9) solving the maximal proper prefix abababababaaababa and the maximal proper suffix bababababaababaa of $w_{a,1}$. Thus, $w_{a,1}$ is a minimal unsolvable word.

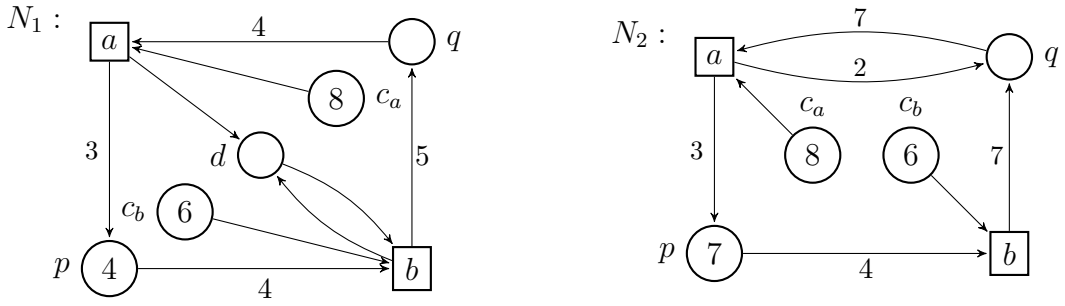


Figure 3.9: N_1 solves the prefix *abababababaaababa* and N_2 solves the suffix *bababababaababaa* of $w_{a,1} = abababababaababaa$.

(Complete proof) Let $w \in \mathcal{E}$ be an arbitrary extendable word. By Lemma 13, w is unsolvable. According to Definition 25, there is a sequence $w_0, w_1, \ldots, w_r$ such

that $w_0 \in \mathcal{BE}$, $w_j \in \mathcal{E}$ and $w_j \in E(w_{j-1})$ for $1 \leq j \leq r$, and $w_r = w$. We will argue by induction on the length r of this sequence. From the previous consideration we know that the base extendable word w_0 is minimal unsolvable, and there are Petri nets N_1^0 and N_2^0 with core parts and additional parts, which are solutions for the maximal proper prefix and the maximal proper suffix of w_0. Assume now that for every $1 \leq j \leq r-1$, there are Petri nets N_1^j and N_2^j which are solutions for the maximal proper prefix and the maximal proper suffix of w_j, and which have been obtained from N_1^{j-1} and N_2^{j-1}, respectively, with the appropriate transformation from Table 3.1 (this transformation is uniquely defined by the morphism $M_{x,i}$ with $x \in \{a, b\}$, that has been used to derive w_j from w_{j-1}). We now prove that knowing morphism $M_{x,i}$ with $x \in \{a, b\}$, which is used for producing w_r from w_{r-1}, and using the corresponding transformation, Petri nets N_1^r and N_2^r, which are derivatives of N_1^{r-1} and N_2^{r-1}, are indeed solutions for the maximal proper prefix and the maximal proper suffix of w_r.

The next four cases are possible:

1. N_1^r derived from N_1^{r-1}, for $w_{r-1} = aw'a$ and $w_r = aM_{b,i}(w'a)$, with $i \geq 1$.
2. N_1^r derived from N_1^{r-1}, for $w_{r-1} = aw'a$ and $w_r = abM_{a,i}(w')a^{i+1}$, with $i \geq 1$.
3. N_2^r derived from N_2^{r-1}, for $w_{r-1} = aw'a$ and $w_r = aM_{b,i}(w'a)$, with $i \geq 1$.
4. N_2^r derived from N_2^{r-1}, for $w_{r-1} = aw'a$ and $w_r = abM_{a,i}(w')a^{i+1}$, with $i \geq 1$.

Let us consider the first case. (The other three cases can be checked analogously.) Let N_1^r be produced from N_1^{r-1}, when $w_{r-1} = aw'a$ and $w_r = aM_{b,i}(w'a)$, for some $i \geq 1$. Having the core part $\widetilde{N}_1^{r-1}$ (see Fig. 3.10) of the solution N_1^{r-1} for aw', with the transformations of the arc weights and the new initial marking

$$\begin{aligned} a^+ &\longmapsto a^+ + i \cdot (a^+ + b^-) \\ b^- &\longmapsto a^+ + b^- \\ b^+ &\longmapsto a^- + b^+ \\ a^- &\longmapsto a^- + i \cdot (a^- + b^+) \\ m\begin{pmatrix} p \\ q \end{pmatrix} &= \begin{pmatrix} a^+ + b^- \\ a^- + i \cdot (a^- + b^+) \end{pmatrix} \end{aligned} \tag{3.5}$$

for morphism $M_{b,i}$ we can construct the new core part $\widetilde{N}_1^r$ for aw'', where $aw''a = w_r$. Let now check that the constructed core part implements the internal part of aw''. We shall show that place p prevents all undesirable b inside aw'' and enables all b that need to occur, and similarly for place q and transition a. Since we have used morphism $M_{b,i}$ for the extension operation, we have a special form of the extension $w_r = ab^{x_1}ab^{x_2}a \ldots ab^{x_n}a \in E(w_{r-1})$, with $x_j \in \{i, i+1\}$. Assume that p disables some b that must occur at state s in

$$aw'' = ab^{x_1}a \ldots ab^{x_k-m} \mid_s b^m a \ldots,$$

where s is the leftmost state in aw'' with this property, and $k \geq 1$. By (3.5), each firing of a brings $a^+ + i \cdot (a^+ + b^-)$ tokens on place p, and b consumes $a^+ + b^-$ tokens on every its occurrence. Hence, p can only disable the last but one b in a group b^{i+1}, i.e. $x_k = i + 1$ and $m = 1$. Assume now that there are l groups of b^{i+1} in $ab^{x_1}a \ldots ab^{x_k-1} \mid_s$. By the initial assumption, the marking of p at s is less than the weight of the arc from p to b, i.e.

$$\begin{aligned} M_s(p) = (a^+ + b^-) + k \cdot (a^+ + i \cdot (a^+ + b^-)) - \\ -k \cdot i \cdot (a^+ + b^-) - l \cdot (a^+ + b^-) < a^+ + b^- \\ \iff \quad (k-l) \cdot a^+ + (1-l) \cdot b^- < b^- \end{aligned}$$

On the other hand, since every sequence $b^{i+1}a$ in w_r corresponds to b in w_{r-1}, and every sequence $b^i a$ corresponds to a, at state s_1 of $w_{r-1} = a \ldots \mid_{s_1} b \ldots a$, where the b right after s_1 corresponds to the block $b^{x_k}a$ in w_r, the marking of place p in net $\widetilde{N}_1^{r-1}$, is $M_{s_1}(p) = b^- + (k-l) \cdot a^+ - l \cdot b^-$. Hence, $M_{s_1}(p) < b^-$, which contradicts the assumption that the net $\widetilde{N}_1^{r-1}$ solves the word aw'. Thus, after the transformation (3.5) place p allows all necessary occurrences of b. Hence place p also allows b to fire initially.

We now have to show that p disables b at all states where a has to occur, except the initial one. Suppose a contrary, i.e. there is a state s in

$$aw'' = ab^{x_1}a \ldots ab^{x_k} \mid_s a \ldots ab^{x_n},$$

with $k \geq 1$, such that $M_s(p) \geq a^+ + b^-$. W.l.o.g. let s to be the leftmost (except

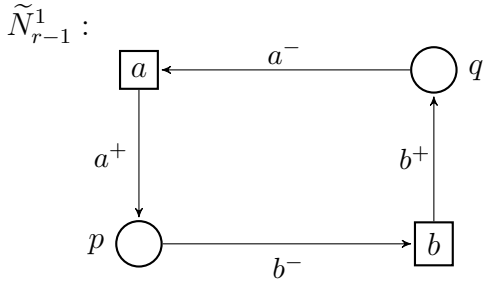


Figure 3.10: Core part of Petri net N_1^{r-1} solving maximal proper prefix of w_{r-1}.

the initial) state with that property. Assume $x_k = i + 1$. For the state s' in

$$ab^{x_1}a \ldots \mid_{s'} ab^{x_k} \mid_s a \ldots ab^{x_n}a$$

we then have

$$\begin{aligned} M_s(p) = M_{s'}(p) + a^+ + i \cdot (a^+ + b^-) - (i+1) \cdot (a^+ + b^-) \geq a^+ + b^- \\ \iff \quad M_{s'}(p) \geq b^- + (a^+ + b^-). \end{aligned}$$

The last inequality means that b is not separated at state s'. If $k = 1$, then, by (3.5), $M_{s'}(p) = M(p) = a^+ + b^-$, which contradicts the last inequality. If $k > 1$, then we get a contradiction to the choice of s. Hence, $x_k = i$. Let l be the number of blocks b^{i+1} in $ab^{x_1}a \ldots ab^{x_k}|_s$. Then

$$\begin{aligned} M_s(p) = (a^+ + b^-) + k \cdot (a^+ + i \cdot (a^+ + b^-)) - \\ - k \cdot i \cdot (a^+ + b^-) - l \cdot (a^+ + b^-) \geq a^+ + b^- \\ \iff \quad (k-l) \cdot a^+ + (1-l) \cdot b^- \geq b^-. \end{aligned}$$

Since w_r has been obtained using morphism $M_{b,i}$, sequence $b^{x_k}a$ corresponds to a letter a in w_{r-1}. Therefore, in $w_{r-1} = a \ldots |_{s_1} a \ldots$, where s_1 fits the state right before $b^{x_k}a$ in w_r, b is not separated at state s_1. This contradicts the assumption that $\widetilde{N}_1^{r-1}$ solves aw'. Thus, in the net $\widetilde{N}_1^r$ that was derived from $\widetilde{N}_1^{r-1}$ by (3.5), p disables b whenever and only if it is necessary inside aw''.

For separating b at the initial marking, one can construct additional place p_1, having 0 tokens on it initially, and being a pure input place for b and pure output place for a, with unit arc weights. For restricting the total number of occurrences

of b, it is enough to construct a place p_2 with $\#_b(w_r)$ tokens on it initially, which is a pure input place for b, with the arc weight equal to 1.

Let us now consider place q and transition a. First we will show that q allows a to fire at each state where this is necessary. It is clear that q enables a initially. Assume that there is a state s in

$$aw'' = ab^{x_1}ab^{x_2}a \ldots ab^{x_k} \mid_s ab^{x_{k+1}}a \ldots,$$

with $k \geq 1$, such that q disables a at s. Due to (3.5), each firing of b brings $a^- + b^+$ tokens on q. Hence $x_k = i$. Suppose that there are l blocks b^{i+1} in $ab^{x_1}a \ldots ab^{x_k} \mid_s$. Then, we have

$$\begin{aligned} M_s(q) \;=\; & a^- + i \cdot (a^- + b^+) - k \cdot (a^- + i \cdot (a^- + b^+)) + \\ & + k \cdot i \cdot (a^- + b^+) + l \cdot (a^- + b^+) < a^- + i \cdot (a^- + b^+) \\ & \iff\; a^- + l \cdot b^+ - (k - l) \cdot a^- < a^- \end{aligned}$$

Due to the fact that $aw''a$ has been obtained from $aw'a$ using morphism $M_{b,i}$, the block $b^{x_k}a$ corresponds to a right after the state s' in $w_{r-1} = a \ldots \mid_{s_1} a \ldots$. The last inequality means that $M_{s_1}(q) < a^-$, which contradicts the assumption that $\widetilde{N}_1^{r-1}$ solves the word aw'. Thus, place q after the transformation (3.5) allows each mandatory firing of a.

We now demonstrate that q disables a at every place, where b has to occur. Suppose that this is not true, i.e. there is a state s in

$$aw'' = ab^{x_1}a \ldots ab^{x_k - m} \mid_s b^m a \ldots,$$

with $k, m > 0$, at which a is enabled by place q. W.l.o.g. let s be the leftmost state in aw'' with that property. Due to the initial marking of q provided in (3.5), $k > 1$. Hence, for state s and place q we have

$$\begin{aligned} M_s(q) \;=\; & a^- + i \cdot (a^- + b^+) - k \cdot (a^- + i \cdot (a^- + b^+)) \\ & + (x_1 + \cdots + x_k - m) \cdot (a^- + b^+) \geq a^- + i \cdot (a^- + b^+). \end{aligned}$$

If $x_{k-1} = i$, then

$$\begin{aligned} M_{s_1}(q) \;&=\; a^- + i \cdot (a^- + b^+) - (k-1) \cdot (a^- + i \cdot (a^- + b^+)) \\ &\quad + (x_1 + \cdots + x_{k-1} - m) \cdot (a^- + b^+) \geq a^- + i \cdot (a^- + b^+) + a^-, \end{aligned}$$

implying that a is enabled by q at state s_1 in

$$aw'' = ab^{x_1}a \ldots ab^{x_{k-1}-m} \mid_{s_1} b^m a \ldots,$$

which contradicts the choice of s. Then, $x_{k-1} = i + 1$. This means, the block $b^{x_k}a$ corresponds to a letter b in $aw'a$, and state s in aw'' corresponds to the state s_0 in $aw' = a \ldots \mid_{s_0} b \ldots$. On the other hand,

$$\begin{aligned} M_s(q) \;&= a^- + i \cdot (a^- + b^+) - k \cdot (a^- + i \cdot (a^- + b^+)) \\ &\quad + (x_1 + \cdots + x_k - m) \cdot (a^- + b^+) \geq a^- + i \cdot (a^- + b^+) \\ &\quad \iff\; a^- - (k-l) \cdot a^- + l \cdot b^+ \geq a^- + (m-1) \cdot (a^- + b^+). \end{aligned}$$

Since $m \geq 1$, we have $M_{s_0}(q) \geq a^-$ in the net $\widetilde{N}_1^{r-1}$, implying that a is enabled at state s_0. This contradicts the fact that $\widetilde{N}_1^{r-1}$ solves aw'. Thus, q disables a at every state in aw'' where b has to occur.

Redundant occurrence of b at the very beginning of aw'', that is not handled by p, can be easily restricted by place p_1, having zero tokens initially, the arc weight from a to p_1 is $i + 1$ and the arc weight from p_1 to b is 1. The length of the execution performed by $\widetilde{N}_1^r$ can be restricted with a letter-counting place, having no inputs and a single output for each transition, and the initial number of tokens equal to the length of aw''. As the result, we have Petri net N_1^r, solving exactly aw'', with the core and the additional parts.

The other three possible cases from Table 3.1 can be checked analogously.

□

So far, we have introduced three classes of minimal unsolvable words over $\{a, b\}$: base-extendable, non-extendable and extendable. These classes are mutually disjoint, and moreover, words from $\mathcal{E}$ are images of words from $\mathcal{BE}$ and $\mathcal{E}$ under the extension operation E, while images of members of $\mathcal{NE}$ under the same operation E are not minimal, though they are unsolvable as well. The goal of the next section

is to complete our classification by demonstrating that all muws are in these three classes. To reach this goal, we will introduce the *compression operation* which will be, in some sense, a complement of the extension operation.

3.4 Compression of muws

Consider minimal unsolvable words w.r.t. the classification obtained in Section 3.2. All possible patterns from (3.1)–(3.3), and more precisely their refined variants from (3.1′)–(3.3′), can be distinguished into base extendable

$ab(ba)^{k+1}a$, with $k \geq 0$, for the second pattern from (3.3′),
$abb^x(bab^x)^k a$, with $x > 0, k > 0$, for the second pattern from (3.1′),
$bab^x(abb^x)^k b$, with $x > 0, k > 0$, for the first pattern from (3.2′),

non-extendable

$abb^{x-1}baa$, with $x \geq 2$ for the first pattern from (3.3′),
$abb^x b^{k-1}bab^x a$, with $x > 0$, $k > 2$ for the first pattern from (3.1′),

and the rest, which we call $\mathcal{C}$ (*compressible*)

$ab^{x_1}ab^{x_2}a \ldots ab^{x_n}a$, with $x_1 = x + 1$, $x_n = x$, $x_i \in \{x, x+1\}, x > 0$, $n \geq 3$,
for the third pattern from (3.1′),
$bab^{x_2}ab^{x_3}a \ldots ab^{x_n}$, with $x_2 = x$, $x_n = x + 1$, $x_i \in \{x, x+1\}, x > 0$, $n \geq 3$,
for the second pattern from (3.2′).

From this classification we derive that

$$\mathcal{MUW} = \mathcal{BE} \cup \mathcal{NE} \cup \mathcal{C},$$

where $\mathcal{BE}$, $\mathcal{NE}$ and $\mathcal{C}$ are mutually disjoint classes, and $\mathcal{MUW}$ denotes the class of all minimal unsolvable words. Note, that since all words in the class $\mathcal{E}$ are unsolvable and minimal with that property, and $\mathcal{E}$ is disjoint with $\mathcal{BE}$ and $\mathcal{NE}$, we have $\mathcal{E} \subseteq \mathcal{C}$. In the rest of this section and in the next section we proceed to show

$\mathcal{C} \subseteq \mathcal{E}$.

In the previous sections we demonstrated how to construct new minimal unsolvable words on the basis of extendable words. The purpose of this section is to introduce an inverse transformation, which allows to compress longer minimal unsolvable words into shorter ones.

Definition 26. (compression function) *For a word $v = xux$ with $u \in \{a, b\}^*$, $x \in \{a, b\}$, a* compression function *C is defined as follows :*

$$\begin{aligned} C(abua^{i+1}) &= aM_{a,i}^{-1}(u)a, \quad C(baub^{i+1}) = bM_{b,i}^{-1}(u)b, \\ C(auba) &= aM_{b,i}^{-1}(uba), \quad C(buab) = bM_{a,i}^{-1}(uab), \end{aligned} \tag{3.6}$$

where $i \geq 1$ and $M_{a,i}^{-1}$, $M_{b,i}^{-1}$ are morphisms defined as follows:

$$M_{a,i}^{-1} : \begin{cases} a^{i+1}b & \mapsto a \\ a^ib & \mapsto b \end{cases} \quad \text{and} \quad M_{b,i}^{-1} : \begin{cases} b^ia & \mapsto a \\ b^{i+1}a & \mapsto b. \end{cases}$$

Among all possible forms from the classification of minimal unsolvable words, C is defined exactly for patterns from the class $\mathcal{C}$. Moreover, the form of the word explicitly defines the morphism $M_{x,i}^{-1}$ which is used when applying C to the word. Let us notice that since $\mathcal{E} \subseteq \mathcal{C}$, all words from $\mathcal{E}$ are compressible by C.

From Definitions 24 and 26, it is clear that $M_{x,i}$ is reciprocal to $M_{x,i}^{-1}$ for $x \in \{a, b\}$, $i \geq 1$. The following lemma establishes that the extension operation E and the application of compression function C are complements of each other.

Lemma 15. (compression and extension functions)

1. *If $v \in \mathcal{BE} \cup \mathcal{E}$ and $u \in E(v)$, then $C(u) = v$;*
2. *If $u \in \mathcal{C}$ and $v = C(u)$, then $u \in E(v)$.*

Proof. 1. Consider some $v \in \mathcal{BE} \cup \mathcal{E}$. Then $v = xv_1x$, where $x \in \{a, b\}$. Hence, for distinct $x, y \in \{a, b\}$ and $i \geq 1$, we have two possible cases:

- $u = xyM_{x,i}(v_1)x^{i+1}$. By the definition of the compression function,

$$C(u) = C(xyM_{x,i}(v_1)x^{i+1}) = xM_{x,i}^{-1}(M_{x,i}(v_1))x = xv_1x = v.$$

- $u = xM_{y,i}(v_1x)$. By the definition of the compression function,

$$C(u) = C(xM_{y,i}(v_1x)) = C(xM_{y,i}(v_1)y^ix) = xM_{y,i}^{-1}(M_{y,i}(v_1)y^ix) = xv_1x = v.$$

2. Consider some $u \in \mathcal{C}$. According to the definition of $\mathcal{C}$, u starts and ends with $x \in \{a, b\}$. Due to Definition 26 of C and the definition of $\mathcal{C}$, u uniquely determines which compression morphism can be applied to it. Two cases are possible:

- $v = C(u) = xM_{x,i}^{-1}(u_1)x$ for $u = xyu_1x^{i+1}$, $x \neq y \in \{a, b\}$. Then,

$$E(v) = \bigcup_{j=1}^{\infty}\{xyM_{x,j}(M_{x,i}^{-1}(u_1))x^{i+1}, xM_{y,j}(M_{x,i}^{-1}(u_1)x)\}.$$

  As $u = xyM_{x,j}(M_{x,i}^{-1}(u_1))x^{i+1}$ for $j = i$, then $u \in E(v)$.

- $v = C(u) = xM_{y,i}^{-1}(u_1xy^ix)$ for $u = xu_1xy^ix$, $x \neq y \in \{a, b\}$. Then,

$$E(v) = \bigcup_{j=1}^{\infty}\{xyM_{x,j}(M_{y,i}^{-1}(u_1x))x^{i+1}, xM_{y,j}(M_{y,i}^{-1}(u_1xy^ix))\}.$$

  As $u = xM_{y,j}(M_{y,i}^{-1}(u_1xy^ix))$ for $j = i$, then $u \in E(v)$.

□

3.5 The generative nature of muws

In this section we will demonstrate that applying the compression function to muws (in class $\mathcal{C}$) is an endomorphism within the class $\mathcal{MUW}$, i.e. the results are also minimal unsolvable words. We begin with three technical lemmas. The first of them supports the intuition that letter a is separable within a block of b's only if it is separable at the end of the block.

Lemma 16. *Suppose $w = \alpha|_s b^{m-1}|_{\tilde{s}} ba\beta$, with $m \geq 1$. If a is not separable at state s, then it is not separable at state $\tilde{s}$, as well.*

Proof. By contraposition. Assume there is a Petri net $N = (P, T, F, M_0)$ with a place $p \in P$ such that w can be fired completely, and $M_{\tilde{s}}(p) < F(p, a)$. Since a is

enabled at state s' such that $\tilde{s}[b\rangle s'$, then $\mathbb{E}(p)(b) > 0$. Hence, $M_s(p) \leq M_{\tilde{s}}(p) < F(p,a)$, i.e. a is separable at state s with place p. □

The next two lemmas explain how a possible failure of separation of the starting (and hence the ending) letter from a state is connected with the position of this state inside the blocks of a muw. We consider both patterns from the definition of the class $\mathcal{C}$, and we will see that the letters cannot be separated either in the end of the long blocks, or immediately after the short blocks, depending on the chosen pattern.

Lemma 17. *If $w = ab^{x_1}ab^{x_2}a \ldots ab^{x_n}a$, with $x_1 = x+1$, $x_n = x$, $x_i \in \{x, x+1\}$, $x > 0$, $n \geq 3$, is a minimal unsolvable word, and separation failure occurs in group b^{x_k}, then $x_k = x+1$.*

Proof. By Lemma 16, a is not separated at some state s in

$$w \;=\; \underbrace{\overbrace{a\, b^{x_1}\, a\; \ldots\; a\, b^{x_{k-1}-1}}^{\alpha'} \mid \overbrace{b\, a\, b^{x_k-1}}^{\beta'}}_{\alpha} \mid_s \underbrace{\overbrace{b\, a\; \ldots\; a\, b^{x_{n-1}}}^{\beta}\, a\, b^{x_n}}_{\beta}\; a,$$

implying, according to Lemma 4, that

$$(x_1 + \ldots + x_k - 1) \cdot (n-k) \;\geq\; (1 + x_{k+1} + \ldots + x_n) \cdot k$$

which can be rewritten as

$$\frac{x_1 + \ldots + x_k - 1}{k} \;=\; \frac{\#_b(\alpha)}{\#_a(\alpha)} \;\geq\; \frac{\#_b(\beta)}{\#_a(\beta)} \;=\; \frac{1 + x_{k+1} + \ldots + x_n}{n-k},$$

where $\#_a(\alpha) \neq 0$ and $\#_a(\beta) \neq 0$. Assume now that $x_k = k$. Since for every $1 \leq i \leq n$ we have $x \leq x_i \leq x+1$, then

$$\frac{\#_b(\alpha')}{\#_a(\alpha')} \geq \frac{\#_b(\alpha)}{\#_a(\alpha)},$$

where $\#_a(\alpha') \neq 0$ because w starts with a. From $x_1 = x+1$, it follows that $k > 1$. Due to $x_n = x = x_k$,

$$\frac{\#_b(\beta')}{\#_a(\beta')} = \frac{\#_b(\beta)}{\#_a(\beta)},$$

where $\#_a(\beta') \neq 0$ since $k > 1$. Thus,

$$\frac{\#_b(\alpha')}{\#_a(\alpha')} \geq \frac{\#_b(\beta')}{\#_a(\beta')},$$

which implies, by Lemma 4, the unsolvability of $\alpha'\beta'a$, contradicting the minimality of w. □

The other pattern from the definition of class $\mathcal{C}$ is handled similarly:

Lemma 18. *If $w = bab^{x_2}ab^{x_3}a \ldots ab^{x_n}$, with $x_2 = x$, $x_n = x + 1$, $x_i \in \{x, x + 1\}$, $x > 0$, $n \geq 3$, is a minimal unsolvable word, and separation failure occurs after group b^{x_k}, then $x_k = x$.*

Proof. For state s in w, from which b is not separated,

$$w = \overbrace{\underbrace{b\,a\,b^{x_2-1}\,b\,a\ \ldots\ a\,b^{x_{k-1}}}_{\alpha}\ |\ a\,b^{x_k}}^{\alpha'}\ |_s\ \overbrace{\underbrace{a\ \ldots\ a\,b^{x_{n-1}-1}\,b\,a\,b^{x_n-1}}_{\beta}}^{\beta'}\ b.$$

According to Lemma 4, we have

$$(k-1) \cdot (x_{k+1} + \ldots + x_n - 1) \geq (1 + x_2 + \ldots + x_k) \cdot (n - k),$$

which is equivalent to

$$\frac{x_{k+1} + \ldots + x_n - 1}{n-k} = \frac{\#_b(\beta)}{\#_a(\beta)} \geq \frac{\#_b(\alpha)}{\#_a(\alpha)} = \frac{1 + x_2 + \ldots + x_k}{k-1},$$

where $\#_a(\beta) \neq 0$ since β starts with a, and $\#_a(\alpha) \neq 0$ because $k > 1$. Assume $x_k = k + 1$. Since for all $2 \leq i \leq n$, we have $x \leq x_i \leq x + 1$,

$$\frac{\#_b(\alpha')}{\#_a(\alpha')} \leq \frac{\#_b(\alpha)}{\#_a(\alpha)},$$

where $\#_a(\alpha') \neq 0$ because $k > 2$. From $x_n = x + 1 = x_k$ it follows that

$$\frac{\#_b(\beta)}{\#_a(\beta)} = \frac{\#_b(\beta')}{\#_a(\beta')},$$

where $\#_a(\beta') \neq 0$ due to β' starting with a. Hence,

$$\frac{\#_b(\beta')}{\#_a(\beta')} \geq \frac{\#_b(\alpha')}{\#_a(\alpha')}.$$

Due to Lemma 4, this implies the unsolvability of $\alpha'\beta'b$, contradicting the minimality of w. □

Consider now an arbitrary minimal unsolvable word

$$w = aw_1 = ab^{x_1}ab^{x_2}a \ldots ab^{x_n}a$$

in $\mathcal{C}$, with $x_1 = x + 1$, $x_n = x$, $x_i \in \{x, x+1\}$, $x > 0$, $n \geq 3$. According to the special form of w, C can merely be applied to w in the form $C(w) = aM_{b,x}^{-1}(w_1)$. Note that $u = C(w)$ is also unsolvable. Due to Lemma 17, for state s in

$$w = \underbrace{a\ b^{x_1}\ a\ \ldots\ a\ b^{x_k-1}}_{\alpha}\ |_s\ \underbrace{b\ a\ \ldots\ a\ b^{x_n}}_{\beta}\ a,$$

from which a is not separated, we have $x_k = x + 1$. By Lemma 4,

$$(n-k) \cdot (x_1 + x_2 + \cdots + x_k - 1) \geq k \cdot (x_{k+1} + \cdots + x_n + 1)$$

Assume that there are l groups of b^x in α (except the part of b^{x_k}), and m groups of b^x in β. Due to the form of w, we have $0 \leq l < k - 1$ and $0 < m \leq n - k$. Hence,

$$\begin{aligned} & \#_a(\beta) \cdot \#_b(\alpha) \geq \#_a(\alpha) \cdot \#_b(\beta) \\ \iff\ & (n-k) \cdot (k \cdot (x+1) - l - 1) \geq k \cdot ((n-k) \cdot (x+1) - m + 1) \\ \iff\ & k \cdot l + k \cdot m - n \cdot l - n \geq 0. \end{aligned}$$

After applying the compression function to w, due to the definition of C and $M_{b,x}^{-1}$, for every sequence $b^x a$ and for every sequence $b^{x+1}a$ in w, we obtain a and b in u, respectively. Hence, u has $n + 1$ letters in all, starts with ab and ends with a, thanks to the definition of C and the shape of w, and, by Lemma 17, has b on the

$(k+1)$th position:

$$u \;=\; \underbrace{a\,b\ldots}_{\alpha'}\;|_{s'}\;\underbrace{b\;\ldots}_{\beta'}\;a,$$

where $|\alpha'| = k$, $|\beta'| = n-k$. Moreover, $\#_a(\alpha') = l+1$ and $\#_a(\beta') = m-1$. Thus, we have $\#_a(\beta')\cdot\#_b(\alpha') = m\cdot(k-l-1)$ and $\#_a(\alpha')\cdot\#_b(\beta') = (l+1)\cdot(n-k-m+1)$. Then,

$$\#_a(\beta')\cdot\#_b(\alpha') - \#_a(\alpha')\cdot\#_b(\beta') \;=\; k\cdot l + k\cdot m - n\cdot l - n + k - l - 1 \;\geq 0,$$

where the last inequality is because of $k\cdot l + k\cdot m - n\cdot l - n \geq 0$ and $l < k-1$. Due to Lemma 4, this implies the unsolvability of u.

Let us now consider an arbitrary minimal unsolvable word

$$w = bab^{x_2}ab^{x_3}a\ldots ab^{x_n}$$

from the class $\mathcal{C}$, with $x_2 = x$, $x_n = x+1$, $x_i \in \{x, x+1\}$, $x > 0$, $n \geq 3$, and check that $u = C(w)$ is unsolvable as well. The form of $w = bw_1b^{x+1}$ explicitly determines that $C(w) = bM_{b,x}^{-1}(w_1)b$. By Lemma 18, for state s from which b is not separated in

$$w \;=\; \underbrace{b\,a\,b^{x_2}\,a\;\ldots\;b^{x_k}}_{\alpha}\;|_s\;\underbrace{a\,b^{x_{k+1}}\,a\;\ldots\;a\,b^{x_n-1}}_{\beta}\;b,$$

we have $x_k = x$. From Lemma 4,

$$(k-1)\cdot(x_{k+1} + \ldots + x_n - 1) \geq (1 + x_2 + \ldots + x_k)\cdot(n-k).$$

Assume, there are l groups of b^{x+1} in α and m groups of b^{x+1} in β. Due to the form of w, we have $0 \leq l < k$ and $0 \leq m \leq n-k$, and

$$\begin{aligned} &(k-1)\cdot(x\cdot(n-k)+m) \geq (1 + x\cdot(k-1) + l)\cdot(n-k) \\ \iff\; &k\cdot m - m - n + k - l\cdot n + l\cdot k \geq 0. \end{aligned}$$

After applying the compression function C to w, according to the definition of $M_{b,x}^{-1}$, for every sequence $b^{x+1}a$ and every sequence $b^x a$ in w, we obtain a and b in

u, respectively. Hence, u has n letters in all, starts with ba and ends with b, by the definition of C and the special shape of w. By Lemma 18, u has a on its kth position:

$$u = \underbrace{ba\ldots}_{\alpha'} |_{s'} \underbrace{a\ldots}_{\beta'} b,$$

where $|\alpha'| = k-1$, $|\beta'| = n-k$. Moreover, $\#_b(\alpha') = l$ and $\#_b(\beta') = m$. Thus, $\#_a(\alpha') \cdot \#_b(\beta') = (k-1-l) \cdot m$ and $\#_b(\alpha') \cdot \#_a(\beta') = l \cdot (n-k-m)$. Then,

$$\begin{aligned} \#_a(\alpha') \cdot \#_b(\beta') - \#_b(\alpha') \cdot \#_a(\beta') &= k \cdot m - m - l \cdot n + l \cdot k \geq \\ &\geq k \cdot m - m - l \cdot n + l \cdot k + k - n \geq 0. \end{aligned}$$

By Lemma 4, this means that u is unsolvable.

So far, we have shown that the compressed image of any word in $\mathcal{C}$ is unsolvable. Now we shall prove that $\mathcal{C} \subseteq \mathcal{E}$. Suppose that this is not true, i.e. $\mathcal{C} \setminus \mathcal{E} \neq \emptyset$. Take some shortest word $u \in \mathcal{C} \setminus \mathcal{E}$ and let $w = C(u)$. Since w is unsolvable, two cases are possible:

Case 1: w is a minimal unsolvable word. Due to the choice of u as shortest in $\mathcal{C} \setminus \mathcal{E}$, and the fact that w is shorter than u, we have $w \notin \mathcal{C} \setminus \mathcal{E}$. Hence, w belongs to one of disjoint classes $\mathcal{BE}$, $\mathcal{NE}$, $\mathcal{E}$. If $w \in \mathcal{BE}$ or $w \in \mathcal{E}$, then, by Definition 25 and Lemma 15, $u \in E(w) \subseteq \mathcal{E}$, which contradicts the choice of $u \in \mathcal{C} \setminus \mathcal{E}$. If $w \in \mathcal{NE}$, then by Proposition 18, $u \in E(w)$ is not a minimal unsolvable word, contradicting the minimality of u.

Case 2: w is not a minimal unsolvable word. We shall prove that u is also not a minimal unsolvable word. Assume now that $w = w_1 v w_2$, where v is a minimal unsolvable word and $w_1 w_2 \neq \epsilon$, and that w has been obtained from u using compression morphism $M_{x,i}^{-1}$, where $x \in \{a, b\}$. Since v is a proper subword of w, and w is shorter than u, then $v \notin \mathcal{C} \setminus \mathcal{E}$. From the minimal unsolvability of v we have $v \in \mathcal{BE} \cup \mathcal{E} \cup \mathcal{NE}$. Hence, any extension v' of v is unsolvable (possibly not minimal in case $v \in \mathcal{NE}$). For $x \neq y$, where $x, y \in \{a, b\}$, we have either $v = xv_1x$, or $v = yv_1y$. Consider these two possibilities.

1. $v = xv_1x$. In this case, according to Definition 24, we consider the extension $v' = xyM_{x,i}(v_1)x^{i+1} \in E(v)$. Assume both w_1 and w_2 are non-empty words. Hence $M_{x,i}(v) = x^{i+1}yM_{x,i}(v_1)x^{i+1}y$ is a proper subword of u. As v' is a

subword of $M_{x,i}(v)$, we get a contradiction to the minimal unsolvability of u. Assume that $w_1 = \epsilon$. Then, being a proper prefix of w, after extension v will be morphed to $xyM_{x,i}(v_1)x^{i+1}y$, which again has v' as a subword, implying a contradiction to the minimality of u. If $w_2 = \epsilon$, extension u of w with morphism $M_{x,i}$ has a proper subword $x^{i+1}yM_{x,i}(v_1)x^{i+1}$, and hence, contains v' as well. This contradicts the minimal unsolvability of u.

2. $v = yv_1y$. Let now $v' = yM_{x,i}(v_1y) \in E(v)$. In case w_1 is a non-empty word, $M_{x,i}(v) = x^iyM_{x,i}(v_1y)$ is a proper subword of u, and contains v' as a factor. This contradicts the minimality of u. If $w_1 = \epsilon$, u has v' as a proper prefix, which again contradicts the minimal unsolvability of u.

Thus, $\mathcal{C} = \mathcal{E}$, which establishes the following result on classification of all minimal unsolvable words according to their generative nature

Theorem 6. [EBMP16](generative nature of minimal unsolvable binary words) *Let w be a minimal Petri net unsolvable binary word. Then we have the following exclusive alternatives:*

- *w is a non-extendable word ($w \in \mathcal{NE}$),*
- *w is a base extendable word ($w \in \mathcal{BE}$),*
- *w is an extendable word ($w \in \mathcal{E}$).*

Based on Theorem 6 and the proofs of Proposition 17 and Lemma 13, we can formulate the following

Corollary 1. (a necessary condition for unsolvability) *A word over $\{a, b\}$ is not PN-solvable if and only if it has a subword of the form*

$$\boxed{(\ a\ b\ \alpha\ )\ b^*\ (\ b\ a\ \alpha\ )^+\ a\ , \qquad \text{with } \alpha \in \{a, b\}^*}.$$

In the last case of the alternatives stated in Theorem 6 (i.e., the case $w \in \mathcal{E}$), applying C to w consecutively, we can recover the (unique) sequence of minimal unsolvable words $w_0, w_1, \ldots, w_r$, such that $w_0 \in \mathcal{BE}$, $w_r = w$, $w_i \in \mathcal{E}$ and $w_{i-1} = C(w_i)$ for $1 \leq i \leq r$. Moreover, starting from a word w_0, its maximal proper prefix and maximal proper suffix, and Petri nets solving them (in special forms,

that are provided in the thesis), using appropriate transformations, we can derive Petri nets solving the maximal proper prefixes and the maximal proper suffixes of w_i for all $1 \leq i \leq r$.

Example 19. *Let us consider word $v = ba\,aabaaabaa\,ab\,aabaaabaa\,b$. It is unsolvable by Proposition 4, because it is of the form $ba\alpha\,a^*\,(ab\alpha)^+\,b$ (which is exactly the form (2.5)) with $\alpha = aabaaabaa$, the star * being repeated zero times, and the plus $^+$ being repeated just once. Due to Theorem 6, if v is minimal, then it belongs to one of the classes $\mathcal{BE}$, $\mathcal{NE}$, $\mathcal{E}$. Since it does not fit the patterns of classes $\mathcal{BE}$, $\mathcal{NE}$, we now aim to check whether $v \in \mathcal{E}$. In order to do this we compress v with the function C. It can be seen that the word can be written in the form*

$$v = b(aaab)(aaab)(aaab)(aab)(aaab)(aab),$$

hence we need to consider the morphism

$$M_{a,2}^{-1} : \begin{cases} aaab & \mapsto a \\ aab & \mapsto b \end{cases},$$

and by the compression we obtain the word $v_{a,2}^{-1} = baaabab$. Let us notice that $v_{a,2}^{-1}$ is dual of the word $w = abbbaba$ (see Example 16), up to swapping a and b, hence it is a minimal unsolvable word. Function C cannot be applied to $w = C(v)$, which accords with the fact that $w \in \mathcal{BE}$.

Moreover, starting with the word $w = abbbaba$, together with Petri nets solving its proper prefix and suffix (see Fig. 3.8) and applying the morphism

$$M_{b,2} : \begin{cases} a & \mapsto bba \\ b & \mapsto bbba \end{cases}$$

we obtain the word $w_{b,2} = ab\,bbabbbabb\,ba\,bbabbbabb\,a$ which is dual of v. By the previous considerations we can easily construct Petri nets solving the maximal proper prefix and the maximal proper suffix of $w_{b,2}$, hence, by swapping letters we can obtain Petri nets for a proper prefix and a proper suffix of v. Such nets are depicted in Fig. 3.11. Now we can state that the word v is not only unsolvable, but also minimal with that property.

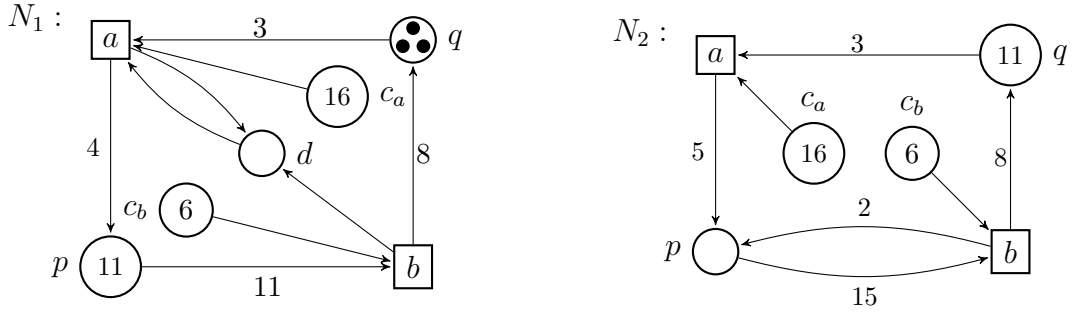


Figure 3.11: N_1 solves the prefix *baaabaaabaaabaabaaabaa* and N_2 solves the suffix *aaabaaabaaabaabaaabaab* of $v = baaabaaabaaabaabaaabaab$.

3.6 A pattern-matching pre-synthesis algorithm

The classification of minimal unsolvable words presented in the previous sections leads to an efficient algorithm for verifying the solvability/unsolvability of a binary word. By Definition 23, all non-extendable words are of the form

(Ia) ab^xab^ya or (Ib) ba^xba^yb,

where $x > y + 2$, $y \geq 0$, and by Definitions 22 and 25, all extendable words (including base extendable ones) are of the form

(IIa) $abw(baw)^ka$ or (IIb) $baw(abw)^kb$,

where $k \geq 1$ and $w \in \{a, b\}^*$.

Recall that a word $v \in \{a, b\}^*$ containing a minimal unsolvable word as a factor is also unsolvable. Moreover, due to the Theorem 6, v is unsolvable if it contains at least one of the patterns (Ia) (Ib), (IIa) or (IIb). Therefore, checking the solvability of a binary word can be presented as a pattern-matching problem.

The algorithm described below takes a binary word v as an input and returns true if v is solvable and false otherwise, i.e. any of the above mentioned patterns was found in v.

As the first step, we search for the patterns (Ia) and (Ib). We scan the input word from left to right comparing the sizes of the two blocks of consecutive b's between any three consecutive occurrences of a and the sizes of the two blocks of

consecutive a's between any three consecutive occurrences of b. This can be done in $O(n)$ time and $O(1)$ space.

The second step is to search for the patterns (IIa) and (IIb). It utilizes the Knuth-Morris-Pratt failure function, called also the border table (see [CLRS09]). For any position i in v, it contains the length of the longest factor u which is at the same time a proper prefix and a proper suffix of $v[1..i]$. Such a factor is called a border of $v[1..i]$.

The search for the patterns (IIa) and (IIb) is performed as follows. For any possible pair of letters $v[i..i+1] = ab$ ($v[i..i+1] = ba$ respectively), we temporarily swap $v[i]$ with $v[i+1]$ and then build the border table for the suffix of v starting at position i. After discovering a repetition $v[i..j]$ (i.e., the fact that the difference between j and the length of the border divides $j - i + 1$), we check whether it is followed by a (b, respectively) and report the occurrence of the pattern.

The border table for a single suffix of the input word v can be constructed in $O(n)$ time and $O(n)$ space (see [CLRS09]). We have to process at most $O(n)$ suffixes of v, therefore the second step and the whole algorithm runs in $O(n^2)$ time and $O(n)$ space.

word length	ABSolve	Pattern-matching alg.	Java regular expressions
1	34	22	135
2	68	76	276
3	215	230	543
4	595	608	1082
5	1710	1646	5107
6	4941	3802	17019
7	21642	9966	48985
8	70047	39403	125728
9	177239	105482	292125
10	437897	227730	660942
15	23218586	6009431	28453021
20	1198560013	145795309	1205013146

Table 3.2: Comparison of the time (in nanoseconds) of different algorithms

In the Table 3.2, we can see some experimental results of checking binary words for PN-solvability with different algorithms. Here we compare the algo-

rithm `ABSolve` described earlier with the algorithms that look for patterns (Ia), (Ib), (IIa) or (IIb) as a subsequence of the word under consideration. The number given in a corresponding entry of the table means the time (in nanoseconds) for checking all possible binary sequences of a fixed length for solvability. These data are normalised in Fig. 3.12, where one can see an average time (in nanoseconds) to check the solvability of a binary sequence of a fixed length. We can see that, while being pretty close in time for short sequences, the pattern-matching algorithm essentially overtakes the `ABSolve` algorithm for longer sequences. Both specialized methods perform better than using inbuilt regular expressions. The results for longer words are almost equal for `ABSolve` and Java regular expressions. For `ABSolve` we expect a roughly linear curve, and likely similar for Java regular expressions. For the pattern-matching approach, a further decreasing of the average time for checking a word is not expected. It is more likely to change for a (relatively slow) increasing. All the implementations are done in Java 8, and we let them run on the same machine. The data in Table 3.2, respectively in Fig. 3.12, are mean values after 10 runs of each single experiment.

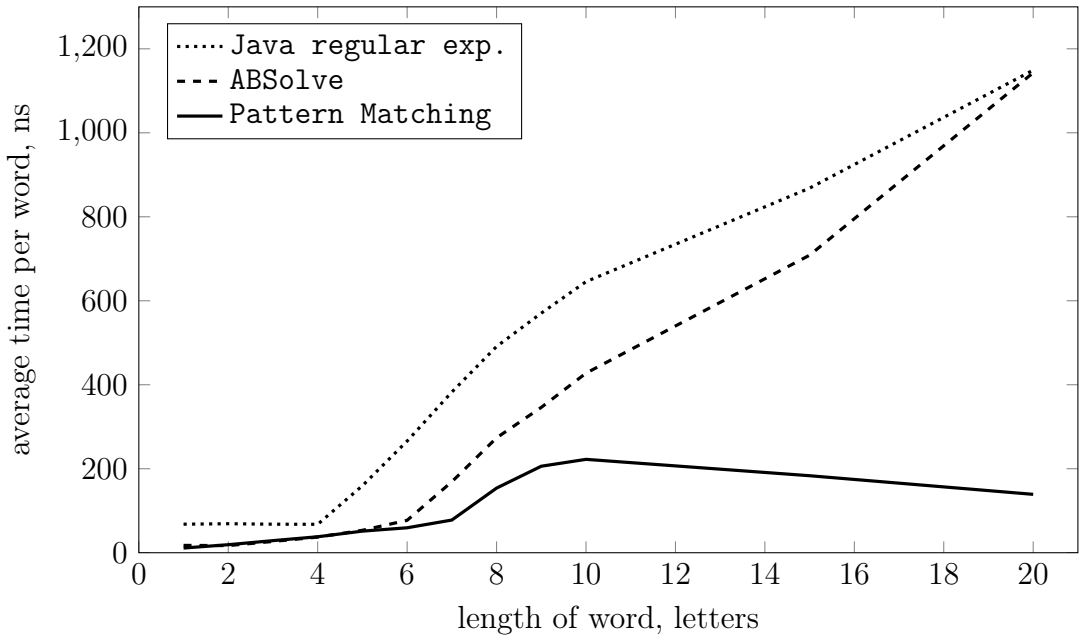


Figure 3.12: Time per word of a given length to check its unsolvability

3.7 Pre-synthesis quick fail check of lts

We have earlier established the fact that the unsolvability of a subsequence implies the unsolvability of the whole sequence under consideration. In the previous section, basing on such a criterion, we have constructed an algorithm for the quick pre-synthesis check of a word for possible unsolvability. The algorithm turned out to be faster than to try at first to synthesise a Petri net and then to fail in case of an unsolvable input transition system. This fact supports the importance of the pre-synthesis check procedures, and leads us to further development of this line of investigation. A first step on this way is presented in the current section, which establishes that unsolvability of the linear parts of a transition system implies (under some restrictions) the unsolvability of the whole input transition system.

The idea of the pre-synthesis quick fail check can be extended to a bigger classes of labelled transition systems, but with some restrictions. The following proposition demonstrates how the unsolvability of some part of a given lts can imply the unsolvability of the whole transition system.

Proposition 19. *Let* $lts = (S, T, \rightarrow, s_0)$ *be a labelled transition system and* $wts = (\{0, \ldots, n\}, \rightarrow_w, \{a, b\}, 0) = TS(w)$ *for a finite word* w *over* $\{a, b\}$*, such that* $\{a, b\} \subseteq T$*,* $\{0, \ldots, n\} \subseteq S$*, and* $\rightarrow_w = \{(i-1, t_i, i) \mid 1 \leq i \leq n \wedge t_i \in \{a, b\}\} \subseteq \rightarrow$*. Suppose that the word* $w = t_1 \ldots t_n$ *is unsolvable, and define the set of unsolvable instances of ESSP* $J = \{(j, t) \mid j \in S,\ 0 \leq j \leq n,\ t \in \{a, b\} \setminus \{t_{j+1}\} \wedge\ ESSP$ *for* (j, t) *is unsolvable for* $wts\}$*. If* $(j, t, s) \notin \rightarrow$ *for all* $(j, t) \in J$ *and* $s \in S$*, then lts is not PN-solvable.*

Proof. By contraposition. Assume there is a Petri net $N = (P, T, F, M_0)$ such that $RG(N) \cong lts$. We are aiming to transform N to a Petri net solving w, in two phases.

(i)
1. Remove every transition $t \notin \{a, b\}$ with its incident arcs. Then remove all places from P which now became isolated. Let P' be the new set of places, derived from P after this step.
2. Construct a new place p_c, having n tokens on it initially, and one output arc with unit weight to each of a and b.

After these two steps Petri net $N_1 = (P_1 = P' \cup \{p_c\}, \{a, b\}, F_1, M_0^1)$, with

$$F_1(p,t) = \begin{cases} F(p,t),\ p \in P' \\ 1,\ p = p_c \end{cases},$$

$$F_1(t,p) = \begin{cases} F(t,p),\ p \in P' \\ 0,\ p = p_c \end{cases},$$

$$M_0^1(p) = \begin{cases} M^0(p),\ p \in P' \\ n,\ p = p_c \end{cases},$$

where $p \in P_1$, $t \in \{a, b\}$, and M^0 is the marking of N, corresponding to state $0 \in S$, can be constructed. We check that N_1 allows to execute the sequence w. Place p_c enables w, since it has $n = |w|$ tokens initially. Since net N allows the execution of w at marking M^0, so does each place $p \in P'$. Hence, N_1 enables the firing of w, and, due to place p_c, finishes all executions after it. But N_1 may also have some more behaviour, which we do not want. We shall treat this in the next phase.

(ii) For each pair $(i, t, s') \in \rightarrow \setminus \rightarrow_w$, where $0 \leq i < n$ and $t \in \{a, b\}$, construct a new place q with initial marking m_q and arc weights a_q^+, a_q^-, b_q^+, b_q^-, such that q separates t at i in $w = t_1 \dots t_i \mid_i t_{i+1} \dots t_n$ and allows the execution of w. By the condition of the proposition, we have $(i, t) \notin J$, implying that such a place q always exists. Let Q be the set of all such constructed places. The Petri net $N_w = (P_w, \{a, b\}, F_w, M_0^w)$ with

$$F_w(p,t) = \begin{cases} F_1(p,t),\ p \in P_1 \\ t_p^-,\ p \in Q \end{cases},$$

$$F_w(t,p) = \begin{cases} F_1(t,p),\ p \in P_1 \\ t_p^+,\ p \in Q \end{cases},$$

$$M_0^w(p) = \begin{cases} M_1^0(p),\ p \in P_1 \\ m_p,\ p \in Q \end{cases},$$

where $p \in P_w$, $t \in \{a, b\}$, now solves w. Indeed, since $P_1 \subseteq P_w$, no new firings of transitions (no new behaviour) have been enabled in comparison

to N_1. On the other hand, for any pair (j, t) such that $t \neq t_{j+1}$ and t was enabled at state j in N_1, there is a corresponding place $q \in Q \subseteq P_w$ which forbids the firing of t at state j.

Hence, $RG(N_w) \cong wts$, contradicting the unsolvability of w. Thus, lts is not isomorphic to the reachability graph of any unlabelled Petri net. □

Remark. One cannot hope to relax the condition of the proposition. Indeed, the labelled transition system $wts = (\{0, \ldots 5\}, \rightarrow_w, \{a, b\}, 0)$ (see Fig. 3.13), which corresponds to the minimal unsolvable word $w = abbaa$, is a subsystem of $lts = (\{0, \ldots, 5, 6\}, \rightarrow, \{a, b\}, 0)$. The set of unsolvable instances of ESSP for wts is $J = \{(2, a)\}$, because a is not separable from state 2. Since $(2, a, 6) \in \rightarrow$, the condition is not satisfied for wts and lts. But lts is isomorphic to the reachability graph of the Petri net N also shown in Fig. 3.13, i.e. it is solvable.

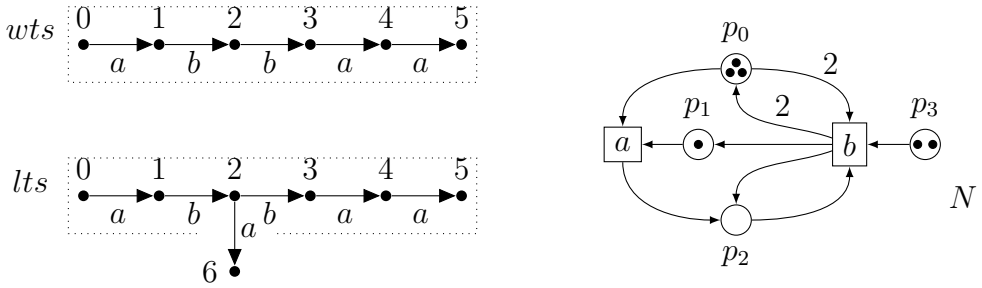


Figure 3.13: wts corresponds to unsolvable word $w = abbaa$. However lts is solvable by N.

3.8 Reversibility of muws

Besides the task of Petri net synthesis itself and the pre-synthesis checking, the results obtained so far have their reverberation in the area of reversible computation which deals with (typically local) mechanisms for undoing the effects of actions executed by a dynamic system. Such an approach has been applied, in particular, to various kinds of process calculi and event structures (see, e.g., [BB92, CL11, DK04, DK05, LMS10, PU15, PUY12, PU07]), and to a category theory based setting [DKS07].

We now use our previous results in the context of reversibility in Petri nets. A key construction we investigate amounts to adding 'reverse' versions of selected net transitions [BMP+16]. A 'straightforward' reverse simply changes the directions of arcs adjacent to a transition being reversed, and we look at synthesising a PT-net with 'reversed' behaviour given by an lts. As shown in [BKMP16], such a static modification can severely impact on the behaviour of the system, e.g., the problem of establishing whether the modified net has the same states as the original one is undecidable.

In this section we show that it is, in general, impossible to reverse a transition using its straightforward reverse without changing the state space. What is more, the situation does not change if we relax the notion of a reverse by only requiring that the effect of its execution is opposite to that of the original transition. We will also have a closer look at the possibility to reverse a word, and in particular we will try to investigate the solvability of reverses of minimal unsolvable words.

Definition 27. (transition reverses) *A* (strict) reverse *of a transition $t \in T$ in a net $N = (P, T, F, M_0)$ is a new transition $\underline{t}$ such that $F(p, \underline{t}) = F(t, p)$ and $F(\underline{t}, p) = F(p, t)$. An* effect-reverse *of a transition $t \in T$ is a new transition $\overline{t}$ such that $\mathbb{E}_p(\overline{t}) = -\mathbb{E}_p(t)$, for all places $p \in P$.*

To improve readability, we depict newly created reverses and adjacent arcs by dashed lines. For a given transition t, its strict reverse $\underline{t}$ is unique and, at the same time, it is an effect-reverse of t. However, an effect-reverse $\overline{t}$ is not necessarily a strict reverse (see Fig. 3.14).

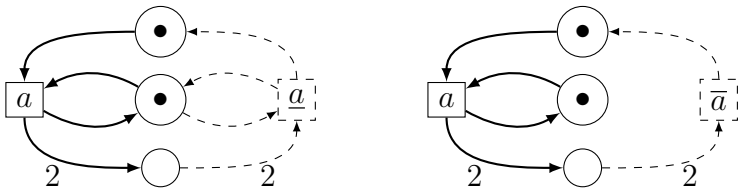


Figure 3.14: A transition a and its (strict) reverse $\underline{a}$ (lhs), and an effect-reverse $\overline{a}$, which is not a strict reverse (rhs).

The following definition explains the modification of a transition systems by adding and removing reverses of its labels.

Definition 28. (lts extension) *Let $TS = (S, T, \rightarrow, s_0)$ be a solvable lts. The* extension *of TS by reversing $t \in T$ is an lts $TS^{[+\overline{t}]} = (S, T \cup \{\overline{t}\}, \rightarrow', s_0)$ such that, for all $s_1, s_2 \in S$:*

- $(s_1, a, s_2) \in \rightarrow'$ *if* $(s_1, a, s_2) \in \rightarrow$, *for all* $a \in T$;
- $(s_1, \overline{t}, s_2) \in \rightarrow'$ *if* $(s_2, t, s_1) \in \rightarrow$.

The notion of extension can be extended to a finite set of labels $\{t_1, \ldots, t_n\} \subseteq T$. The resulting lts will be denoted by $TS^{[+\overline{t_1,\ldots,t_n}]}$.

With the next proposition we establish that, despite the inequality between the strict and an effect-reverse of the transition from the point of their definitions, when talking about solvability of the modified transition system, the existence of one of them implies the existence of another one.

Proposition 20. [BMP+16] *Let $TS = (S, T, \rightarrow, s_0)$ be a solvable lts and $a \in T$. If $TS^{[+\overline{a}]}$ is solvable then there exists its solution such that $\overline{a}$ is a strict reverse of a.*

Proof. Let us consider a net $N = (P, T \cup \{\overline{a}\}, F, M_0)$ that solves $TS^{[+\overline{a}]}$ and its transitions $a, \overline{a}$. The effects of these transitions are opposite (by the definition of $TS^{[+\overline{a}]}$). This means that, for every $p \in P$, we have $F(a, p) - F(p, a) = F(p, \overline{a}) - F(\overline{a}, p)$.

We will show that $N' = (P, T \cup \{\overline{a}\}, F', M_0)$ is also a solution for $TS^{[+\overline{a}]}$, where: (i) $F'(a, p) = F'(p, \overline{a}) = \max(F(a, p), F(p, \overline{a}))$ and $F'(p, a) = F'(\overline{a}, p) = \max(F(p, a), F(\overline{a}, p))$, for every $p \in P$; and (ii) $F'(b, p) = F(b, p)$ and $F'(p, b) = F(p, b)$, for all $p \in P$ and $b \notin \{a, \overline{a}\}$.

By the monotonicity of Petri nets and their firing rule, it is enough to prove that, for every reachable marking M of the net N and every transition $x \in T$, if x is enabled at M in N then x is enabled at M in N'.

Indeed, suppose that there exists a marking M at which x is enabled in N but not in N'. By the definition of F', $x = a$ or $x = \overline{a}$. Let us suppose that $x = a$. Then we have $M[a\rangle M'$, and there exists a place $p \in P$ such that $F(p, a) \leq M(p)$ but $F'(p, a) > M(p)$. By the definition of F', we get $F'(p, a) = F'(\overline{a}, p) = F(\overline{a}, p) > M(p) \geq F(p, a)$. Since $F(a, p) - F(p, a) = F(p, \overline{a}) - F(\overline{a}, p)$, we also get $F'(a, p) = F(p, \overline{a}) = F'(p, \overline{a})$.

Hence, we know that $M'[\overline{a}\rangle M$ both in N and in N'. This means that $M(p) \geq F(\overline{a}, p) = F'(\overline{a}, p) = F'(a, p)$, which yields a contradiction and proves that a is enabled at M in N'.

The case $x = \overline{a}$ is similar. □

Petri net N_1 in Fig. 3.15 without the dashed part. It solves the word *bbabab*, and so its reachability graph is isomorphic to TS_1. TS_2 obtained from TS_1 by adding a reverse for transition b is solvable by N_1 with the dashed part. Note that, in N_1, $\overline{b}$ is a strict reverse of b. Similarly, we may reverse a in TS_1, obtaining TS_3 in Fig. 3.16. This lts is solvable by the net N_2 with the dashed part.

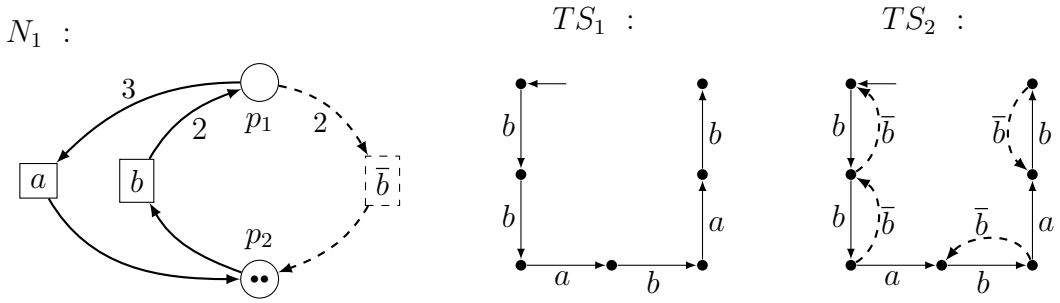


Figure 3.15: Adding a reverse $\overline{b}$ in $TS_2 = TS_1^{[+\overline{b}]}$ does not violate solvability.

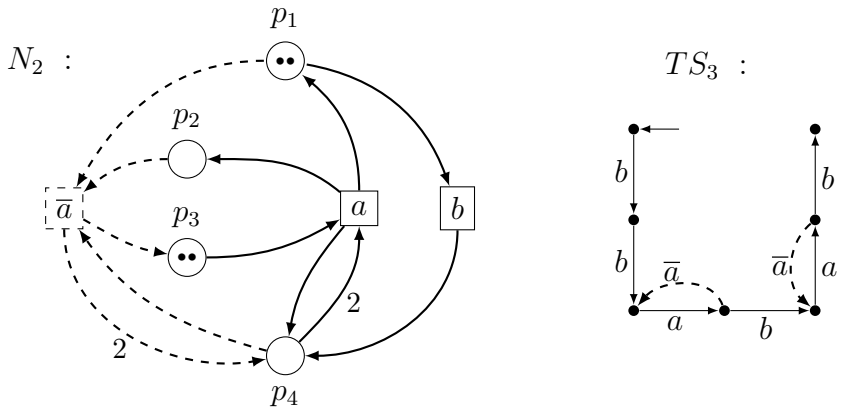


Figure 3.16: $TS_3 = TS_1^{[+\overline{a}]}$ is solvable by N_2.

If adding reverses for two labels yields a solvable lts, then the lts containing both reverses is also solvable.

Proposition 21. [BMP+16] *Let $TS = (S, T, \rightarrow, s_0)$ be solvable and $a \neq b \in T$. If both $TS^{[+\overline{a}]}$ and $TS^{[+\overline{b}]}$ are solvable, then so is $TS^{[+\overline{a,b}]}$.*

Proof. Let $N^a = (P^a, T^a, F^a, M_0^a)$ be a solution for $TS^{[+\overline{a}]}$ with an added isolated transition $\overline{b}$. Thus, $\overline{b}$ is always enabled and its execution does not affect the marking. We define $N^b = (P^b, T^b, F^b, M_0^b)$ similarly.

Without loss of generality, we assume that the sets of places of N^a and N^b are disjoint, and that their sets of transitions are the same. Thus, we can define a net N as a synchronisation on transitions of N^a and N^b, with an additional modification: (i) for every $p \in P^b$, $F(p, \overline{a}) = F(a, p)$ and $F(\overline{a}, p) = F(p, a)$; and (ii) for every $p \in P^a$, $F(p, \overline{b}) = F(b, p)$ and $F(\overline{b}, p) = F(p, b)$.

Any firing sequence of N not containing $\overline{a}$ or $\overline{b}$ is not affected and leads to the same projected markings as in N^a and N^b. Note that the reachability graph of N has almost the same structure as in case of N^a and N^b, with all the transitions except $\overline{a}$ and $\overline{b}$ behaving in the same way as before synchronisation.

Assume now that $M^a[\overline{a}\rangle M'^a$ in N^a. By the above argument, we can look at the corresponding markings M and M' of N, and have to show that $M[\overline{a}\rangle M'$. By the definition of $TS^{[+\overline{a}]}$, we know that $M'^a[a\rangle M^a$ and, by the above argument, $M'[a\rangle M$. Thus, by the firing rule, we have that for all places p, $M(p) \geq F(a, p)$. By the construction, this means that $M(p) \geq F(p, \overline{a})$, and so $M[\overline{a}\rangle$. By $F(a, p) - F(p, a) = -(F(\overline{a}, p) - F(p, \overline{a}))$ and the firing rule, we derive $M[\overline{a}\rangle M'$.

The same argument applies for $\overline{b}$.

□

It follows from the proof of the last result that a solution for $TS^{[+\overline{t,u}]}$ can be obtained by synchronising any solutions for $TS^{[+\overline{t}]}$ and $TS^{[+\overline{u}]}$ with disjoint sets of places on the (common) transitions in T, and then making $\overline{t}$ a strict reverse w.r.t. the solution for $TS^{[+\overline{u}]}$, and making $\overline{u}$ a strict reverse w.r.t. the solution for $TS^{[+\overline{t}]}$.

Using Proposition 21 and starting from two solutions, N_1 for $TS_2 = TS_1^{[+\overline{a}]}$ and N_2 for $TS_3 = TS_1^{[+\overline{b}]}$, we can construct a solution N_3 for $TS_4 = TS_1^{[+\overline{a,b}]}$ depicted in Fig. 3.17.

A sequence w written from right to left will be called the *mirror image* (or simply *mirror*) of a word w and denoted as w^R. The next proposition establishes that mirrors of minimal unsolvable words are always solvable.

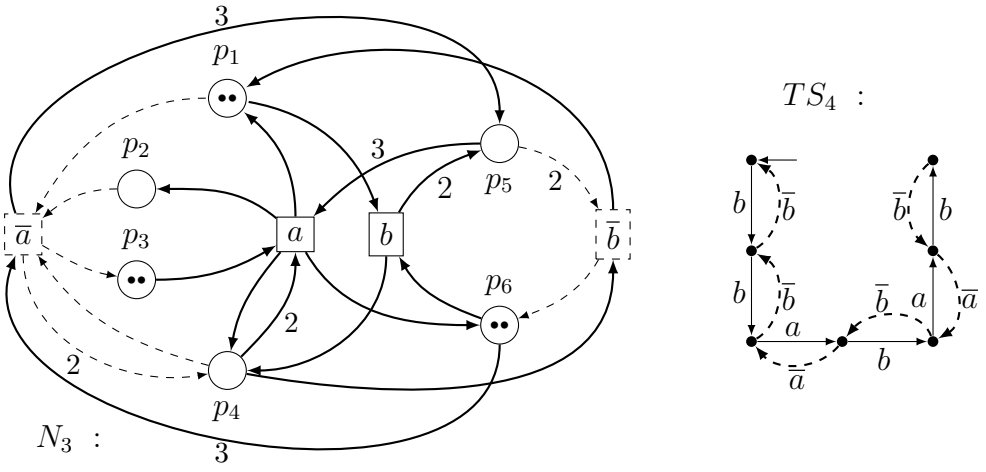


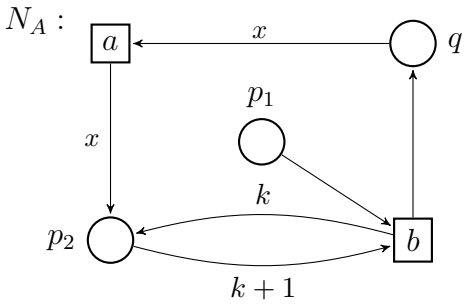
Figure 3.17: N_3 solving $TS_4 = TS_1^{[+\overline{a,b}]}$ derived by synchronising the transitions of N_1 in Fig. 3.15 and N_2 in Fig. 3.16.

Proposition 22. [BMP+16] *Let $w \in \{a,b\}^*$ be a minimal unsolvable word. Then $TS(w^R)$ is solvable.*

Proof. Let w be a minimal unsolvable word. According to Theorem 6, we can consider three subcases (the others are symmetrical modulo swapping a and b):

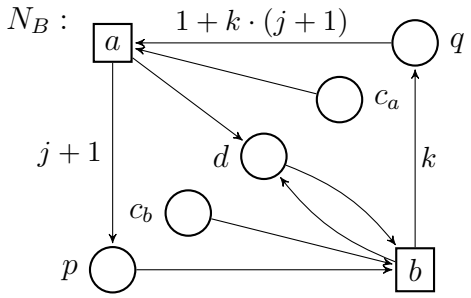
Case 1: $w = ab^x ab^{x-k} a$, then $w^R = ab^{x-k} ab^x a$ and $3 \leq k \leq x$.

Petri net N_A depicted in Figure 3.18 is a possible solution for this word. Indeed, initial marking M_0 allows only firing of transition a. In case $k = x$, this single firing does not enable b, and place q has enough tokens for one more occurrence of a. This second a consumes all the tokens from place q, disabling a, and enables b to fire x times. Transition a remains disabled until b has occurred x times in a row, and can be fired only once after. This finishes the execution of N_A for the case $x = k$. If $x > k$, place q enables a to fire only once at the beginning. This first occurrence of a enables b to fire $x-k$ times due to the side-condition place p_2. After the execution of block b^{x-k}, a becomes enabled to occur once again. This occurrence brings x tokens on p_2, and together with the tokens that have already been there, it allows for firing block b^x. Then, due to place p_1, only single a can occur, which finishes the execution of N_A.

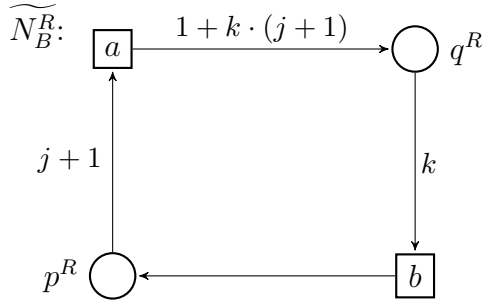


$$M_0 \begin{pmatrix} p_1 \\ p_2 \\ q \end{pmatrix} = \begin{pmatrix} 2x-k \\ 0 \\ x+k \end{pmatrix}$$

Figure 3.18: N_A solves $ab^{x-k}ab^xa$.



$$M_0 \begin{pmatrix} p \\ q \\ d \\ c_a \\ c_b \end{pmatrix} = \begin{pmatrix} 1 \\ 1+k\cdot(j+1) \\ 0 \\ k+1 \\ (j+1)\cdot(k+1) \end{pmatrix}$$



$$M_0 \begin{pmatrix} p^R \\ q^R \end{pmatrix} = \begin{pmatrix} 1 \\ k\cdot j \end{pmatrix}$$

Figure 3.19: N_B solves the prefix w_1. $\widetilde{N_B^R}$ defines the internal execution of w_1^R.

Consider the case $w = abb^j(bab^j)^k a$ (for its extensions the proof is analogous). Due to the minimality, the maximal proper prefix $w_1 = abb^j(bab^j)^k$ of w is solvable, and the Petri net N_B in Fig. 3.19 is a possible solution for this prefix. Let us consider the *inversed* core part $\widetilde{N_B^R}$ of this net, which is derived by changing the directions of arcs between places and transitions, with the final marking of N_B as the initial marking. In Lemma 14 it has been shown that the places p and q completely define the order of execution of a and b inside w_1. Transition a is not enabled initially in $\widetilde{N_B^R}$. Since the markings of places p^R and q^R in this net now repeat all reachable markings of N_B (restricted to places p and q) in an inversed order while firing w_1^R, this net allows the execution of w_1^R. We can add counter places c_a^R and c_b^R for a and b with initial markings $k+1$ and $(j+1)(k+1)$, respectively. These places will stop executions after the firing of w_1^R. We now show that the resulting net $\overline{N_B}$ (see Fig. 3.20) does not have any additional behaviour except w_1^R. Assume that for some marking $\overline{M}$ reachable in $\overline{N_B}$, $\overline{M}[a\rangle$ and $\overline{M}[b\rangle$. Let M be the marking of N_B corresponding to $\overline{M}$ and consider the case when M is reached in N_B by firing transition a from some other marking M'. Due to place c_b^R, $\overline{M}$ is not the marking before the very last a in w_1^R. Hence, as a does not occur more than once in a row, there is a marking M'' such that $M''[b\rangle M'[a\rangle M$ in N_B. From $\overline{M}[b\rangle$ we have $M(q) = \overline{M}(q^R) \geq k$. This implies $M'(q) \geq k + 1 + k \cdot (j+1)$. Since $M''[b\rangle M'$, $M''(q) \geq 1 + k \cdot (j+1)$. Therefore, $M''[a\rangle$ which contradicts $M''[b\rangle$. Hence, there is no marking M' reachable in N_B, such that $M'[a\rangle M$. The case when M is reached from M' through firing of transition b is similar.

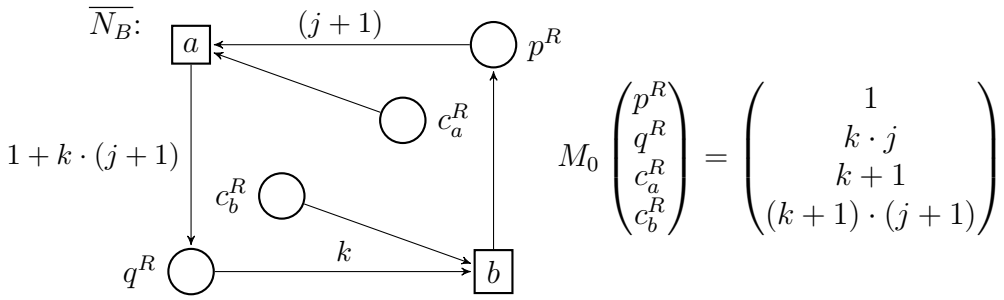


Figure 3.20: $\overline{N_B}$ solves the reversed prefix w_1^R.

Thus, w_1^R is solvable. By Proposition 3 we can 'unfire' one a in N_B^R. In order to do this we increase arc weights between q^R and b by $1+k\cdot(j+1)$, and change initial

markings of c_a^R and p^R to $k+2$ and $j+2$, correspondingly. The net obtained after such a transformation allows to execute a initially and has the same behaviour as N_B^R afterwards, thus solves w^R.

Case 3: $w = bab^j(abb^j)^k b$ with $j \geq 0$, $k \geq 1$, or one of its extensions.
This case is analogous to case 2. Petri net N_C solving w^R is depicted in Fig. 3.21.

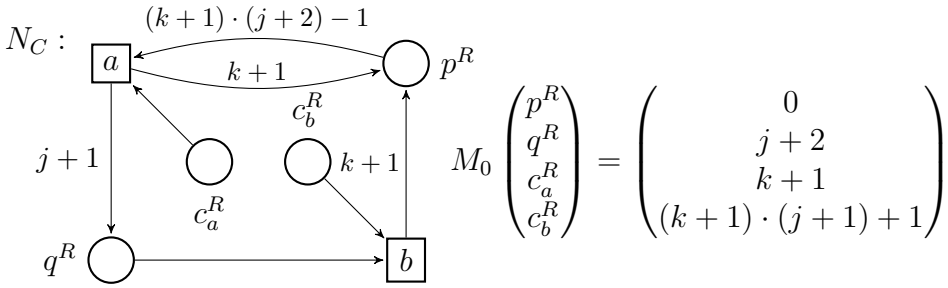


Figure 3.21: N_C solves $(bab^j(abb^j)^k b)^R$.

□

Due to Propositions 21 and 22, reversing both transitions in the mirror image w^R of some minimal unsolvable word w over $\{a, b\}$ yields the solvability of w, which is a contradiction. Hence, the following corollary holds:

Corollary 2. *Let $w \in \{a, b\}^*$ be a minimal unsolvable word and $TS = TS(w^R)$. Then $TS^{[+\bar{a}]}$ or $TS^{[+\bar{b}]}$ is unsolvable.*

As we have seen, the unsolvability of sequences is not an isolated issue: it has its implications in a wider context. In particular, it allows to talk about the solvability of complicated transition systems which contain a linear one as a part. Besides, in the current section we have discussed a less obvious continuation of this line of research – reversible computations. In this area we also have seen a possibility for applying existing results in order to detect the impossibility of reversing for some labels in the transition system without its solvability.

3.9 Summary

The class of minimal unsolvable binary words was studied in the present chapter. Among the main results of the investigation, we can recall an introduced classi-

fication for a complete enumeration of the muws (first initiated in [BBE+16] and completed in [EBMP16]), and a presented algorithm for a quick check of the unsolvability of a given binary sequence. Besides some repercussions of the characterisation in the field of reversible computations were mentioned (see also [BMP+16]). The classification of muws partitions the entire class into three main groups. Non-extendable minimal unsolvable words form the first main group. These words satisfy the patterns ab^xab^ya and ba^xba^yb with $x > y + 2$, $y \geq 2$. The other two main groups, namely base extendable and extendable muws, follow the patterns $abw(baw)^ka$ and $baw(abw)^kb$ with $k \geq 1$. All of these forms can be generalised by the pattern $(ab\alpha)b^*(ba\alpha)^+a$ ([BBE+15]), which was first discussed in the previous chapter, and whose presence as a subword is now proved to be a characterisation of the (possibly non-minimal) unsolvability of a binary sequence. Base extendable words can serve as origins from which, with the use of the extension operation, the class of extendable muws can be derived. The classification of muws suggests a pattern-matching algorithm which allows to detect an unsolvable synthesis input without initiating the synthesis process. The algorithm demonstrates a better runtime in comparison with the earlier introduced algorithm `ABSolve`. This exploiting of the characterisation of unsolvable state spaces (which are binary words here) via general patterns continues the discussion on the pre-synthesis idea, which was initiated before.

Chapter 4

Synthesis of Petri nets from finite languages

Throughout the previous chapters we have established a number of necessary and sufficient conditions for the solvability of finite sequences with Petri nets. These conditions have different forms: some of them can be formulated as a pattern-matching, while the other ones can be expressed as graph-theoretical conditions between subsequences of the whole sequences. We have also seen that the solvability of this restricted class can find applications in some other (not directly related) areas. In this chapter, which partly based on [EW17] by the author and a co-author, we aim to broaden our view, to look at finite alphabets and finite languages over them, and to relax our restriction to the binary case. The results and the intuition that we gained from the previous chapters will be of use in this broader area. We begin with an observation of which of our earlier results can be extended and to what border, in particular for finite sequences, and continue with cyclic structures and finite languages. Besides, we will also discuss why (possible) extending the counting criterion is not too promising in terms of the efficiency of synthesis. In the following, we will present a characterisation of a bigger class of solvable finite transition systems with the synthesis algorithms based on this characterisation.

4.1 Synthesisability of words over finite alphabets

While sufficient conditions for Petri net synthesisability are scarce in the literature, the ones provided earlier (see Theorem 3 in Section 2.6 and Corollary 1 in Section 3.5) for binary sequences turned out to be strong enough to use them for sequences over arbitrary finite alphabets. Here we present a sufficient condition for the synthesisability of finite sequences over an arbitrary finite alphabet.

Definition 29. *Let T be a finite alphabet, and $w \in T^*$ a finite sequence over T. For a subset $T' \subseteq T$, a sequence $w' \in T'^*$ is called a projection of w to T' if $w' = Pr_{T'}(w)$, where $Pr_{T'}$ is defined as follows:*

$$Pr_{T'}(u) = \begin{cases} \epsilon, & \text{for } u = t \in T \setminus T', \\ t, & \text{for } u = t \in T', \\ Pr_{T'}(t)Pr_{T'}(v), & \text{for } u = tv,\ t \in T,\ v \in T^*. \end{cases}$$

Proposition 23. (sufficient condition for solvability of arbitrary sequences) *Let w be a finite sequence over an alphabet T. If $Pr_{\{t_1,t_2\}}(w)$ is solvable for all $t_1, t_2 \in T$, then w is solvable.*

Proof. Let w be a sequence over T, and let all projections of w to binary subalphabets of T be solvable. Consider $a, b \in T$ such that $a \neq b$ and

$$w = \ldots\ |_s b\ u\ |_{s'} a\ \ldots \quad \wedge \quad \Psi(u)(a) + \Psi(u)(b) = 0.$$

In order to construct a Petri net solving w, we have to find a place that separates a at state s.

Using the isomorphism between sequences and labelled transition systems, and Definition 29, for $T' \subseteq T$ we can define a mapping $Pr_{T',w} : S \to S'$ between sets of states of $TS(w) = (S, T, \to, s_0)$ and $TS(Pr_{T'}(w)) = (S', T', \to', s'_0)$ as follows:

$$Pr_{T',w}(s_i) = s_{\max\{j \leq i | t_j \in T'\}}, \text{ for } s_i \in S$$

If $Pr_{\{a,b\}}(w)$ is solvable with some Petri net, then there is a place p such that p disables a at $Pr_{\{a,b\},w}(s)$. Place p is only affected by transitions a and b. Hence it

disables a at s and enables a at all states in $\{s'' \in S \mid Pr_{\{a,b\},w}(s'') = Pr_{\{a,b\},w}(s)\}$, i.e., p solves ESSP for $\neg s[a\rangle$. Similarly, we can construct a solution for every instance of ESSP in w, thus w is solvable. □

Proposition 23 implicitly suggests an approach for constructing of a Petri net solving a sequence over a 'big' alphabet from the solutions of its subsequences. In order to do this we have to merge Petri nets solving projections by transition pairs, and retain all places, markings and arcs. The following example demonstrates this approach.

Example 20. *We examine the sequence $w = abacba$ over the three-letter alphabet $T = \{a, b, c\}$. Since T has 3 binary subalphabets, there are 3 corresponding binary projections of w:*

$$Pr_{\{a,b\}}(w) = ababa, \quad Pr_{\{a,c\}}(w) = aaca, \quad Pr_{\{b,c\}}(w) = bcb.$$

All three projections are solvable – their possible solutions are Petri nets N_{ab}, N_{ac} and N_{bc}, respectively (see Fig. 4.1). In order to construct a solution for the

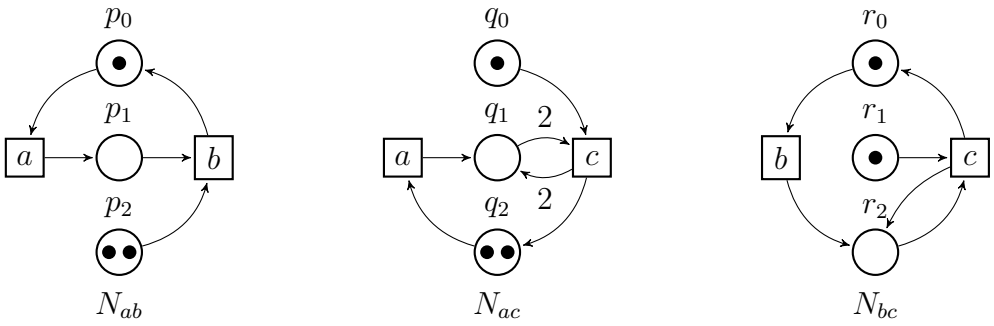


Figure 4.1: Petri nets N_{ab}, N_{ac}, N_{bc} are solutions for *ababa*, *aaca*, *bcb*, respectively.

sequence w, we can compose all the solutions for its binary projections by merging them through the transitions. The Petri net N in Fig. 4.2 – the result of this composition – has the reachability graph isomorphic to $TS(w)$, and hence solves $w = abacba$. Let us notice that places r_1 and q_0 in the net N duplicate each other: both restrict the total number of firings of c. Hence one of these places is redundant.

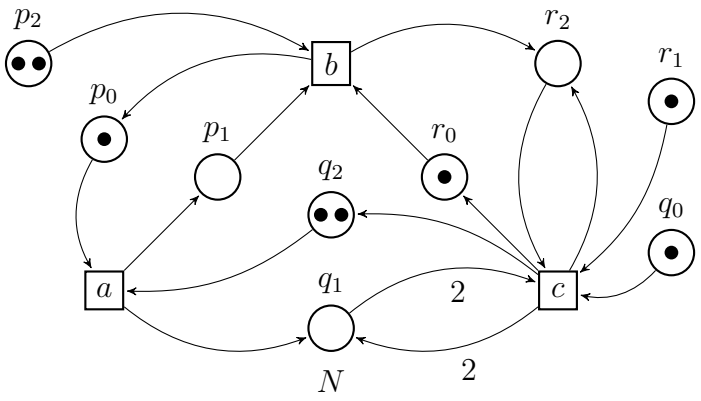


Figure 4.2: Petri net N, solving *abacba*, is composed from N_{ab}, N_{ac} and N_{bc} (see Fig. 4.1) by merging them via transitions.

The next example shows that the condition from Proposition 23 is not necessary for the solvability of a sequence. Thus, the solvability of binary projections of a given sequence can be used in a pre-synthesis phase, but it is not a necessary condition for a successful synthesis of a Petri net.

Example 21. *Consider the sequence $w = abcbaa$ over the alphabet $T = \{a, b, c\}$. The sequence is solvable, since the Petri net N depicted in Fig. 21 has a reachability graph isomorphic to $TS(w)$. Nevertheless, the projection $Pr_{\{a,b\}}(w) = abbaa$ is known to be a (minimal as well as lenght-minimal) unsolvable word.*

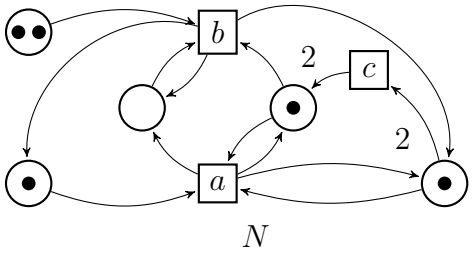


Figure 4.3: Petri net N has a reachability graph isomorphic to $TS(abcbaa)$.

4.2 Generalising the counting condition is complicated

In this section, we explain why the counting criterion is not likely to yield substantially better synthesis algorithms than region-based synthesis, even if pre-synthesis is taken into account, subject to alphabets consisting of 3 and more letters. We start our consideration with several examples and then try to generalise what they suggest.

The sequence $abcbaca$ is unsolvable which can be seen as follows. Since $\Psi(abc) = \Psi(bac)$, both words must have the same effect as firing sequences. Consider the separation point y in $|_x abc|_y bac|_z a$: a cannot fire at y, but it can fire at x (the start of abc) and at z (after bac). So there must be a critical place p with enough tokens on it to fire a at x and z, but not enough at y. This contradicts the Parikh equivalence of abc and bac, which forces both to have the same effect on p – either both positive or both negative or both with no effect.

Let us look at another unsolvable sequence $abcbaccbaa$. Here, $2 \cdot \Psi(abc) = \Psi(baccba)$, so the effect of both words has the same sign. Again, in $|_x abc|_y baccba|_z a$ the number of tokens on a critical place must be lower at y than at x and at z, which is impossible. Therefore, a separation failure occurs at y, i.e. there exists no Petri net that has $abcbaccbaa$ as a firing sequence and disables a at y at the same time. This even holds for Petri nets that can fire more than one word.

The sequence $|_x accb|_y c|_s ab|_z a$ is also unsolvable. At y, the number of tokens on some critical place p is too low to fire a, but at s and z it is high enough. Assume there would be a net solving this word. Remove all non-critical places (with enough tokens at y to allow a). The remaining net can still fire the word and can still not fire a at y. Since c adds enough tokens to all critical places to enable a (after y), we can add another c to our firing sequence at this point, getting the word $|_x accb|_y ccab|_z a$. This word can be fired in our reduced net, and at y the transition a can still not fire. This contradicts the above paragraph telling us that such a Petri net cannot exist. Therefore, there is a separation failure at y in $|_x accb|_y c|_s ab|_z a$ also, i.e. the word is unsolvable.

So, if we have $|_x av|_y w_1|_{z_1} a w_2|_{z_2} a \dots w_n|_{z_n} a$ such that

$$\Psi(av) = \sum_{i=1}^{n} k_i \cdot \Psi(w_1 \dots a w_i)$$

with all rational $k_i \geq 0$, then the word is unsolvable since all the words $w_1 \dots a w_i$ increase the token count on all critical places while av with the same relative occurrence of transitions decreases the token count.

The sequence $|_x abca|_y caaab|_z a$ is unsolvable. We cannot obtain the same relative Parikh vector on both sides of y, i.e. $\Psi(abca) = k_1 \cdot \Psi(c) + k_2 \cdot \Psi(ca) + k_3 \cdot \Psi(caa) + k_4 \cdot \Psi(caaab)$ cannot be solved. There is a different argument, though: the transition a must have a negative effect on all critical places, since a fires directly before y. On the other hand, the effect of $abca$ is also negative while the effect of $caaab$ is positive, meaning that $\Psi(caaab) - \Psi(abca) = \Psi(a)$ also must have a positive effect on the critical places, a contradiction.

Let us now examine the unsolvable sequence $|_x accbc|_y ccab|_z a$. Since $accbc$ has a negative effect on the critical places and $ccab$ has a positive effect, we conclude from $\Psi(accbc) - \Psi(ccab) = \Psi(c)$ that c has a negative effect. On the other hand, cc enables a after y, showing that c must have a positive effect, a contradiction.

Finally, the sequence $|_x accab|_y cbaa|_z a$ is also unsolvable. This case is a bit more complicated. Due to $\Psi(accab) - \Psi(cbaa) = \Psi(c)$ we know that c has a negative effect on the critical places. But since $\Psi(cba) - \Psi(ab) = \Psi(c)$, where cba has a positive effect and ab (in front of y) has a negative effect, c itself must also have a positive effect, a contradiction.

With these types of argument, all minimal unsolvable words over $\{a, b, c\}$ up to at least length 10 can be shown unsolvable. The *general argument* (that also holds in the first examples) is that if we find a transition [sequence] which must have a positive and a negative effect for the same critical place(s), then we have found a separation failure and the word is unsolvable. This sounds a lot like a region approach.

Note that the two-letter case is a special case of our general argument of 'a positive and a negative effect' for unsolvable words. In a word $|_x av|_y bw|_z a$ with $v, w \in \{a, b\}^*$, b must have a positive effect, since a is disabled at y but not at z, and it cannot enable itself – so b must do it. If $\#_a(av) \cdot \#_b(bw) \leq \#_b(av) \cdot \#_a(bw)$

– i.e. the condition for unsolvability of words over two letters – we can find factors $j, k > 0$ such that $j \cdot \#_a(av) = k \cdot \#_a(bw)$ and $j \cdot \#_b(av) \geq k \cdot \#_b(bw)$, giving a a negative effect on the critical place(s).

Let us now think about the following *Question Q*: If a positive and a negative effect on the same place does not follow from any logical deduction, is the word then solvable? This could be possible. Any quantitative (not sign-based) relation on token effects might be achievable by multiplying all arc weights and token counts by some high number and then doing small changes somewhere in the net.

If the answer to Question Q is "yes", then we can also 'easily' detect whether a word is minimal unsolvable, and we also know why minimal unsolvable words must start and end with the same letter and why no separation failure against the other letters occurs anywhere in such words.

If a minimal unsolvable word starting and ending with a would contain a separation failure against another letter (say b), then the contradictory argument could not use the whole word, i.e. we could at least cut off a prefix up to (but excluding) the first b and a postfix after the last b and still have an unsolvable word.

Example 22. *Consider the word abccabbcabaa. This word has, according to* `APT`*, two separation failures, one between the two c's, one between the two b's. So, for*

$$|_x abc|_y c|_{z_1} abbc|_{z_2} ab|_{z_3} a|_{z_4} a,$$

in $\Psi(z_1 - y) = \Psi(c)$*, transition c has a positive effect on the critical places at y while in* $\Psi(z_4 - y) - 3 \cdot \Psi(y - x) = -\Psi(c)$ *it has a negative effect on the same places. Since the latter calculation uses the whole word (from x to the final a which defines* z_4*), cutting off any letters at the start or end would invalidate this argument. It also seems to be the only way to obtain the necessary contradiction.*

If we find another separation failure at which we do not need the whole word for our contradiction, we could cut off letters at the start or end and the reduced word would remain unsolvable. So it would not be minimal. Let us check the other separation failure:

$$|_{x_1} abcc|_{x_2} ab|_y bc|_{z_1} ab|_{z_2} a|_{z_3} a.$$

Here, we can argue for a negative effect for a: $2 \cdot \Psi(z_1 - y) - \Psi(y - x_1) = -2 \cdot \Psi(a)$,

and for c: $\Psi(z_3 - y) - \Psi(y - x_1) = \Psi(c)$*, as well as positive effects for b:* $2 \cdot \Psi(z_2 - y) - \Psi(y - x_1) = 2 \cdot \Psi(b)$*, and for c:* $\Psi(z_3 - y) - 2 \cdot \Psi(y - x_2) = \Psi(c)$*. The contradiction again comes from transition c and for that argument we need again the whole word. There seems to be no shorter argument. Since there are no other separation failures, the word must be minimal unsolvable.*

We never need to show that a transition sequence has a positive and a negative effect at the same time. If we have

$$\sum_i j_i \cdot \Psi(z_i - y) - \sum_i k_i \cdot \Psi(y - x_i) = \Psi(\tau)$$

and

$$\sum_i j_i' \cdot \Psi(z_i - y) - \sum_i k_i' \Psi(y - x_i) = -\Psi(\tau)$$

then

$$\sum_i (j_i + j_i') \cdot \Psi(z_i - y) = \sum_i (k_i + k_i') \cdot \Psi(y - x_i).$$

It is sufficient to sum up the same Parikh vector in both the (allowed) prefixes and postfixes at y to get a contradiction and thus a separation failure.

Revisit $abca|_y caaaba$ and mark the start and end points of the subwords ending and starting at y, respectively:

$$|_{x_1} abc|_{x_2} a|_y c|_{z_1} a|_{z_2} a|_{z_3} ab|_{z_4} a.$$

Now $\Psi(z_4 - y) = \Psi(y - x_1) + \Psi(y - x_2)$, which contradicts *caaab* having a positive and *abca* and a having a negative effect on the critical places at y.

We can show the following ring of conclusions:

Take some word $u_1 u_2 \dots u_m | v_1 v_2 \dots v_n a$ over Σ, such that each subword u_i and v_j except v_1 starts with an a and no other subwords do. Assume there is a separation failure against a for the ESSP in front of v_1.

Then, from the theory of regions, we know that the linear system

$$\begin{pmatrix} \#_a(u_1 \dots u_m) & \#_b(u_1 \dots u_m) & \#_c(u_1 \dots u_m) & \dots \\ \#_a(u_2 \dots u_m) & \#_b(u_2 \dots u_m) & \#_c(u_2 \dots u_m) & \dots \\ \vdots & \vdots & \vdots & \\ \#_a(u_m) & \#_b(u_m) & \#_c(u_m) & \dots \\ -\#_a(v_1) & -\#_b(v_1) & -\#_c(v_1) & \dots \\ -\#_a(v_1 v_2) & -\#_b(v_1 v_2) & -\#_c(v_1 v_2) & \dots \\ \vdots & \vdots & \vdots & \\ -\#_a(v_1 \dots v_n) & -\#_b(v_1 \dots v_n) & -\#_c(v_1 \dots v_n) & \dots \end{pmatrix} \begin{pmatrix} \mathbb{E}_a \\ \mathbb{E}_b \\ \mathbb{E}_c \\ \vdots \end{pmatrix} < 0$$

is unsolvable. The subwords starting with a and ending at the separation point must reduce the number of tokens on the place to be constructed for the ESSP, while the subwords starting at the separation point and ending before an a must increase this token count (expressed by the negative coefficients).

Then, by Gordan's Theorem [Dan63], the following dual linear system must have a non-zero, non-negative solution:

$$\begin{pmatrix} \#_a(u_1 \dots u_m) & \dots & \#_a(u_m) & -\#_a(v_1) & \dots & -\#_a(v_1 \dots v_n) \\ \#_b(u_1 \dots u_m) & \dots & \#_b(u_m) & -\#_b(v_1) & \dots & -\#_b(v_1 \dots v_n) \\ \#_c(u_1 \dots u_m) & \dots & \#_c(u_m) & -\#_c(v_1) & \dots & -\#_c(v_1 \dots v_n) \\ \vdots & & \vdots & \vdots & & \vdots \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_m \\ y_1 \\ \vdots \\ y_n \end{pmatrix} = 0$$

We can rewrite this as

$$\begin{pmatrix} \#_a(u_1 \dots u_m) & \dots & \#_a(u_m) \\ \#_b(u_1 \dots u_m) & \dots & \#_b(u_m) \\ \#_c(u_1 \dots u_m) & \dots & \#_c(u_m) \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} \#_a(v_1) & \dots & \#_a(v_1 \dots v_n) \\ \#_b(v_1) & \dots & \#_b(v_1 \dots v_n) \\ \#_c(v_1) & \dots & \#_c(v_1 \dots v_n) \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

or, using Parikh vectors, even shorter as

$$\begin{pmatrix} \Psi(u_1 \dots u_m) & \dots & \Psi(u_m) \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} \Psi(v_1) & \dots & \Psi(v_1 \dots v_n) \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

We can interpret this as the existence of a sequence σ whose Parikh vector can be expressed as a linear combination of the u_i's (always ending at the separation point) as well as of the v_j's (starting from there), i.e.

$$\Psi(\sigma) = \sum_i \Psi(u_i \dots u_m) \cdot x_i \quad \text{and} \quad \Psi(\sigma) = \sum_j \Psi(v_1 \dots v_j) \cdot y_j.$$

This sequence can be arbitrarily long (consider $aw|waa$ with $w \in (\Sigma \setminus \{a\})^+$) for which the only solution to the linear system is $x_1 = y_2$ and $y_1 = 0$).

The effect of this sequence on the place p to be constructed for the ESSP must be negative ($\mathbb{E}_p(\sigma) < 0$)) and positive ($\mathbb{E}_p(\sigma) > 0$) at the same time. We conclude that there must be a separation failure for this ESSP.

This not only is a positive solution to Question Q, it also shows that a generalised counting condition for words over three or more letters, if the one were found, would be derivable from the theory of regions. In other words, one cannot hope for a significant enhancement of the synthesis efficiency in comparison with the region based approach.

4.3 Generalised cycles of lts

In the previous section, we have argued that the counting criterion, which is a convenient tool when talking about the solvability of the class of binary sequences, does not seem to offer much advantages in the case of more than two letters. In this section, we will extend our investigation in another respect: we analyse the solvability of finite sets of sequences (languages) while keeping the condition for the transition systems representing a language to have at most two labels.

Definition 30. *For state s and s' of a transition system, $s \overset{t}{\rightsquigarrow} s'$ is used as an abbreviation for ($s \overset{t}{\rightarrow} s' \vee s \overset{t}{\leftarrow} s'$) and called a* generalised edge *or a* g-edge *(in the underlying undirected graph). A* path *(*g-path*) $\sigma \in T^*$ from s to s', written as*

$s \xrightarrow{\sigma} s'$ ($s \overset{\sigma}{\rightsquigarrow} s'$), is given inductively by $s = s'$ for the empty word $\sigma = \epsilon$ and by $\exists s'' \in S$: $s \xrightarrow{w} s'' \xrightarrow{t} s'$ ($s \overset{w}{\rightsquigarrow} s'' \overset{t}{\rightsquigarrow} s'$) for $\sigma = wt$ with $w \in T^*$ and $t \in T$. A path $s \xrightarrow{\sigma} s'$ (g-path $s \overset{\sigma}{\rightsquigarrow} s'$) is a cycle (g-cycle) *if and only if $s = s'$.*

For the sake of convenience we will consider undirected paths within lts. Parikh vectors can be generalised for them by counting arrows negatively if they point in backward direction. We will use the notation $\overleftarrow{a}$ to indicate that label a occurs in backward direction in a path.

Definition 31. *A labelled transition system is called* general-cycle-neutral *(**gc0**) if for every general cycle $\sigma \in T^*$: $\Psi(\sigma) = \mathbf{0}$, $\Psi(\sigma)$ being the Parikh vector of σ.*

Example 23. *Transition system TS_1 on the left of Fig. 4.4 has a cycle ab at its initial state with Parikh vector $\Psi(ab) = (1, 1)$. For the same path in backward direction $\overleftarrow{b}\,\overleftarrow{a}$, the Parikh vector is $\Psi(\overleftarrow{b}\,\overleftarrow{a}) = (-1, -1)$ again not equal to $\mathbf{0}$, which means TS_1 is not general-cycle-neutral. On the right of Fig. 4.4 for transition system TS_2 and its cycles $ab\overleftarrow{a}\,\overleftarrow{b}$ and $ba\overleftarrow{b}\,\overleftarrow{a}$ we have $\Psi(ab\overleftarrow{a}\,\overleftarrow{b}) = \Psi(ba\overleftarrow{b}\,\overleftarrow{a}) = (0, 0)$ implying that the property **gc0** is satisfied for TS_2.*

Figure 4.4: TS_1 is not **gc0**; TS_2 is general-cycle-neutral.

A deterministic labelled transition system, with a label set T, which satisfies property **gc0** can be interpreted as a part of a $|T|$-dimensional space where the dimensions correspond to the labels, and the nodes of the lts correspond to the points of the space having (non-negative) integral coordinates.

Proposition 24. *Let $TS = (S, T, \rightarrow, s_0)$ be a deterministic, totally-reachable labelled transition system satisfying the property* **gc0**. *Then every state $s \in S$ uniquely corresponds to a T-vector.*

Proof. Due to total-reachability, for any state $s \in S$, there is a path $\sigma_s \in T^*$ such that $s_0[\sigma_s\rangle s$. Assume that for some $s \in S$, there are $\sigma'_1, \sigma'_2 \in T^*$ such that $s_0[\sigma'_1\rangle s$, $s_0[\sigma'_2\rangle s$, and $\Psi(\sigma'_1) \neq \Psi(\sigma'_2)$. Then, there is a (possibly empty) path $\tau \in T^*$ and such that $\sigma'_1 = \tau\delta_1$ and $\sigma'_2 = \tau\delta_2$. Let $s'' \in S$ be such that $s_0[\tau\rangle s''$. By $\Psi(\sigma_1) \neq \Psi(\sigma_2)$, we have $\Psi(\delta_1) \neq \Psi(\delta_2)$. On the other hand, it holds that $s''[\delta_1\rangle s$ and $s''[\delta_2\rangle s$. This means that $\delta_1(\delta_2)^{-1}$ is a general cycle (from s'' to s''), and $\Psi(\delta_1(\delta_2)^{-1}) \neq \mathbf{0}$, contradicting the property **gc0**. □

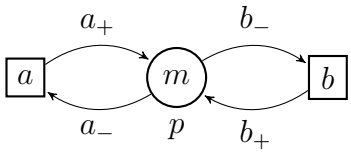


Figure 4.5: General place p of a Petri net with two transitions a and b.

With the interpretation of states of the lts as points of space, we can study the synthesisability of the class of totally-reachable, deterministic, **gc0** labelled transition systems with Petri nets. To this end, we consider the circumstances under which labels can be enabled or disabled at some state. For a synthesisable labelled transition system, enabling and disabling of labels is determined by the markings of the places in the Petri net solving the lts. Let us restrict our attention to the case of binary set of labels T (the more complicated cases can be considered similarly). A general place of a Petri net with a binary set of transitions is depicted in Fig. 4.5. Every state of the lts uniquely corresponds to a Parikh-vector, or a point in 2-dimensional space represented as a coordinate plane with x-axis as the abscissa line, y-axis as the ordinate line, and O as the origin, Oxy for short. A marking $M_s(p)$ of the place p at state s can be described by the following equation:

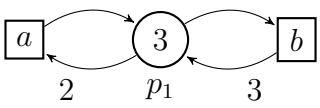
$$M_s(p) \;=\; m + (a_+ - a_-) \cdot x + (b_+ - b_-) \cdot y,$$

where $x = \Psi(s)(a)$ and $y = \Psi(s)(b)$. This equation is linear for both variables x and y, hence it determines a line in space Oxy. Assume that we aim to enable transition a with the place p. This can be done for points of Oxy satisfying the

following inequation

$$m + (a_+ - a_-) \cdot x + (b_+ - b_-) \cdot y \ \geq \ a_-. \tag{4.1}$$

Inequation (4.1) determines a half-space of Oxy. For instance, place p_1 in Fig. 4.6 allows transition a for firing only at the states corresponding to points from the shaded domain (r.h.s. of Fig. 4.6).



$$-x + 2 \cdot y + 1 \geq 0$$

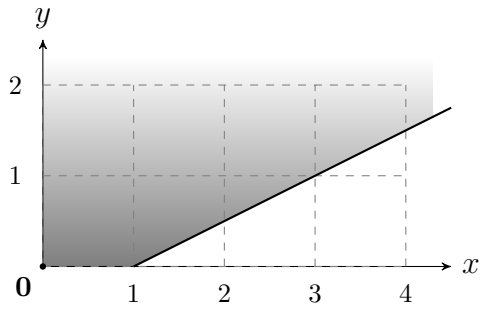


Figure 4.6: p_1 enables a only at the states corresponding to the points from the shadowed half-space of Oxy.

Example 24. *Transition system TS in Fig. 4.7 is deterministic and satisfies the property* **gc0***, hence it has a 2-dimensional interpretation (r.h.s. of Fig. 4.7). The initial state s_0 of TS corresponds to the origin O of the coordinate system Oab, and other states correspond to the space-points whose coordinates are defined by the Parikh vector of the state. For instance, state s_2 can be reached from the initial state s_0 by executing ab, hence s_2 corresponds to the point S_2 with coordinates $\Psi(ab) = (1, 1)$ in Oab.*

The example together with Proposition 24 may give us a first intuition about the class of finite languages which are solvable with Petri nets. In the following sections, this intuition will be presented more clearly and developed into a characterisation, together with over-approximation algorithms based on it.

Figure 4.7: TS satisfies property **gc0**. A 2-dimensional interpretation of TS in Oab (r.h.s.).

4.4 Abstract regions of lts

In Section 2.2, where we made a short presentation of the theory of regions, we have introduced the notion of a region. In the following paragraphs, we shall use the notion of an *abstract region*, which is more general compared to the regions considered before. This generality of the new notion of regions allows to adjust the initial state space in order to make it synthesisable with Petri nets.

Definition 32. *An abstract region r of an lts $(S, \{t_1, \ldots, t_n\}, \rightarrow, s_0)$ is a tuple $r = (r_0, r_1, \ldots, r_n) \in \mathbb{N}^S \times \mathbb{Z}^n$ with two consistency properties:*

- $\forall s, s' \in S \; \forall i \in \{1, \ldots, n\}$*:* $s \xrightarrow{t_i} s' \Rightarrow r_0(s') = r_0(s) + r_i$
- *for every g-cycle σ in the lts,* $\sum_{i=1}^{n} \Psi(\sigma)(t_i) \cdot r_i = 0$.

From this, it follows easily that the value $r_0(s)$ is fully defined by $r_0(s_0)$ and the r_i with $1 \leq i \leq n$. Let us also note that an abstract region $r = (r_0, r_1, \ldots, r_n)$ for $T = \{t_1, \ldots, t_n\}$ can be obtained from a region $\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ introduced earlier and its effect $\mathbb{E}$ as follows:

$$r_0 = \mathbb{R}, \;\; r_i = \mathbb{E}(t_i), \text{ for } 1 \leq i \leq n.$$

An abstract region in a reachability graph gives rise to an equivalence class of places p where $M_0(p) = r_0(s_0)$ and $r_i = F(t_i, p) - F(p, t_i)$. This means that unlike ordinary regions, the places obtained from abstract regions are defined up to the effect of transitions (the exact values of arc weights can vary). If the

region distinguishes the states s and s' of two reachable markings M and M' (by $r_0(s) \neq r_0(s')$), the firing rule of the Petri net ensures that $M(p) \neq M'(p)$.

Lemma 19. [EW17](indistinguishable regions) *Let $r = (r_0, r_1, \ldots, r_n)$ be an abstract region of some (totally reachable) lts $(S, \{t_1, \ldots, t_n\}, \rightarrow, s_0)$. If r does not solve an instance $\{s, s'\}$ of SSP (by distinguishing s and s'), neither does $r' = (k \cdot r_0 + i, k \cdot r_1, \ldots, k \cdot r_n)$ with $i, k \in \mathbb{Z}$.*

Proof. If $k \cdot r_0(s) + i < 0$, r' is not a region, there is nothing to prove. Assume that $r_0(s) = r_0(s')$, i.e. r does not distinguish s ans s'. Then, $(k \cdot r_0(s') + i) - (k \cdot r_0(s) + i) = k \cdot (r_0(s') - r_0(s)) = 0$, which means that r' does not distinguish s and s' as well. □

Places constructed from abstract regions are sufficient to deal with all states separation problems but not with all event-state separation problems. For fully determined places (with exactly known values for $F(p, t_i)$ and $F(t_i, p)$) one would have to consider ordinary regions (which are more refined). Instead, here we argue directly about the loops at some place p in a Petri net, i.e. about the value

$$k_{t_i} = \min\{F(p, t_i), F(t_i, p)\}$$

for each transition $t_i \in \{t_1, \ldots, t_n\}$. Together with a region $r = (r_0, r_1, \ldots, r_n)$ for p, they fully determine the arc weights between p and its neighbouring transitions. More precisely, since r_i is equal to the effect of t_i for p, i.e. $r_i = F(t_i, p) - F(p, t_i)$, then the arc weights are determined as:

$$\begin{array}{llll} F(t_i, p) = k_{t_i}, & F(p, t_i) = k_{t_i} + |r_i|, & \text{if } r_i < 0, & \text{or} \\ F(t_i, p) = k_{t_i} + r_i, & F(p, t_i) = k_{t_i}, & \text{if } r_i \geq 0. & \end{array}$$

Places derived from the same region, distinguished by the loop values k_t only, can easily be unified.

Lemma 20. [EW17](loop maximisation) *Let $N = (P, T, F, M_0)$ be a Petri net with $p_1, p_2 \in P$ and $k_t \in \mathbb{Z}$ for each $t \in T$ such that $M_0(p_2) = M_0(p_1)$, and $\forall t \in T$: $F(p_2, t) = F(p_1, t) + k_t$ and $F(t, p_2) = F(t, p_1) + k_t$. Define a new Petri net N' by adding k_t to each of $F(p_1, t)$ and $F(t, p_1)$ if $k_t > 0$, for every $t \in T$, and then deleting p_2 including all adjacent arcs. N' has the same reachability graph as N.*

Proof. Note that $M(p_1) = M(p_2)$ holds for all reachable markings M in N. A transition t is disabled in N at M by place p_1 or p_2, if $M(p_1) < F(p_1, t)$ or $M(p_2) < F(p_2, t)$, or put differently, if $M(p_1) < \max\{F(p_1, t), F(p_2, t)\}$. This maximum is exactly the arc weight $F(p_1, t)$ as defined for N'. Since the token change on p_1 by firing a transition is the same in N and N' and enabledness of transitions also remains unchanged from N to N', both nets have the same reachability graph. □

Example 25. *For the transition system TS on the left of Fig. 4.8 we can construct an abstract region $r = (0, 1, -1)$. It can be directly checked that all the constraints for r and TS from Definition 32 are satisfied. This region gives rise to a class of places p with $M_0(p) = 0$, $F(a, p) - F(p, a) = 1$ and $F(b, p) - F(p, b) = -1$. Places p_1 and p_2 of net N in the middle of Fig. 4.8 satisfy these restrictions, hence belong to the class. These places differ in the number of loops around transition b in N, in particular p_2 being a side-condition with a single loop and p_1 being pure. For $k_a = 0$ and $k_b = 1$, the premises of Lemma 20 are satisfied. Hence, one can modify N into net N' (in the right of Fig. 4.8) which has the reduced number of places but the same behaviour. Indeed, the reachability graphs of N and N' are isomorphic to TS.*

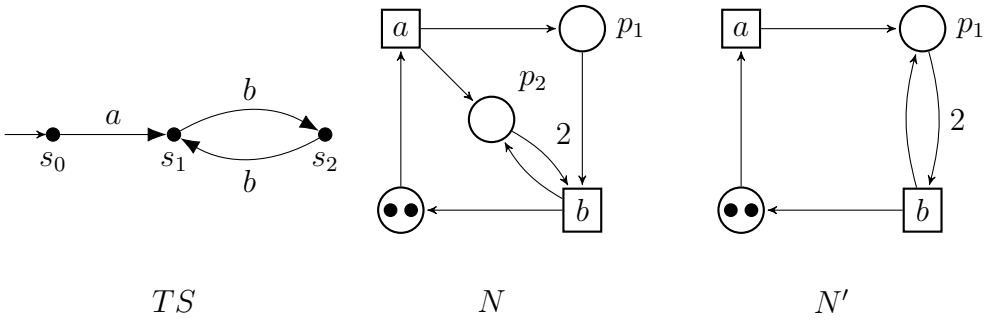


Figure 4.8: Places p_1 and p_2 in N and p_1 in N' correspond to the same abstract region of TS.

So, of all places constructed from the same abstract region, it is enough to have only one, i.e. the place with the maximal number k_t of loops, separately computed for each adjacent transition t in the Petri net. This place performs the functions of all the others, obtained from the same region.

4.5 Generalised cycles with non-zero Parikh vectors

In the present and the following sections we continue our investigations in characterising of solvable state spaces. We now relax the restrictions for the shape of the initial lts, but retain the binary transition set. In other words, we now would like to characterise the reachability graphs of Petri nets with at most two transitions. As a first case, we consider reachability graphs that contain at least one generalised cycle with a non-zero Parikh vector.

Theorem 7. [EW17](shapes with non-zero Parikh cycles) *If the reachability graph of a Petri net $(P, \{a, b\}, F, M_0)$ contains a g-cycle with a non-zero Parikh vector, it has one of the seven general shapes shown in Figures 4.9, 4.11, and 4.13 or it consists of the initial state without any edges.*

Proof. Let $(S, \{a, b\}, \rightarrow, s_0)$ be a synthesisable lts, i.e. the reachability graph of some Petri net $(P, \{a, b\}, F, M_0)$, with some g-cycle $s \overset{\sigma}{\rightsquigarrow} s$ such that $\Psi(\sigma) \neq 0$.

From the definition of abstract regions, we know that every g-cycle must have a weighted sum of zero in any abstract region of the lts. Let $r = (r_0, r_a, r_b)$ be any such region, then $\Psi(\sigma)(a) \cdot r_a + \Psi(\sigma)(b) \cdot r_b = 0$. Our knowledge about r_a and r_b depends directly on the Parikh vector of σ.

Case 1: $\Psi(\sigma)(a) = 0$. Then, $\Psi(\sigma)(b) \neq 0$ and thus $r_b = 0$, but we know nothing about r_a. As a consequence, following a b-edge $s \overset{b}{\rightarrow} s'$ in the lts cannot modify the region value, i.e. $r_0(s') = r_0(s) + r_b = r_0(s)$. Therefore, no region can distinguish s and s', and neither can any place distinguish the corresponding markings in the Petri net; the markings must be identical. We conclude that $s = s'$ for every b-edge $s \overset{b}{\rightarrow} s'$, all b-edges must be loops in the reachability graph. Fig. 4.9 depicts all possible reachability graphs depending on whether regions with negative, zero, or positive r_a occur in the lts, together with a class of sample Petri nets with such reachability graphs.

- If in all regions $r = (r_0, r_a, 0)$, $r_a = 0$, transition a cannot change the marking in the Petri net either. If a can fire at all, we obtain the reachability graph in the first row of Fig. 4.9. If a cannot fire, we get the second row.

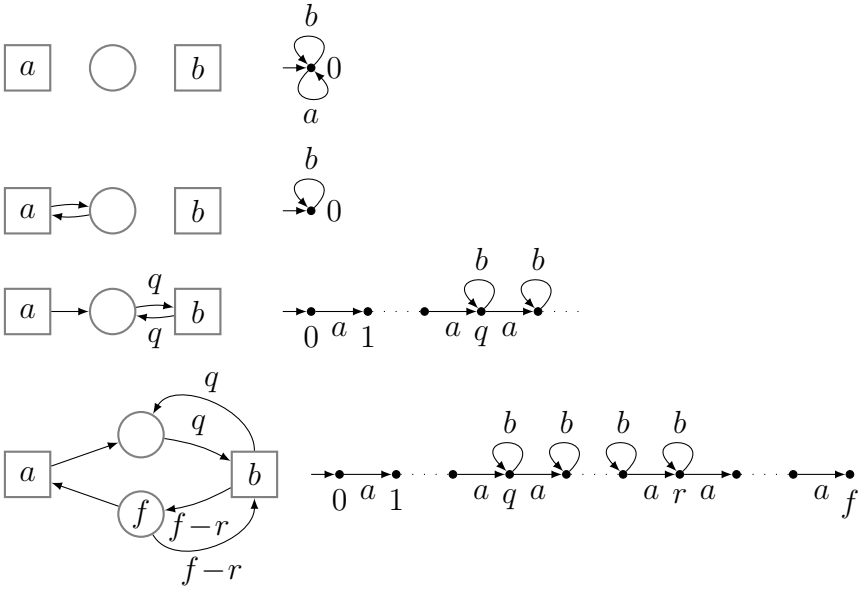


Figure 4.9: Reachability graphs with loops (cycles of length 1). For easy identification of markings in the Petri nets on the left side, consider corresponding states as consecutively named 0, 1, 2, ...

- If there is a region with $r_a > 0$ but none with $r_a < 0$, and assuming that a can fire at all, its firing will increase the number of tokens on any place connected to it. With increasing markings we obtain an infinite series of pairwise distinguishable states as in the third row of Fig. 4.9. Since b occurs in the reachability graph, there is some earliest state q at which it forms a loop. As the markings rise from this point on, b is also activated at all later states.

- If there is a region with $r_a < 0$, there must be a place from which a removes tokens. There is some $f \in \mathbb{N}$ such that a can fire exactly f times. The transition b can be deactivated at an earlier state r, if the place is a side-place of b (in the picture with arc weight $f - r$). If there is also a region with $r_a > 0$, b may not be activated until some state q just as in the previous subcase.

Case 2: $\Psi(\sigma)(b) = 0$. This case is analogous to the previous one.

Case 3: $\Psi(\sigma)(a) \cdot \Psi(\sigma)(b) > 0$. Then, $r_a \cdot r_b < 0$ and $r_b = -\frac{\Psi(\sigma)(a)}{\Psi(\sigma)(b)} \cdot r_a$, so r_a and r_b have a fixed ratio in every region. For every pair of such regions, we

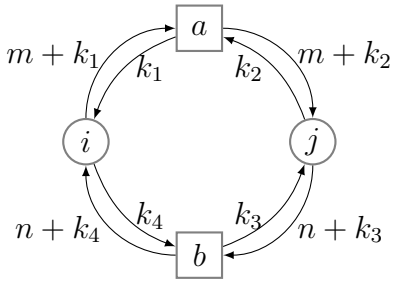


Figure 4.10: A representative class of Petri nets for the case $\Psi(\sigma)(a) \cdot \Psi(\sigma)(b) > 0$.

can use Lemma 19 to find a new region that is a common multiple with modified initial value r_0, i.e. any two such regions solve exactly the same state separation problems. Therefore, only two regions are of interest, one with $r_a < 0$ and $r_b > 0$ that can prevent an a, and one with $r_a > 0$ and $r_b < 0$ which can prevent b. By Lemma 20, the number of places constructed from each region can be reduced to one, the one with the maximal number of loops at each transition. Therefore, it is sufficient to have at most two regions, (r_0, r_a, r_b) and $(r'_0, -r_a, -r_b)$, and to construct one place from each region. A representative class of Petri nets with such two regions is shown in Fig. 4.10.

An lts $TS = (S, \{a, b\}, \rightarrow, 0)$ representing the reachability graph of the Petri net in Fig. 4.10 can be formally described as follows:

$$
\begin{aligned}
&S = \{-j, \ldots, +i\} \cap \gcd(m, n) \cdot \mathbb{Z} \\
&\forall x \in S : \\
&x \xrightarrow{a} x + m \quad \text{is present in } \rightarrow \quad \Longleftrightarrow \quad k_2 - j \leq x \wedge x + m \leq i - k_1 \\
&x \xrightarrow{b} x - n \quad \text{is present in } \rightarrow \quad \Longleftrightarrow \quad k_3 - j \leq x - n \wedge x \leq i - k_4
\end{aligned}
$$

For simplicity, we have named the initial state 0, which leads to states with negative numbers. By adding j, we can obtain the values of a possible region. Some unreachable states are easy to exclude (by intersecting with $\gcd(m, n) \cdot \mathbb{Z}$), others may occur if the initial state does not lie on a cycle.

If we omit one of the two regions (places), the state space will become infinite, as either the boundary $-j$ or the boundary i will fall. We may think of this as replacing $-j$ by $-\infty$ or i by ∞. If we omit both places, the reachability graph

collapses to the first row of Fig. 4.9.

Fig. 4.11 visualises an example of a reachability graph with a cycle, using a Petri net according to Fig. 4.10 which has parameters $m = 5$, $n = 3$, $i = 32$, $j = 0$, $k_1 = 0$, $k_2 = 0$, $k_3 = 9$, and $k_4 = 0$. The value $k_3 = 9$ ensures that the two initial a's cannot be undone (due to the loop between b and the right place, the initial marking 0 of the place can never be reached again). This also makes the states 1 to 4 and 6 to 8 unreachable. Incrementing i by one will add one a-edge at the highest state without such an edge to some new state and one b-edge at the new state. Setting i to 33 will add an a-edge from 28 to 33 (new) and one b-edge from 33 to 30. Each increment will complete a new diamond in the graph. E.g., here $28 \xrightarrow{b} 25 \xrightarrow{a} 30$ will be complemented by $28 \xrightarrow{a} 33 \xrightarrow{b} 30$, and states 28, 25, 33, 30 will form a diamond. Incrementing j will analogously add a diamond at the other end. Adding loops in the net via incrementing parameters k_1 to k_4 essentially cuts off such diamonds again (by disabling transitions at the vertexes of dimonds), unless the cutting point is near the initial state. In this case, only one kind of edge is removed. The initial state will then not lie on a cycle anymore but form a path (using the other kind of edges) that leads to the cyclic part of the lts. The parameters m and n determine the length of cycles in the lts and the ratio of a and b on these cycles.

Case 4: $\Psi(\sigma)(a) \cdot \Psi(\sigma)(b) < 0$. With the same reasoning as in the previous case except that $r_a \cdot r_b > 0$, we find again that two regions (r_0, r_a, r_b) and $(r'_0, -r_a, -r_b)$ and one place constructed from each region are enough. This leads to the class of Petri nets in Fig. 4.12. An lts $TS = (S, \{a, b\}, \rightarrow, 0)$ representing the reachability graph is formally described as follows:

$$
\begin{aligned}
&S = \{0, \ldots, i\} \\
&\forall x \in S : \\
&x \xrightarrow{a} x + m \quad \text{is present in } \rightarrow \quad \Longleftrightarrow \quad k_2 - j \leq x \wedge x + m \leq i - k_1 \\
&x \xrightarrow{b} x + n \quad \text{is present in } \rightarrow \quad \Longleftrightarrow \quad k_3 - j \leq x \wedge x + n \leq i - k_4
\end{aligned}
$$

Some states may be unreachable.

Unlike in Case 3, there is a steady token flow from the left to the right place, no matter which transition is fired. Therefore, the right place determines when a transition may start firing and the left place when the firing must cease. Since

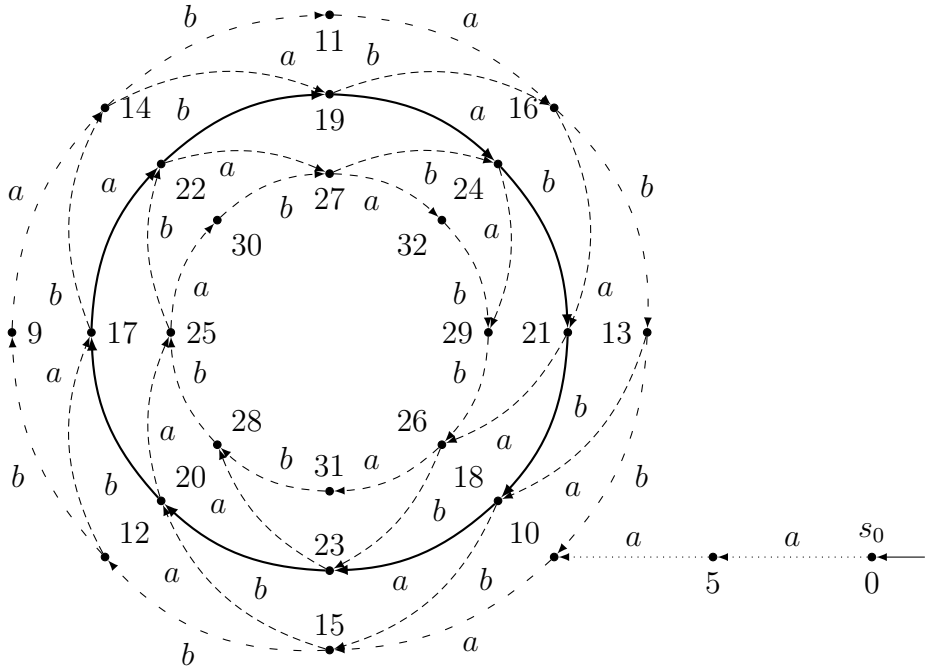


Figure 4.11: Visualisation of a reachability graph with (standard) cycles

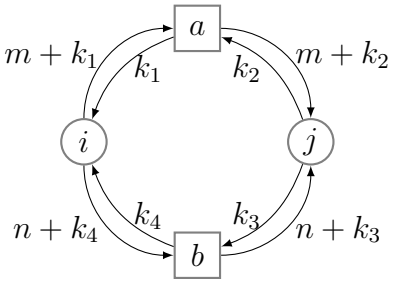


Figure 4.12: A representative class of Petri nets for the case $\Psi(\sigma)(a) \cdot \Psi(\sigma)(b) < 0$.

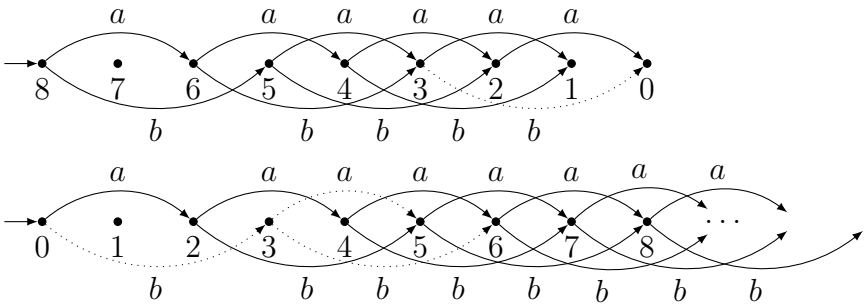


Figure 4.13: Visualisation of reachability graphs containing g-cycles with non-zero Parikh vector, but no directed cycles

the left place will at some point stop both transitions from firing, the reachability graph will normally be finite. Only if we remove the left place from the net, we can obtain an infinite behaviour (with states then named according to the number of tokens on the right place). These possibilities are shown in Fig. 4.13.

E.g., the upper reachability graph (with the dotted edge) corresponds to the Petri net with parameters $i = 8$, $j = 0$, $m = 2$, $n = 3$, and $k_1 = k_2 = k_3 = k_4 = 0$. Introducing loops at the left place will prevent the rightmost edges, i.e. setting $k_4 = 1$ will eliminate the dotted b-edge, setting it to 2 will also prevent the b-edge ending at state 1, and so on.

The lower reachability graph corresponds to a Petri net without the left place and its adjacent edges, having the parameters $j = 0$, $m = 2$, $n = 3$, and $k_2 = k_3 = 0$. Adding a loop at the right place prevents edges beginning at the initial state. Setting $k_3 = 1$ will remove the b-edge at 0 and make state 3 unreachable. Therefore, the edges from state 3 also become unusable (shown as dotted lines).

With Cases 1 to 4, we have covered all possible g-cycles with non-zero Parikh vector that might occur in a reachability graph of a Petri net with transitions a and b. □

4.6 Generalised cycles with zero Parikh vectors

Let us now assume that our transition system does not have any g-cycle σ with $\Psi(\sigma) \neq 0$. In this case, the transitions a and b are independent, and we may project the lts onto the plane $\mathbb{N}^2$, with the initial state mapped to $(0, 0)$. The transitions a and b become the base vectors, i.e. firing a increments the first component

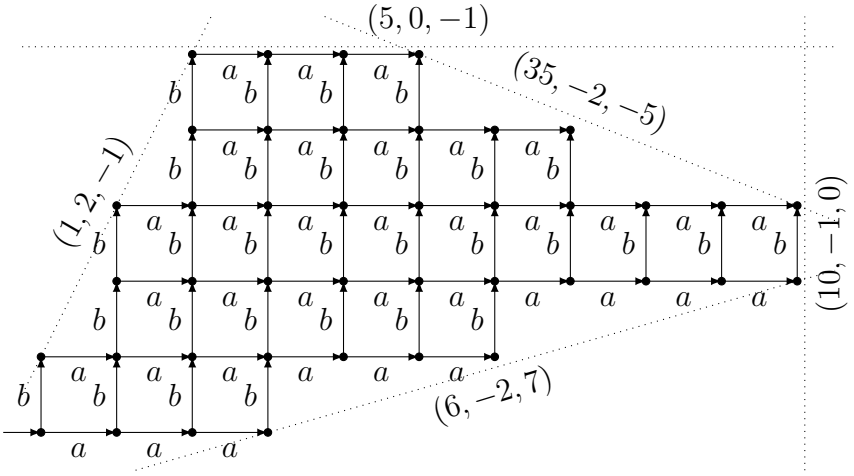


Figure 4.14: An lts without Parikh-non-zero g-cycles, limited by 5 regions (shown as dotted lines at which the value of a region (r_0, r_a, r_b) is zero). The initial state is at the lower left and can be viewed as the origin $(0, 0)$ of the plane $\mathbb{N}^2$, where a and b are the unit vectors of the two dimensions

of a state, $(x, y) \xrightarrow{a} (x + 1, y)$, and firing b increments the second component, $(x, y) \xrightarrow{b} (x, y + 1)$, whenever the transitions are allowed.

Since all g-cycles σ have zero Parikh vectors, $\Psi(\sigma) = 0$, the equation

$$\Psi(\sigma)(a) \cdot r_a + \Psi(\sigma)(b) \cdot r_b = 0$$

does not restrict the values for r_a and r_b of a region in any way. If we distinguish regions (r_0, r_a, r_b) by the signs of r_a and r_b, there are essentially nine types of regions. Regions with non-negative values for r_a as well as r_b do not restrict the enabledness of transitions. With positive values for r_a and r_b, the five remaining types can be written as

$$(r_0, -, +),\ (r_0, -, 0),\ (r_0, -, -),\ (r_0, 0, -),\ \text{ and }\ (r_0, +, -),$$

where $+$, or $-$, or 0 means that the corresponding coordinate is > 0, or < 0, or $= 0$, respectively. An example lts with one region of each of the five types is shown in Fig. 4.14.

Let now $N = (P, \{a, b\}, F, M_0)$ be a Petri net and G the projection of the reachability graph onto $\mathbb{N}^2$. Furthermore, let R be the set of regions of G.

Lemma 21. [EW17](inner states) *Let N be pure (i.e. $\forall p, t: F(p,t) \cdot F(t,p) = 0$) and $(x,y) \in G$ be some state. If $\forall (r_0, r_a, r_b) \in R$: $r_0((x+1,y)) \geq 0$, then $(x+1,y) \in G$ with $(x,y) \xrightarrow{a} (x+1,y)$. If $\forall (r_0, r_a, r_b) \in R$: $r_0((x,y+1)) \geq 0$, then $(x,y+1) \in G$ with $(x,y) \xrightarrow{b} (x,y+1)$.*

Proof. Note first that for any state $(x,y) \in \mathbb{N}^2$ and any region $(r_0, r_a, r_b) \in R$, $r_0((x+1,y)) = r_0((x,y)) + r_a$ and $r_0((x,y+1)) = r_0((x,y)) + r_b$ holds by the definition of a region, since we use a and b as base vectors in $\mathbb{N}^2$.

If $\neg(x,y) \xrightarrow{a}$ in G, there is a region $(r_0, r_a, r_b) \in R$ with $r_a < 0$. For a place p constructed from such a region, $r_a = F(a,p) - F(p,a) < 0$ holds. Since N is pure, $F(p,a) > 0$ and $F(a,p) = 0$. Then, $r_0((x+1,y)) - r_0((x,y)) = r_a = -F(p,a)$, and with $r_0((x+1,y)) \geq 0$ we conclude $r_0((x,y)) \geq F(p,a)$. Since by construction of p, $r_0((x,y))$ is the number of tokens on p at the state (x,y), p cannot prevent a firing of a at the corresponding marking. Since the region and p were arbitrary, $(x,y) \xrightarrow{a} (x+1,y)$ holds in G, i.e. $(x+1,y) \in G$.

An analogous reasoning holds for state $(x,y+1)$ and transition b. □

Theorem 8. [EW17](reachability graphs of pure nets without Parikh-non-zero g-cycles) *An lts G in which all g-cycles have Parikh vector zero is the reachability graph of a pure Petri net $N = (P, \{a,b\}, F, M_0)$ if and only if it can be viewed as a weakly connected convex subset of $\mathbb{N}^2$ containing the initial state $(0,0)$ such that for each two states $(x,y), (x+1,y) \in G$: $(x,y) \xrightarrow{a} (x+1,y)$ and for each two states $(x,y), (x,y+1) \in G$: $(x,y) \xrightarrow{b} (x,y+1)$.*

Proof. Convex subsets of $\mathbb{N}^2$ can be defined by cutting off parts of $\mathbb{N}^2$ using straight lines, and these lines may not cut off the initial state, so the five types of regions identified above give rise to exactly this kind of convex subset.

Using Lemma 21 proves that all necessary states and edges exist. In some extreme cases we may obtain states that cannot be connected via g-paths. E.g. with regions $(0,1,-1)$, $(0,-1,1)$, $(0,1,0)$, and $(0,0,1)$, the states (x,x) with $x \in \mathbb{N}$. Then, only the weakly connected component of the graph that contains the initial state forms the reachability graph.

It remains to show that a pure Petri net can be found such that its reachability graph does not identify any two of the states in $\mathbb{N}^2$. This can be done using the regions $(0,1,0)$ and $(0,0,1)$, i.e. by adding to each postset of a and b one new

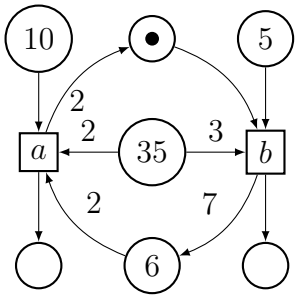


Figure 4.15: A pure Petri net with the lts from Fig. 4.14 as its reachability graph. Each region forms one place. Though unnecessary in this case, we add regions $(0, 1, 0)$ and $(0, 0, 1)$ to ensure that no two states can have the same marking

place with an empty initial marking, counting the number of a's or b's that have occurred. □

Figure 4.14 is a typical representative of such a reachability graph of a pure net (over transitions a and b). The regions cutting off parts of $\mathbb{N}^2$ may vary in number and direction, and they may not even make the graph finite, but the 'inside area' (the convex subset of $\mathbb{N}^2$) will always be completely filled with states and edges. A pure Petri net synthesising the lts from Fig. 4.14 can be seen in Fig. 4.15.

To characterise the reachability graphs of non-pure nets in the same way, we need to consider the effect of adding a self-loop between a place (representing some region (r_0, r_a, r_b)) and the transition a or b. The dotted lines in Fig. 4.14 show where the corresponding region has the value zero, or put differently, they are the border that edges do not cross. We can also interpret them as two different lines at the same location, one preventing the a-edges, the other preventing the b-edges from crossing into the half-plane of negative region values. Let us call these lines the *a-line* and *b-line* of the region.

Lemma 22. [EW17](shifting enabledness lines) *Let $N = (P, \{a, b\}, F, M_0)$ be a pure Petri net with a place $p \in P$ (with corresponding region (r_0, r_a, r_b)), and let N' by derived from N by adding $k \in \mathbb{N}$ to both $F(p, a)$ and $F(a, p)$. Let G and G' be the reachability graphs of N and N' projected to $\mathbb{N}^2$. If $r_a \neq 0$ then, from G to G', the a-line is shifted by the fraction of $\frac{k}{r_a}$ of the unit-a-vector. If $r_a = 0$ and $r_b \neq 0$, then the a-line is shifted by the fraction of $\frac{k}{r_b}$ of the unit b-vector. If $r_a = r_b = 0$, either nothing happens (in case $r_0 \geq k$) or G' collapses and does not contain any a-edges (if $r_0 < k$).*

Proof. Note first that G' is a subgraph of G since N' has a more restricted behaviour than N. If $r_a \neq 0$, any a-edge changes the number of tokens on p by r_a, therefore the $\frac{k}{r_a}$th fraction of an a-edge increases the number of tokens on p by k. This is exactly the additional amount of tokens needed in G' to allow an a-edge, thus the a-line moves from G to G' by the $\frac{k}{r_a}$th fraction of a unit-a-vector (possible in the opposite direction if $r_a < 0$).

If $r_a = 0$, still a-edges are allowed only if the region value is at least k, since in N' we have $F(p,a) = k = F(a,p)$. With the same reasoning as above, the line with exactly k tokens on p can be moving the $\frac{k}{r_b}$th fraction of a unit-b-vector away from the b-line, which makes this the new a-line in G'.

If the region is $(r_0, 0, 0)$ neither transition can change the start value r_0. We either have $r_0 \geq k$, i.e. $r_0 \geq F(p,a) = F(a,p)$, so the number of tokens on p is sufficient to allow an a at all states. Or we have $r_0 < k$, in which case there are no a-edges in G' at all and G' represents some word from b^*. □

This lemma can symmetrically be formulated for the transition b as well. In Fig. 4.16, we see the effect of adding one self-loop around a for each of the five places representing one of the regions $(1, 2, -1)$, $(5, 0, -1)$, $(35, -2, -5)$, $(10, -1, 0)$, and $(6, -2, 7)$. The lemma also includes regions like $(0, 1, 0)$, that usually will not be shown in the lts because they are positioned left of or below the origin. A shifted line from such a region can easily cut off the origin and thus collapse the reachability graph, as in the case of $(r_0, 0, 0)$ in the lemma.

We can then characterise the reachability graphs of nets $(P, \{a, b\}, F, M_0)$ as follows.

Theorem 9. [EW17](reachability graphs of nets without Parikh-non-zero g-cycles) *Let $C \subseteq \mathbb{N}^2$ be a convex area and let C_a, C_b be derived from C by shifting borders of C only. Let G be the graph including $(0,0)$ such that $(x,y) \xrightarrow{a} (x+1,y) \in G \iff (x,y) \in G \cap C_a, (x+1,y) \in C_a$ and $(x,y) \xrightarrow{b} (x,y+1) \in G \iff (x,y) \in G \cap C_b, (x,y+1) \in C_b$. Then, G is the projection of a reachability graph of a Petri net $N = (P, \{a,b\}, F, M_0)$ to $\mathbb{N}^2$. If the reachability graph of a Petri net $(P, \{a,b\}, F, M_0)$ does not contain g-cycles σ with $\Psi(\sigma) \neq 0$, C_a and C_b meeting the above conditions can always be found.*

As the main result of the section, we have obtained a characterisation of the state spaces of bounded Petri nets having at most two transitions. The condition of

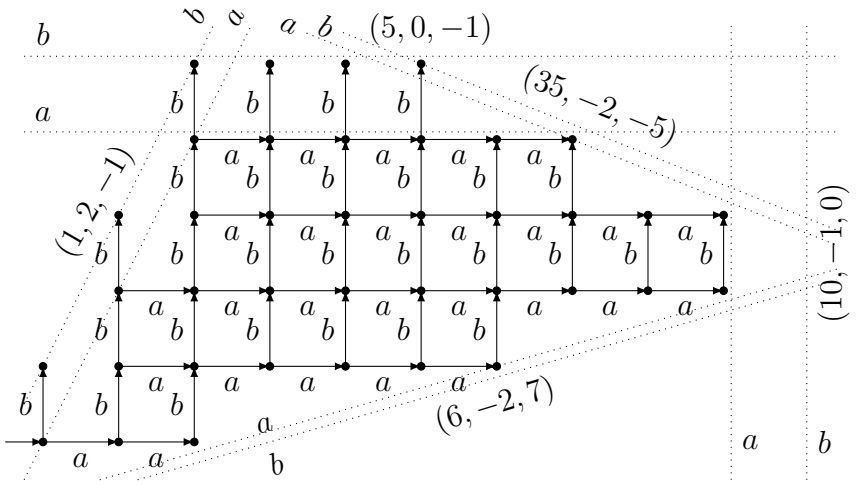


Figure 4.16: The reachability graph of the net of Fig. 4.15 if we add, for each of the five original regions, a self-loop between the corresponding place and a. The line where a region has value zero is split into two: one line that a-edges cannot permeate and one line that b-edges cannot cross (marked with letters a and b)

boundedness relates to the fact that we initiated our investigation for finite labelled transition systems. The characterisation establishes that these state spaces can be represented as a convex sets of integral points of $|T|$-dimensional space, where T is the alphabet being used. In what follows, we present a synthesis algorithm based on this characterisation in two versions: when a Petri net is sought in the class of pure nets, and without this restriction. An important remark should be made, namely, unlike in the first chapters, we will now look for a possibly minimal Petri net over-approximation of the input behaviour.

4.7 Over-approximation of finite languages

Assume now that we are given a finite language $L \subseteq T^*$ over a binary alphabet $T = \{a, b\}$, and we aim to synthesise a Petri net which allows firing all the words of this language. Due to the fact that for each Petri net, every prefix of a feasible transition sequence is also feasible, we will assume L to be *prefix-closed*, i.e. for every $uv \in L$ with $u, v \in \{a, b\}^*$, $u \in L$ holds. For a Petri net N, let $\mathcal{L}(N)$ be the set of all transition sequences fireable in N. It may be the case that there is no net N such that $\mathcal{L}(N) = L$. E.g., if $L = \{abbaa\}$. Nevertheless, there exists a net

N such that $L \subseteq \mathcal{L}(N)$. For instance, the Petri net N having a single place p with its initial marking $M_0(p) = \max_{w \in L} |w|$, and two transitions a and b such that $F(p,a) = F(p,b) = 1$ and $F(a,p) = F(b,p) = 0$. Hence the challenging problem is to find a net N such that $L \subseteq \mathcal{L}(N)$ and the difference between $\mathcal{L}(N)$ and L is 'minimal'. In order to define this minimality formally, a projection of a word or a finite language over T to a $|T|$-dimensional space is defined.

Definition 33. *A word $w = t_1 t_2 \dots t_n$ of length $n \in \mathbb{N}$ over alphabet T uniquely corresponds to a finite* (projective) transition system

$$\begin{aligned} PTS(w) &= (S, T, \rightarrow, s_0), \quad \textit{where} \\ S &= \{\Psi(\epsilon), \Psi(t_1), \dots, \Psi(t_1 \dots t_n)\}, \\ \rightarrow &= \{(\Psi(t_1 \dots t_{i-1}), t_i, \Psi(t_1 \dots t_i)) \mid 0 < i \leq n \wedge t_i \in T\}, \\ s_0 &= \Psi(\epsilon) \end{aligned}$$

For a finite language L we can uniquely define a (projective) transition system

$$PTS(L) = \bigcup_{w \in L} PTS(w),$$

where for words $w_1, w_2 \in T^*$ and $PTS(w_1) = (S_1, T, \rightarrow_1, \Psi(\epsilon))$, $PTS(w_2) = (S_2, T, \rightarrow_2, \Psi(\epsilon))$ we write

$$PTS(w_1) \cup PTS(w_2) = (S_1 \cup S_2, T, \rightarrow_1 \cup \rightarrow_2, \Psi(\epsilon)).$$

Unlike the introduced in Chapter 2 transition system $TS(w)$ for a word w over T, where the states were ordinary numbers, in $PTS(w)$ each state is a Parikh vector, i.e. it can be considered as a point of $|T|$-dimensional space.

Definition 34. *Given a finite language L, we say that a Petri net N* minimally over-approximates *L iff $L \subseteq \mathcal{L}(N)$ and for transition system $PTS(L) = (S_1, T, \rightarrow_1, \mathbf{0})$ and $PTS(\mathcal{L}(N)) = (S_2, T, \rightarrow_2, \mathbf{0})$, the set $S_2 \setminus S_1$ is minimal.*

The characterisations established in Theorems 8 and 9 suggest possible algorithms for such an over-approximation of finite languages over the binary alphabet, dealt with in the next subsections 4.7.1 and 4.7.2.

4.7.1 Pure over-approximation

Given a finite language L over the binary alphabet, regarding Theorem 8, in order to find a (minimal) language of a pure Petri net which includes L, we have to find the convex hull of S in $\mathbb{N}^2$ – the minimal convex set of points of $\mathbb{N}^2$ containing S, where S is the set of states of $PTS(L)$.

Let $L = \{abbabaa,\ bbabaaa\}$ be an example language for which we seek to produce a Petri net whose language includes L. From L we can construct $PTS(L) = (S, \{a, b\}, \rightarrow, (0, 0))$ without Parikh-non-zero g-cycles (l.h.s. in Fig. 4.17) whose states can be considered as points of $\mathbb{N}^2$ (r.h.s. in Fig. 4.17).

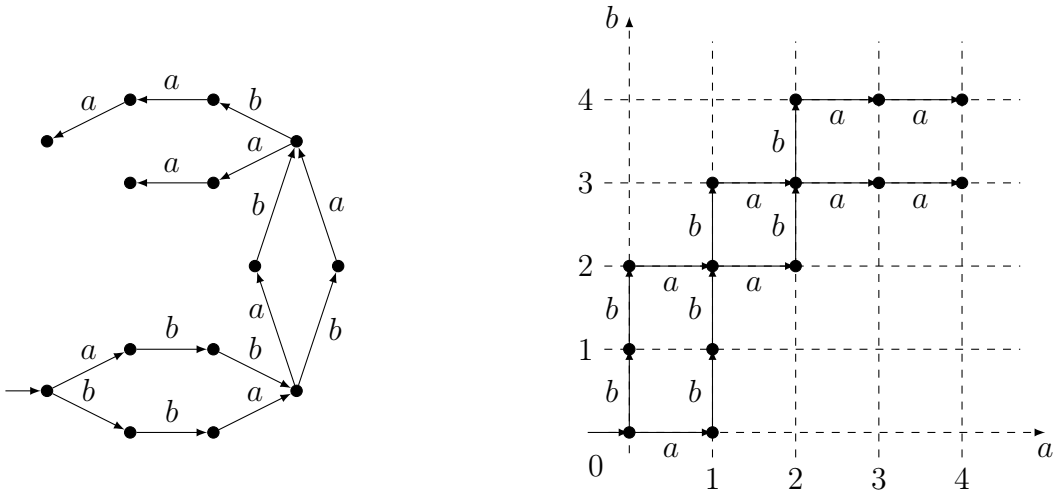


Figure 4.17: The lts (l.h.s.) derived from language L and its projection (r.h.s.) to $\mathbb{N}^2$.

Constructing of the convex hull of S can be done using `Jarvis march` [Jar73], Graham's algorithm [Gra72] or `Quickhull` [Edd77] algorithms which produce the hull of a given finite set of points as an ordered set of points.

Then the borders of the hull (which are lines in case of $\mathbb{N}^2$) are essentially regions of a reachability graph, and hence they are places of the net implementing the over-approximation of L. Since we are considering the case of pure nets, these regions can immediately be translated into places of the net (see r.h.s. in Fig. 4.18). To obtain the sought language, we have to add all the points of $\mathbb{N}^2$ inside the hull to S together with all the missing arrows inside the hull. The set of sequences enabled in the derived labelled transition system determines the

over-approximating language (l.h.s. in Fig. 4.18).

Algorithm 3 Pure over-app.

Input: finite (prefix-closed) language $L \in \{a, b\}^*$

Output: pure Petri net over-approximating L

compute set $S = \{(x, y) \mid \Psi(w) = (x, y),\ w \in L\}$

find the convex hull $H = \{(x_i, y_i)\}_{i=0}^{k} \subseteq S$ of S (enumerated clockwise)

define the set of places $P = \emptyset$

for $i = 0$ **to** $k - 1$ **do** {construct the set of regions R (and places P)}

$r_a^i \leftarrow y_i - y_{i+1}$ {define region as a line through two points}

$r_b^i \leftarrow x_{i+1} - x_i$ {orthogonal direction is chosen according to the ordering of H}

$r_0^i \leftarrow -r_a^i \cdot x_i - r_b^i \cdot y_i$

define place p_i

$M_0(p_i) \leftarrow r_0^i$

if $r_a^i \geq 0$ **then** $F(a, p_i) \leftarrow r_a^i$, $F(p_i, a) \leftarrow 0$ **else** $F(p_i, a) \leftarrow r_a^i$, $F(a, p_i) \leftarrow 0$

if $r_b^i \geq 0$ **then** $F(b, p_i) \leftarrow r_b^i$, $F(p_i, b) \leftarrow 0$ **else** $F(p_i, b) \leftarrow r_b^i$, $F(b, p_i) \leftarrow 0$

add place p_i to P

endfor

return $(P, \{a, b\}, F, M_0)$

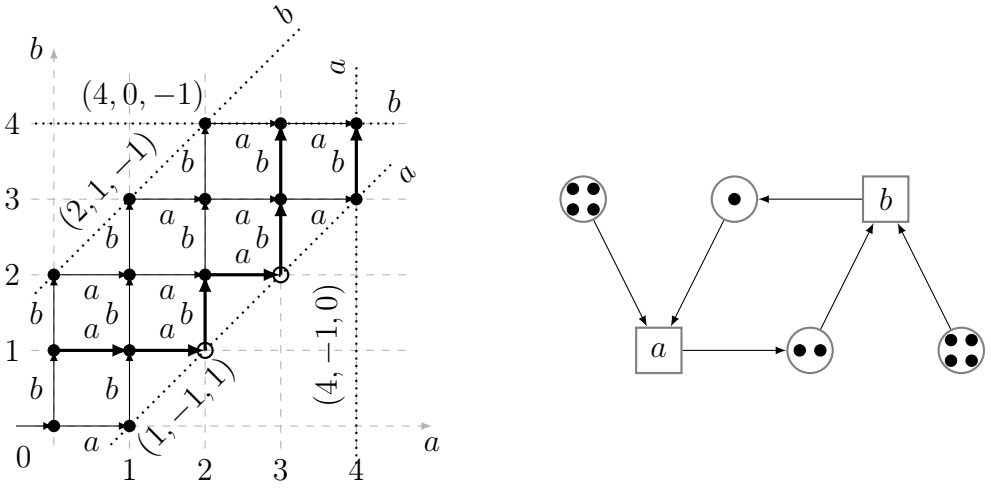


Figure 4.18: The minimal convex hull of S (l.h.s.) and a pure Petri net (r.h.s.) derived from it; hollow states and thick arrows on the left were added to $PTS(L)$.

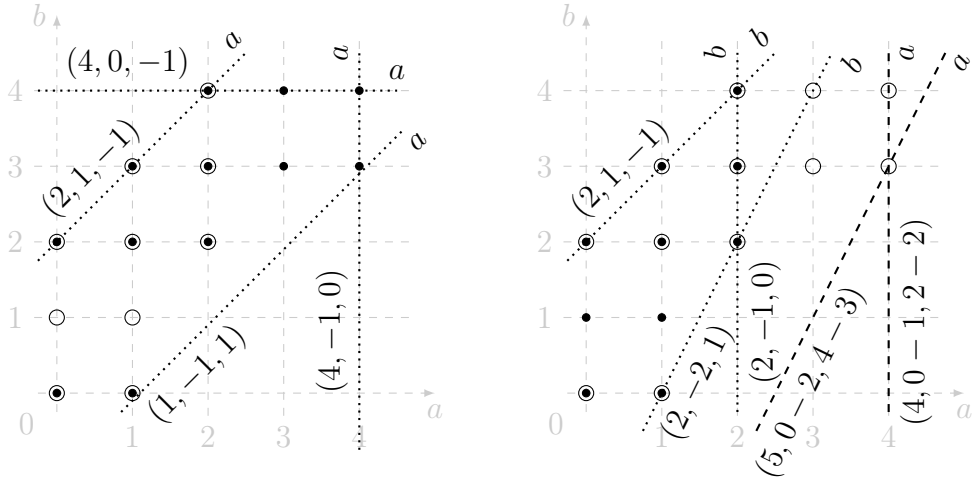


Figure 4.19: L.h.s.: solid dots denote the a-adjacent states, dotted lines form the convex hull of this set of states, r.h.s.: symmetrical under swapping a and b; regions $(5, 0-2, 4-3)$ and $(4, 0-1, 2-2)$ were derived by transforming places corresponding to $(2, -2, 1)$ and $(2, -1, 0)$, respectively, into side-conditions.

4.7.2 Over-approximation with side-conditions

Petri nets with side-conditions are a more powerful class than pure Petri nets. Hence we can achieve better results in the minimal over-approximation of a given language. As was established in Lemma 22, the same region can make different borders for a and b in $\mathbb{N}^2$ when loops around transitions are present.

Let $L = \{abbabaa,\ bbabaabaa\}$ be again the same language as before, and the lts representing it with the projection as in Fig. 4.17. As we have established in Section 4.6, regions are represented as lines in $\mathbb{N}^2$, and these lines are essentially the borders restricting the state space (the arcs cannot cross these borders). According to Lemma 22, being considered as a line, the same region can simultaneously impose for each transition its own border. In order to find an over-approximating Petri net, we first build regions for each letter, a and b, separately. E.g., for b in the example in r.h.s. Fig. 4.19, we construct the regions $(2, 1, -1)$, $(2, -1, 0)$, and $(2, -2, 1)$, represented as dotted lines, which only take states into account that have an adjacent b-edge. These regions, together with the abscissa and the ordinate axis of $\mathbb{N}^2$, form the convex hull for b-adjacent states, but some states only adjacent to a-edges may be outside, precisely states $(3, 3)$, $(4, 3)$, $(3, 4)$, $(4, 4)$. Using the mechanism of Lemma 22, we can adjust the regions $(2, -1, 0)$ and $(2, -2, 1)$,

obtaining new borders (drawn as dashed) for a-edges. Each region (border of the hull) is then translated into a place of a net. The same must be done for the states adjacent to a-edges, taking care of b-edges outside the convex hull. The sought net is obtained as a union of the places derived from the borders of a-adjacent and of b-adjacent states. Algorithm 4 describes this process formally. In order to construct a convex hull, one can use the `Quickhull` [Edd77] algorithm which produces the hull in $O(n^2)$ in the worst case, where n is the size of the initial set. For a language L, we have $n \leq |L| \cdot l$ (l being the length of the longest word). The complexity of the **partialSolution** procedure is $O(n^2)$ in the worst case. Hence, the total complexity of the algorithm does not exceed $O(n^2)$. Applying the algorithm to the language L, we obtain the Petri net on the left hand side of Fig. 4.20, its reachability graph being depicted on the right hand side.

Algorithm 4 Over-approximation of a finite language

Input: finite (prefix-closed) language $L \in \{a, b\}^*$

Output: Petri net over-approximating L

compute sets $W_a = \{(x, y) \mid \Psi(wa) = (x, y) \vee \Psi(w) = (x, y),\ wa \in L\}$

$W_b = \{(x, y) \mid \Psi(wb) = (x, y) \vee \Psi(w) = (x, y),\ wb \in L\}$

find the convex hulls $H_a = ((x_i, y_i))_{0 \le i \le k_a} \subseteq W_a$ of W_a (enumerated clockwise)

$H_b = ((x_j, y_j))_{0 \le j \le k_b} \subseteq W_b$ of W_b (enumerated clockwise)

$(P_a, T, F_a, M_{0,a}) \leftarrow$ **partialSolution**(a, b, W_a, W_b, H_a)

$(P_b, T, F_b, M_{0,b}) \leftarrow$ **partialSolution**(b, a, W_b, W_a, H_b)

$N \leftarrow (P_a \cup P_b, \{a, b\}, F_a \cup F_b, M_{0,a} \cup M_{0,b})$

return N

procedure partialSolution(a, b, W_a, W_b, H)

{construct the net restricting the firings of one transition}

begin procedure

$m \leftarrow |H|$, $P \leftarrow \emptyset$ {find the size m of the hull, define the set of places P}

for $i = 0$ **to** $m - 1$ **do** {construct places from H taking into account W_b}

$r_a^i \leftarrow y_i - y_{i+1}$ {define a region as a line through two points}

$r_b^i \leftarrow x_{i+1} - x_i$ {orthogonal's direction accords with the ordering of H}

$r_0^i \leftarrow -r_a^i \cdot x_i - r_b^i \cdot y_i$

define place p_i

$M_0(p_i) \leftarrow r_0^i$

if $r_a^i \ge 0$

then $F(a, p_i) \leftarrow r_a^i$, $F(p_i, a) \leftarrow 0$

else $F(p_i, a) \leftarrow r_a^i$, $F(a, p_i) \leftarrow 0$

if $r_b^i \ge 0$

then $F(b, p_i) \leftarrow r_b^i$, $F(p_i, b) \leftarrow 0$

else $F(p_i, b) \leftarrow r_b^i$, $F(b, p_i) \leftarrow 0$

if $W_b \setminus W_a \neq \emptyset$

then

$(x', y') \leftarrow \arg\max_{\{(x,y) \in W_b \setminus W_a \mid r_0^i + x' \cdot r_a^i + y' \cdot r_b^i < 0\}} \{|r_0^i + x \cdot r_a^i + y \cdot r_b^i|\}$

{arg max returns the argument yielding the maximum of the set}

$k \leftarrow |x' \cdot r_a^i + y' \cdot r_b^i| - r_0^i$ {define the 'moving factor'}

$M_0(p_i) \leftarrow M_0(p_i) + k$, $F(p_i, a) \leftarrow F(p_i, a) + k$, $F(a, p_i) \leftarrow F(a, p_i) + k$

{adjust the restrictions to include outer states}

add place p_i to P

endfor

return $(P, \{a, b\}, F, M_0)$

end procedure

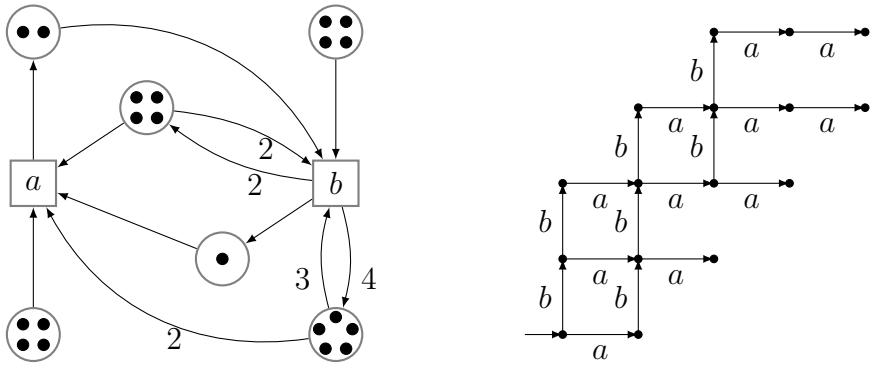


Figure 4.20: The net obtained by the algorithm and its reachability graph.

4.8 Zero g-cycles and language equivalence

The algorithm presented in the previous section for the over-approximating of a given language with a Petri net language translates the initial language into a labelled transition system with all zero g-cycles. We will now show that the restriction of not using of non-zero g-cycles does not influence the result language of the over-approximation procedure, i.e. that for any finite solvable lts TS with non-zero g-cycles there is a solvable general-cycle-neutral lts TS' such that the languages of TS and TS' coincide. To this end we introduce the notion of *language equivalence* of two transition systems. The set $\mathcal{L}(TS) = \{w \in T^* \mid \exists s \in S\colon s_0[w\rangle s\}$ is called the *language of the transition system* TS. Transition systems $TS_1 = (S_1, T, \to_1, s_{01})$ and $TS_2 = (S_2, T, \to_2, s_{02})$ over the same set of labels are called *language equivalent* if their languages are equal, i.e. $\mathcal{L}(TS_1) = \mathcal{L}(TS_2)$.

Lemma 23. *If $TS = (S, T, \to, s_0)$ is solvable, there is a language equivalent transition system TS' which is solvable and satisfies the property* **gc0**.

Proof. If TS satisfies **gc0** then we are done. Otherwise let $N = (P, T, F, M_0)$ be a Petri net with $RG(N) \cong TS$. For every $t \in T$, construct a place p_t such that $F'(t, p_t) = 1$, $F'(p_t, t) = 0$, $M_0'(p_t) = 0$. The net $N' = (P \cup \{p_t \mid t \in T\}, T, F', M_0')$ with $F'(t, p) = F(t, p)$, $F'(p, t) = F(t, p)$ and $M_0'(p) = M_0(p)$ for all $t \in T$, $p \in P$ has the reachability graph $RG(N')$ which is language equivalent to TS. Indeed, since the newly created places do not forbid any firing sequence, the set of firable sequences of N' coincides with the one of N. On the other hand, the new

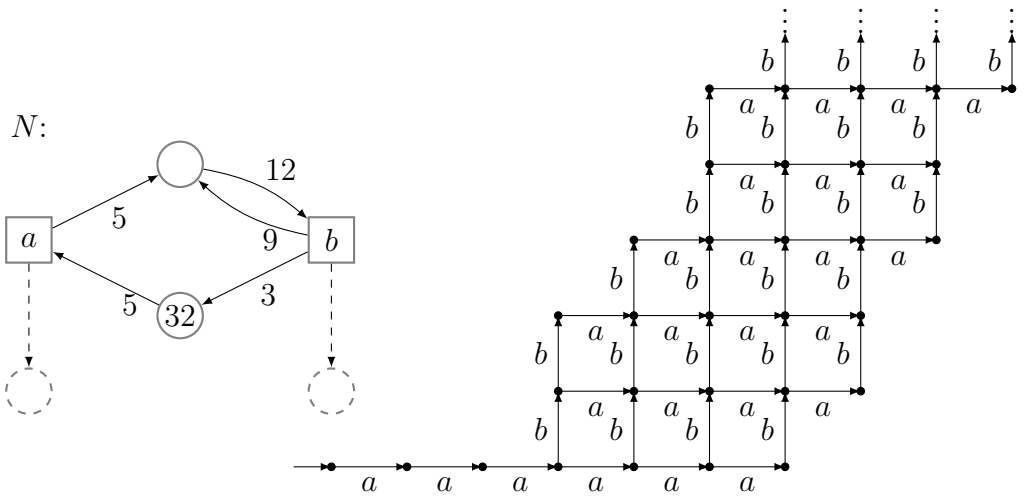


Figure 4.21: An 'unfolded' Petri net with infinite reachability graph.

places distinguish the states of $RG(N')$ which are reached through the execution of Parikh-non-equivalent sequences, implying that $RG(N')$ satisfies **gc0**. □

Remark. For a finite lts which is not **gc0** the 'unfolding' construction of Lemma 23 creates an infinite language equivalent transition system. For instance, the lts from Fig. 4.11, which has the net N (the solid part) from Fig. 4.21 as a possible solution, the lts from Fig. 4.21 will be constructed. This lts is the reachability graph of N with additional places which are depicted as dashed.

4.9 Summary

In the current chapter we continued the investigation of solvability of labelled transition systems. Possibilities for generalising the earlier results, in particular the counting criterion, for a bigger alphabets were discussed. Besides, we dealt with arbitrary transition systems over the binary label set. For the class of finite transition systems that do not satisfy the property **gc0**, i.e. that have non-zero g-cycles, and that are isomorphic to reachability graphs of some Petri nets, a classification was presented. Moreover, a characterisation for the class of labelled transition systems without non-zero generalised cycles, which are synthesisable into Petri nets, was suggested. This characterisation first appeared in [EW17], and it

represents synthesisable state spaces as convex sets of integral points of space. Basing on the characterisation, two algorithms for over-approximating a finite language with a Petri net language were described, for side-condition-free Petri nets and for the general case. The sought Petri net in both of these algorithms is obtained from the convex hull of the points which represent the states of the initial transition system.

Chapter 5

Conclusion

5.1 Summary

We investigated the question of state spaces characterisation in the context of Petri net synthesis. For the class of finite binary sequences a language-theoretical characterisation was presented in the form of the counting criterion. The criterion states that for a given sequence, the solvability of every instance of ESSP can be verified by comparing the fractions of letters within the sequence. A synthesis algorithm `ABSolve`, which is based on the criterion, was presented. The algorithm has a better runtime in comparison to the region-base synthesis algorithms implemented in the tools `APT` and `Synet`. The counting criterion was extended for the cycles, and the synthesis algorithm `ABCycSolve` for this case of input was presented.

We introduced the notion of minimal unsolvable word as a word which is unsolvable and whose proper subwords are solvable. For minimal unsolvable words a classification for a complete enumeration is suggested, which distinguishes two main classes of muws: extendable and non-extendable. Non-extendable muws are characterised by extended regular expressions. In the class of extendable words, the subclass of base extendable ones is considered, which is defined by extended regular expressions; all the extendable words which are not base extendable were proven to be images of base extendable, and they can be derived with the extension morphisms. Another characterisation of the unsolvability of binary sequence was derived from the classification. It was formulated in the form of an extended

regular expression $(abw)b^*(baw)^+a$ whose presence in a sequence is necessary and sufficient for the unsolvability. The presented classification of minimal unsolvable words resulted in the construction of the `Pattern-matching` algorithm for a fast detecting of the unsolvability of a sequence. The algorithm utilises the extended regular expressions from the classification, without initiating the synthesis process itself, and demonstrates a faster runtime than `ABSolve`. It can be used as a pre-synthesis tool for a quick reject of a failure synthesis input.

A graph-theoretical characterisation of the reachability graphs of Petri nets over the binary transition set is presented. The characterisation relies on the notion of generalised cycles. For reachability graphs with non-zero g-cycles eight possible shapes were adduced, with the corresponding forms of Petri nets. A geometrical characterisation for the reachability graphs without non-zero g-cycles was provided. The characterisation establishes that the set of states of such a reachability graph, being projected on $\mathbb{N}^2$, must form a convex set. With the use of the characterisation, an algorithm for (minimal) over-approximating a finite (binary) language by a Petri net language is suggested.

5.2 Outlook

In the case of binary sequences and cycles, the presented characterisations of solvable state spaces demonstrated their utility in synthesis (the counting criterion), as well as in pre-synthesis checking (the classification of muws). But, as it was discussed, trying to get some language-theoretical conditions for the solvability of sequences over alphabets with three and more letters inevitably leads to the theory of regions. Hence, there is no much hope for a significant improvement of the synthesis efficiency in comparison with the region based approach.

Nevertheless, it would be interesting to extend the characterisations to less restricted classes of transition systems. We did the first step in this direction, representing the characterisation of Petri net solvable state spaces as convex sets, and suggesting algorithms for constructing an over-approximation of languages. A natural continuation of this line of work is to use more than two transitions. In these cases the places of the net should be constructed from the hyper-planes which are the borders of the convex hull in the corresponding $|T|$-dimensional space, T being the set of labels of the transition system (the alphabet of the language).

Another possible line of continuation is further developing of pre-synthesis techniques. We have seen that they can be beneficial for a quick reject of unsolvable input on earlier stages of synthesis. This approach could be especially interesting being combined with the question of goal-oriented Petri net synthesis, i.e. synthesis of Petri nets which satisfy some prescribed properties (as boundedness, pureness, being a marked graph, choice-freeness, etc.). In this case, a possibility for a pre-synthesis check whether the required properties can be satisfied for the input, would exclude a verification phase which can happen to be time-consuming.

Bibliography

[AK77] Toshiro Araki and Tadao Kasami. Decidable problems on the strong connectivity of Petri net reachability sets. *Theoretical Computer Science*, 4(1):99 – 119, 1977.

[BB92] Gérard Berry and Gérard Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, 1992.

[BBD15] Eric Badouel, Luca Bernardinello, and Philippe Darondeau. *Petri Net Synthesis*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2015.

[BBE+15] Kamila Barylska, Eike Best, Evgeny Erofeev, Łukasz Mikulski, and Marcin Piątkowski. On binary words being Petri net solvable. In *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data, ATAED 2015, Brussels, Belgium*, volume 1371, pages 1–15. CEUR-WS.org, 2015.

[BBE+16] Kamila Barylska, Eike Best, Evgeny Erofeev, Łukasz Mikulski, and Marcin Piątkowski. Conditions for Petri net solvable binary words. *T. Petri Nets and Other Models of Concurrency*, 11:137–159, 2016.

[BD98] Eric Badouel and Philippe Darondeau. Theory of regions. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 529–586. Springer, 1998.

[BD14a] Eike Best and Raymond Devillers. Persistent Systems with Unique Minimal Cyclic Parikh Vectors. Technical Report 02/14, Dep. Informatik, Carl von Ossietzky Universität Oldenburg, 2014. 80 pages.

[BD14b] Eike Best and Raymond R. Devillers. Characterisation of the state spaces of live and bounded marked graph Petri nets. In *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings*, pages 161–172, 2014.

[BDLM07] Robin Bergenthum, Jörg Desel, Robert Lorenz, and Sebastian Mauser. Process mining based on regions of languages. In *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, pages 375–383, 2007.

[BE16] Eike Best and Javier Esparza. Existence of home states in Petri nets is decidable. *Information Processing Letters*, 116(6):423 – 427, 2016.

[BESW16] Eike Best, Evgeny Erofeev, Uli Schlachter, and Harro Wimmel. Characterising Petri net solvable binary words. In *Application and Theory of Petri Nets and Concurrency - 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings*, pages 39–58, 2016.

[BKMP16] Kamila Barylska, Maciej Koutny, Łukasz Mikulski, and Marcin Piątkowski. Reversible computation vs. reversibility in Petri nets. In *Reversible Computation - 8th International Conference, RC 2016, Bologna, Italy, July 7-8, 2016, Proceedings*, pages 105–118, 2016.

[BMP+16] Kamila Barylska, Łukasz Mikulski, Marcin Piątkowski, Maciej Koutny, and Evgeny Erofeev. Reversing transitions in bounded Petri nets. In *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming, Rostock, Germany, September 28-30, 2016*, pages 74–85, 2016.

[BS15] Eike Best and Uli Schlachter. Analysis of Petri nets and transition systems. In *Proceedings 8th Interaction and Concurrency Experience, ICE 2015, Grenoble, France, 4-5th June 2015*, pages 53–67, 2015.

[BV84] Eike Best and Klaus Voss. Free choice systems have home states. *Acta Informatica*, 21(1):89–100, 1984.

[Cai02] Benoit Caillaud, 2002. The tool Synet, `http://www.irisa.fr/s4/tools/synet`.

[CEP93] Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity results for 1-safe nets. In *Foundations of Software Technology and Theoretical Computer Science, 13th Conference, Bombay, India, December 15-17, 1993, Proceedings*, pages 326–337, 1993.

[CHEP71] F. Commoner, A.W. Holt, S. Even, and A. Pnueli. Marked directed graphs. *Journal of Computer and System Sciences*, 5(5):511 – 523, 1971.

[CL11] Luca Cardelli and Cosimo Laneve. Reversible structures. In François Fages, editor, *Proceedings of 9th International Computational Methods in Systems Biology (CMSB'11)*, pages 131–140. ACM, 2011.

[CLRS09] Thomas H. Cormen, Charles Eric Leiserson, Ronald L. Rivest, and Clifford Stein, editors. *Introduction to algorithms*. MIT Press, third edition, 2009.

[Dan63] George B. Dantzig. *Linear programming and extensions*. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ, 1963.

[DK04] Vincent Danos and Jean Krivine. Reversible communicating systems. In *Proceedings of 15th International Conference on Concurrency Theory (CONCUR'04)*, volume 3170 of *Lecture Notes in Computer Science*, pages 292–307. Springer-Verlag (New York), 2004.

[DK05] Vincent Danos and Jean Krivine. Transactions in RCCS. In *Proceedings of 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of *Lecture Notes in Computer Science*, pages 398–412. Springer-Verlag (New York), 2005.

[DKS07] Vincent Danos, Jean Krivine, and Pawel Sobocinski. General reversibility. *Electronic Notes Theoretical Computer Science*, 175(3):75–86, 2007.

[EBMP16] Evgeny Erofeev, Kamila Barylska, Łukasz Mikulski, and Marcin Piątkowski. Generating all minimal Petri net unsolvable binary words. In *Proceedings of the Prague Stringology Conference 2016, Prague, Czech Republic, August 29-31, 2015*, pages 33–46, 2016.

[Edd77] William F. Eddy. A new convex hull algorithm for planar sets. *ACM Trans. Math. Softw.*, 3(4):398–403, 1977.

[ER90] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Partial (set) 2-structures. *Acta Informatica*, 27(4):343–368, 1990.

[Esp98] Javier Esparza. Reachability in live and safe free-choice Petri nets is NP-complete. *Theoretical Computer Science*, 198(1):211 – 224, 1998.

[EW17] Evgeny Erofeev and Harro Wimmel. Reachability graphs of two-transition Petri nets. In *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data, ATAED 2017, Zaragoza, Spain*, pages 39–54, 2017.

[Gra72] R.L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132 – 133, 1972.

[GRX02] Asma Ghaffari, Nidhal Rezg, and Xiaolan Xie. Maximally permissive and non-blocking control of Petri nets using theory of regions. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*, pages 1895–1900, 2002.

[HDK15] Thomas Hujsa, Jean-Marc Delosme, and Alix Munier Kordon. On the reversibility of live equal-conflict Petri nets. In *Application and Theory of Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings*, pages 234–253, 2015.

[Jar73] R.A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18 – 21, 1973.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[KCK+95] Alex Kondratyev, Jordi Cortadella, Michael Kishinevsky, Enric Pastor, Oriol Roig, and Alexandre Yakovlev. Checking signal transition graph implementability by symbolic BDD traversal. In *European Design and Test Conference, ED&TC 1995, Paris, France*, pages 325–332, 1995.

[Kos82] S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 267–281, New York, NY, USA, 1982. ACM.

[KPP06] Danko Kezić, Nedjeljko Peric, and Ivan Petrovic. An algorithm for deadlock prevention based on iterative siphon control of Petri nets. *AUTOMATIKA: Journal for Control, Measurement, Electronics, Computing and Communications*, 47(1-2):19–30, 2006.

[Lam92] J. L. Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99(1):79–104, 1992.

[LMS10] Ivan Lanese, Claudio Antares Mezzina, and Jean-Bernard Stefani. Reversing higher-order Pi. In *Proceedings of 21th International Conference on Concurrency Theory (CONCUR'10)*, volume 6269 of *Lecture Notes in Computer Science*, pages 478–493. Springer, 2010.

[LPK00] GNU Linear Programming Kit, 2000. `https://www.gnu.org/software/glpk/`.

[LR78] Lawrence H. Landweber and Edward L. Robertson. Properties of conflict-free and persistent Petri nets. *J. ACM*, 25(3):352–364, 1978.

[May81] Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, pages 238–246, New York, NY, USA, 1981. ACM.

[MKMH86] Tomohiro Murata, Norihisa Komoda, Kuniaki Matsumoto, and Koichi Haruna. A Petri net-based controller for flexible and maintainable sequence control and its applications in factory automation. *IEEE Transactions on Industrial Electronics*, 33(1):1 – 8, 1986.

[Mur89] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[nlA06] Sergey Bochkanov. The numerical library ALGLIB, 2006. `http://www.alglib.net`.

[ÖA08] Hanife Apaydin Özkan and Aydın Aybar. A reversibility enforcement approach for Petri nets using invariants. *Wseas Transaction on Systems*, 7:672–681, 2008.

[P+15] Marcin Piatkowski et al., 2015. A list of unsolvable words, `http://folco.mat.umk.pl/unsolvable-words`.

[PU07] Iain Phillips and Irek Ulidowski. Reversing algebraic process calculi. *Journal of Logic and Algebraic Programming*, 73(1-2):70–96, 2007.

[PU15] Iain Phillips and Irek Ulidowski. Reversibility and asymmetric conflict in event structures. *Journal of Logic and Algebraic Methods in Programming*, 84(6):781–805, 2015.

[PUY12] Iain Phillips, Irek Ulidowski, and Shoji Yuen. A reversible process calculus and the modelling of the ERK signalling pathway. In *Proceedings of 4th Workshop on Reversible Computation (RC'12)*, volume 7581 of *Lecture Notes in Computer Science*, pages 218–232. Springer, 2012.

[Rei13] Wolfgang Reisig. *Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies*. Springer, 2013.

[S+13] Uli Schlachter et al., 2013. The tool APT, `http://github.com/CvO-Theory/apt`.

[vdA00] Wil M. P. van der Aalst. Workflow verification: Finding control-flow errors using Petri-net-based techniques. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 161–183, 2000.

[vdAG07] Wil M. P. van der Aalst and Christian W. Günther. Finding structure in unstructured processes: The case for process mining. In *Proceedings of the Seventh International Conference on Application of Concurrency to System Design*, ACSD '07, pages 3–12, Washington, DC, USA, 2007. IEEE Computer Society.

[WDC11] Pengwei Wang, Zhijun Ding, and Hua Chai. An algorithm for generating home states of Petri nets. *Journal of Computational Information Systems*, 7(12):4225–4232, 2011.

[Yen91] Hsu-Chun Yen. A polynomial time algorithm to decide pairwise concurrency of transitions for 1-bounded conflict-free Petri nets. *Information Processing Letters*, 38(2):71 – 76, 1991.

[YK98] Alexandre V. Yakovlev and Albert M. Koelmans. Petri nets and digital hardware design. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets II: Applications: Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 154–236, Berlin, Heidelberg, 1998. Springer.

Index

Appendix

A list of minimal unsolvable binary words

A list of minimal unsolvable words up to length 20 modulo swapping a and b. Notation L = n means the length of the words in the corresponding group. Since the shortest unsolvable word is *abbaa*, the list begins with length 5.

[L = 5] :
abbaa

[L = 6] :
abbbaa

[L = 7] :
ababaaa
abbabaa
abbbaba
abbbbaa

[L = 8] :
abbbbaba
abbbbbaa

[L = 9] :
abaabaaaa
abbababaa
abbbabbba
abbbbbabba
abbbbbbaa

[L = 10] :
ababaabaaa
abbbabbaba
abbbbbabba
abbbbbbaba
abbbbbbbaa

[L = 11] :
abaaabaaaaa
abababaabaa
abbababababa
abbabbababa
abbbbbabbba
abbbbbbabba
abbbbbbbaba
abbbbbbbbaa

[L = 12] :
abbbbbbabbba
abbbbbbbabba
abbbbbbbbaba
abbbbbbbbbaa

[L = 13] :
abaaaabaaaaaa
abaabaaabaaaa
ababaabaabaaa
abbababababaa
abbbabbabbaba
abbbbabbbabba
abbbbbbabbbba
abbbbbbbabbba
abbbbbbbbabba
abbbbbbbbbaba
abbbbbbbbbbaa

[L = 14] :
abbbbbbbabbbba
abbbbbbbbabbba
abbbbbbbbbabba
abbbbbbbbbbaba
abbbbbbbbbbbaa

[L = 15] :
abaaaaabaaaaaaa
abaabaabaaabaaa
abababababaabaa
abbababababababa
abbababbabababa
abbbbabbbabbbba
abbbbbbbbabbbbba
abbbbbbbbbabbbba
abbbbbbbbbbabbba
abbbbbbbbbbbbaa

[L = 16] :
abaaabaaaabaaaaa
ababaabaabaabaaa
abababaababaabaa
abbabbababbababa
abbbabbabbabbaba
abbbbbabbbbabbba
abbbbbbbbabbbbba
abbbbbbbbbabbbba
abbbbbbbbbbabbba
abbbbbbbbbbbbaba
abbbbbbbbbbbbbaa

[L = 17] :
abaaaaaabaaaaaaaa
abaabaaabaaabaaaa
ababaababaabaabaa
abbababababababaa
abbabbabbababbaba
abbbbabbbabbbbba
abbbbbbbbabbbbbba
abbbbbbbbbabbbbba
abbbbbbbbbbabbbba

abbbbbbbbbbbabbba
abbbbbbbbbbbbabba
abbbbbbbbbbbbbaba
abbbbbbbbbbbbbbaa

[L = 18] :
abbbbbbbbbabbbbbba
abbbbbbbbbbabbbbba
abbbbbbbbbbbabbbba
abbbbbbbbbbbbabbba
abbbbbbbbbbbbbabba
abbbbbbbbbbbbbbaba
abbbbbbbbbbbbbbbaa

[L = 19] :
abaaaaaaabaaaaaaaaa
abaaaabaaaaabaaaaaa
abaaabaaabaaaabaaaa
ababaabaabaabaabaaa
abababababaababababaa
abbababababababababaa
abbababababbabababaa

abbbabbabbabbabbaba
abbbbabbbbabbbabbba
abbbbbbabbbbbabbbba
abbbbbbbbbabbbbbbba
abbbbbbbbbbabbbbbba
abbbbbbbbbbbabbbbba
abbbbbbbbbbbbabbbba
abbbbbbbbbbbbbabbba
abbbbbbbbbbbbbbaba
abbbbbbbbbbbbbbbaa

[L = 20] :
abbbbbbbbbbbabbbbbba
abbbbbbbbbbbbabbbbba
abbbbbbbbbbbbbabbbba
abbbbbbbbbbbbbbabbba
abbbbbbbbbbbbbbbabba
abbbbbbbbbbbbbbbbaba
abbbbbbbbbbbbbbbbbaa