Characterising Petri Net Solvable Binary Words

Eike Best, Evgeny Erofeev, Uli Schlachter, and Harro Wimmel(✉)

Parallel Systems, Department of Computing Science,
Carl von Ossietzky Universität, D-26111 Oldenburg, Germany
{eike.best,evgeny.erofeev,uli.schlachter,
harro.wimmel}@informatik.uni-oldenburg.de

Abstract. A word is called Petri net solvable if it is isomorphic to the reachability graph of an unlabelled Petri net. In this paper, the class of finite, two-letter, Petri net solvable words is studied. A linear time, necessary condition allows for an educated guess at which words are solvable and which are not. A full decision procedure with a time complexity of $O(n^2)$ can be built based on letter counting. The procedure is fully constructive and can either yield a Petri net solving a given word or determine why this fails. Algorithms solving the same problem based on systems of integer inequalities reflecting the potential Petri net structure are only known to be in $O(n^3)$. Finally, the decision procedure can be adapted from finite to cyclic words.

Keywords: Binary words · Labelled transition systems · Petri nets · Synthesis

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) reachability graph serves to facilitate its behavioural analysis [9]. 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 [7]. 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 labelled transition system [1]. We may get structural results of a different nature in this case. For example, [4] describes a structural characterisation of the class of marked graph reachability graphs in terms of a carefully chosen list of graph-theoretical properties.

This research has been supported by DFG (German Research Foundation) through grants Be 1267/15-1 ARS (Algorithms for Reengineering and Synthesis), Be 1267/14-1 CAVER (Comparative Analysis and Verification for Correctness-Critical Systems), and Graduiertenkolleg GRK-1765 SCARE (System Correctness under Adverse Conditions).

F. Kordon and D. Moldt (Eds.): PETRI NETS 2016, LNCS 9698, pp. 39–58, 2016.
DOI: 10.1007/978-3-319-39086-4_4

Region theory [1] establishes an indirect characterisation of the class of Petri net reachability graphs. This characterisation essentially consists of an algorithm solving many systems of linear inequalities derived from a given transition system [5,10]. By its linear-algebraic nature, it provides little insight into the structural properties of Petri net reachability graphs. The aim of the present paper is to complement this indirect characterisation by a direct one, and to show that such a direct characterisation can lead to different, time-efficient, algorithms for checking synthesisability. However, we shall limit ourselves to a special class of transition systems, namely to finite, non-branching ones having at most two edge labels. That is, we study the class of binary, finite or cyclic words (possibly with some finite initial part). We shall obtain a characterisation of the Petri net synthesisable ones amongst them, along with corresponding algorithms.

In a first step, we shall develop a necessary criterion that must hold for finite, binary, synthesisable words. This will frequently allow us to spot non-synthesisable words in linear time. In a second step, we shall provide characterisations of binary, synthesisable words in the finite as well as in the cyclic case with a quadratic time complexity, allowing for a faster decision procedure than via the region based approach. More specifically, the structure of the paper is as follows. Section 2 contains some basic definitions about labelled transition systems, Petri nets, and regions. Section 3 describes properties of synthesisable words leading to a necessary criterion for synthesisability. Sections 4 and 5 characterise synthesisable word in the finite case and in the cyclic case, respectively. Section 6 compares an implementation of our results with the region based algorithms of Synet [5] and APT [10]. Section 7 concludes the paper.

2 Basic Concepts, and Region-Based Synthesis

2.1 Transition Systems, Words, and Petri Nets

A *finite labelled transition system* with initial state is a tuple $TS = (S, \rightarrow, T, s_0)$ with nodes S (a finite set of states), edge labels T (a finite set of letters), edges $\rightarrow \subseteq (S \times T \times S)$, and an initial state $s_0 \in S$. A label t is enabled at $s \in S$, denoted by $s[t\rangle$, if $\exists s' \in S\colon (s,t,s') \in \rightarrow$. A state s' is reachable from s through the execution of $\sigma \in T^*$, 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 $s[\sigma\rangle s'$. We use $\sigma|_s\sigma'$ as an abbreviation for $s_0[\sigma\rangle s[\sigma'\rangle$. Two labelled transition systems $TS_1 = (S_1, \rightarrow_1, T, s_{01})$ and $TS_2 = (S_2, \rightarrow_2, T, s_{02})$ 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 $s, s' \in S_1$.

A *word over* T is a sequence $w \in T^*$, and it is *binary* if $|T| = 2$. For a word w and a letter t, $\#_t(w)$ denotes the number of times t occurs in w. A word $w' \in T^*$ is called a *subword* (or *factor*) of $w \in T^*$ if $\exists u_1, u_2 \in T^*\colon w = u_1 w' u_2$. A word $w = t_1 t_2 \ldots t_n$ of length $n \in \mathbb{N}$ uniquely corresponds to a finite transition system $TS(w) = (\{0, \ldots, n\}, \{(i-1, t_i, i) \mid 0 < i \leq n \wedge t_i \in T\}, T, 0)$.

An *initially marked Petri net* is denoted as $N = (P, T, F, M_0)$ where P is a finite set of places, T is a finite set of transitions with $P \cap T = \emptyset$, F is the flow function $F\colon ((P \times T) \cup (T \times P)) \to \mathbb{N}$ specifying the arc weights, and M_0 is the initial marking (where a marking is a mapping $M\colon P \to \mathbb{N}$, indicating the number of tokens in each place). A side-place is a place p with $p^\bullet \cap {}^\bullet p \neq \emptyset$, where $p^\bullet = \{t \in T \mid F(p,t){>}0\}$ and ${}^\bullet p = \{t \in T \mid F(t,p){>}0\}$. A transition $t \in T$ is enabled at a marking M, denoted by $M[t\rangle$, if $\forall p \in P\colon M(p) \geq F(p,t)$. The firing of t leads from M to M', denoted by $M[t\rangle M'$, if $M[t\rangle$ and $M'(p) = M(p) - F(p,t) + F(t,p)$. This can be extended, as usual, to $M[\sigma\rangle M'$ for sequences $\sigma \in T^*$, and $[M\rangle$ denotes the set of markings reachable from M. The reachability graph $RG(N)$ of a bounded (such that the number of tokens in each place does not exceed a certain finite number) Petri net N is the labelled transition system with the set of vertices $[M_0\rangle$, initial state M_0, label set T, and set of edges $\{(M,t,M') \mid M, M' \in [M_0\rangle \wedge M[t\rangle M'\}$. 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, and that TS is *synthesisable* to N. We say that N solves a word w if it solves $TS(w)$. We frequently identify the states of TS with the markings of N then, writing e.g. $s(p) \geq F(p,t)$.

2.2 Basic Region Theory, and an Example

Let a finite labelled transition system $TS = (S, \to, T, s_0)$ be given. In order to synthesise – if possible – a Petri net with isomorphic reachability graph, T must, of course (since we do not consider any transition labels), be used directly as the set of transitions. For the places, $\frac{1}{2}\cdot(|S|\cdot(|S|{-}1))$ state separation problems and up to $|S|\cdot|T|$ event/state separation problems have to be solved, as follows:

- A *state separation problem* consists of a set of states $\{s, s'\}$ with $s \neq s'$ where s and s' must be mapped to different markings in the synthesised net. Such problems are always solvable if $TS = TS(w)$ originates from a word w, for instance by introducing a counting place which has j tokens in state j.
- An *event/state separation problem* consists of a pair $(s,t) \in S{\times}T$ with $\neg(s[t\rangle)$. For every such problem, one needs a place p such that $M(p) < F(p,t)$ for the marking M corresponding to state s, where F refers to the arcs of the hoped-for net.

For example, in Fig. 1, TS_1 is PN-solvable, since the reachability graph of N_1 is isomorphic to TS_1. Note that N_1 has exactly two transitions a and b, which is true for any net solving a binary word over $\{a, b\}$. By contrast, TS_2 is not PN-solvable. The word *abbaa*, from which TS_2 is derived, is actually one of the two shortest non-solvable binary words (the other one being *baabb*, its dual under swapping a and b).

To see that *abbaa* (viz., TS_2) is not PN-solvable, we may use the following argument. State $s = 2$ generates an event/state separation problem $\neg(s[a\rangle)$, for which we need a place q whose number of tokens in the marking corresponding to state 2 is less than necessary for transition a to be enabled. Such a place q

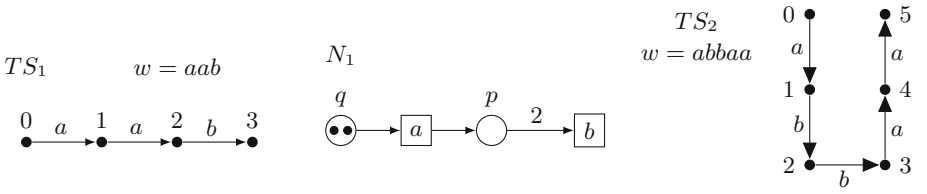

Fig. 1. TS_1 and TS_2 correspond to *aab* and *abbaa*, respectively. N_1 solves TS_1. No Petri net solution of TS_2 exists.

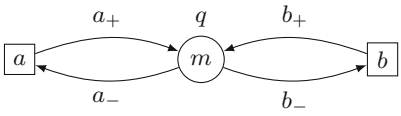

Fig. 2. A place with four arc weights a_-, a_+, b_-, b_+ and initial marking m

has the general form shown in Fig. 2. We now show that such a place does not exist.

In order to present this proof succinctly, it is useful to define the *effect* $\mathbb{E}(\tau)$ of a sequence $\tau \in T^*$ on place q. The effect of the empty sequence is $\mathbb{E}(\varepsilon) = 0$. The effect of a sequence $a\tau$ is defined as $\mathbb{E}(a\tau) = (a_+ - a_-) + \mathbb{E}(\tau)$, and similarly, $\mathbb{E}(b\tau) = (b_+ - b_-) + \mathbb{E}(\tau)$. For instance, $\mathbb{E}(abbaa) = 3{\cdot}(a_+ - a_-) + 2{\cdot}(b_+ - b_-)$. In general, $\mathbb{E}(\tau) = \#_a(\tau) \cdot \mathbb{E}(a) + \#_b(\tau) \cdot \mathbb{E}(b)$.

If q (as in Fig. 2) prevents a at the marking corresponding to state 2 in *abbaa* (cf. TS_2 in Fig. 1), then it must satisfy the following inequalities: $a_- \leq m$, since state 0 enables a; $a_- \leq m + \mathbb{E}(abba)$, since state 4 enables a; $m + \mathbb{E}(ab) < a_-$, since q prevents a at state 3. Using $\mathbb{E}(abba) = \mathbb{E}(ab) + \mathbb{E}(ab)$, it is immediate to see that this set of inequalities cannot be solved in the natural numbers.

2.3 Worst Case Complexity of the General Algorithm

In a word of length n, the equation system for a single event/state separation problem comprises $n + 1$ inequalities, n for the states $0, \ldots, n - 1$ and one for the event/state separation. In binary words, we have $n + 2$ such problems, one for every state $0, \ldots, n - 1$ and two for the last state. A word w of length n is PN-solvable if and only if all $n + 2$ systems, each having $n + 1$ inequalities and five unknowns a_-, a_+, b_-, b_+, m, are solvable in $\mathbb{N}$.

Suppose that we solve this special case (with five unknowns) by Khachiyan's algorithm [6]. Solving $O(n)$ systems of inequalities, we may roughly expect a running time of $O(n^3)$.

3 Necessary Conditions for Solvability

As a first step of characterising solvable words over $\{a, b\}^*$, and quoting various related partial results from [2], we develop a necessary criterion with a linear time complexity. From this we will get a good idea of how solvable words are structured and can easily sort out the majority of unsolvable words just by looking at them.

Proposition 1. [2] Solvability of Subwords.
If $w = xvy$ with $x, y \in \{a, b\}$ is solvable, then xv and vy are also solvable.

The reverse does not hold, of course, otherwise there would be no unsolvable words at all. With $x \neq y$ though, solvability can be propagated:

Proposition 2. [2] Solvability of awb from aw and wb.
If both aw and wb are solvable, then awb is also solvable.

This also holds for bw and wa, of course. It is also possible to prefix a solvable word by its first letter.

Proposition 3. [2] Prefixing Solvable Words by their First Letter.
Let v be a solvable word starting with a letter $x \in \{a, b\}$. Then, xv is solvable.

An unsolvable word w is *minimal* if all subwords of w are solvable. For this, it is sufficient that for $w = xvy$ with $x, y \in \{a, b\}$ both xv and vy are solvable. So, due to Proposition 2, minimal unsolvable words must start and end with the same letter. They are also restricted to which subwords aa or bb can be contained:

Proposition 4. [2] Never aa and bb inside a minimal unsolvable word.
If a minimal non-solvable word is of the form $u = a\alpha a$, then either α does not contain the factor (subword) aa or α does not contain the factor bb.

Propositions 3 and 4 together now also suggest a restriction for solvable words. 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 5. 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 "neighboring" 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 neighbors 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 $w = \beta|_{s'} ba^i a|_s a(ba)^j b|_{s''} b\gamma$. Since b cannot fire at s, there must be a place $p \in P$ with $s(p) < b_-$ (compare Fig. 2). At s' and s'' the transition b can fire, so $s'(p) \geq b_- \leq s''(p)$ holds. As firing a enables b again after s, a must produce tokens on p and $\mathbb{E}(a) > 0$. Since b does not remain enabled from s' to s, it has to consume tokens from p, so $\mathbb{E}(b) < 0$. Computing the token differences on p between our chosen states we then obtain

$$0 > s(p) - s'(p) = (i+1) \cdot \mathbb{E}(a) + \mathbb{E}(b) \text{ and}$$
$$0 < s''(p) - s(p) = (j+1) \cdot \mathbb{E}(a) + (j+1) \cdot \mathbb{E}(b).$$

Comparing the lines gives $\mathbb{E}(a) > -\mathbb{E}(b) > (i+1) \cdot \mathbb{E}(a)$, which is a contradiction, i.e. w is not solvable. □ 5

This reduces the potentially solvable words to the regular expression

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

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, of course.

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. Block Lengths Differ by at most 1.

Let $w \in a^*b^+(ab^+)^*(a|\varepsilon)$ be a word that contains both bab^ia and abb^ib with $i \geq 1$ as subwords. Then, w is not solvable.

Proof: Consider first the case $w = \alpha|_s bab^i|_{s'} (abb^i)^k abb^i|_{s''} b\beta$ 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 $s'(p) < b_- \leq s(p)$, i.e. $\mathbb{E}(bab^i) = s'(p) - s(p) < 0$. Due to Parikh equivalence, $\mathbb{E}(bab^i) = \mathbb{E}(abb^i)$, we know $s''(p) = s'(p) + (k+1) \cdot \mathbb{E}(abb^i) < s'(p) < b_-$, which is a contradiction to b being enabled at s''.

The second case, $w = \alpha|_s abb^ib^j(bab^i)^k|_{s'} bab^i|_{s''} a\beta$ 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 to be a place with $s'(p) < a_- \leq s''(p)$, then with $s''(p) - s'(p) = \mathbb{E}(bab^i) = \mathbb{E}(abb^i) > 0$ and $\mathbb{E}(b) > 0$ (since firing b at s' enables a), we obtain $s(p) = s'(p) - k \cdot \mathbb{E}(bab^i) - j \cdot \mathbb{E}(b) - \mathbb{E}(abb^i) < s'(p) < a_-$. This contradicts a being enabled at s. □ 1

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

Proposition 6. Balancing Property.
Let $w = a^k b^{x_1} a b^{x_2} \dots a b^{x_n}$ 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. □ 6

The first block of b's can have arbitrary length (e.g., both $abab^9ab^9ab^9a$ and $b^9abbabbabba$ are solvable). 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; e.g. $ab^9ab^9ab^9ab$ is solvable while $abababab^9$ is not. In the former case, we may even append some more b's.

Lemma 2. Prolonging the Last b-Block.
Let $w = a^k b^{x_1} a b^{x_2} a \dots a b^{x_n}$ be a solvable word with $k \geq 0$ and $x_i - 1 > x_n$ for all $1 \leq i < n$. Then, $w' = a^k b^{x_1} a b^{x_2} a \dots a b^{x_n+1}$ is solvable.

Proof: Consider the case $k \leq 1$ first. Assume $N = (P, T, F, M_0)$ to be a Petri net solving $w = a^k b^{x_1} a b^{x_2} \dots a b^{x_{i-1}}|_{s'} a b^{x_i-1}|_{s''} b a \dots b|_s a b^{x_n}|_f$ with a place p that prevents b at some s' before s. Then, $s'(p) < b_-$ and $s''(p) = s'(p) + \mathbb{E}(ab^{x_i-1}) \geq b_-$, i.e. $\mathbb{E}(ab^{x_i-1}) > 0$. With $s(p) \geq b_+$ (b fires directly before s) and $\mathbb{E}(b) < 0$ (b fires directly before s'), we conclude $f(p) = s(p) + \mathbb{E}(ab^{x_n}) \geq b_+ + \mathbb{E}(ab^{x_n}) = b_+ + \mathbb{E}(ab^{x_i-1}) - \mathbb{E}(b^{x_i-1-x_n}) > b_+ - (x_i - 1 - x_n) \cdot \mathbb{E}(b) \geq b_+ - \mathbb{E}(b) = b_-$. 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 q with $b_- = 1$, $b_+ = 0$, $a_- = 0$, $a_+ = \min\{x_1, \dots, x_{n-1}\}$, and an initial token count of $(\sum_{i=1}^{n-1} x_i) - (n - 2 + k) \cdot \min\{x_1, \dots, x_{n-1}\}$ (which is non-negative). Then, $s(q) = 0$ and $f(q) = s(q) + a_+ - x_n > 0$. A place preventing a (except after the last a) must have $\mathbb{E}(b) > 0$, 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. □ 2

Deleting one b from each block of b's will also not turn a word unsolvable.

Lemma 3. 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 $h(b) = b$, we get $h(w') = w$.

Let N be a Petri net solving w. For each place p with arc weights a_+, a_-, b_+, and b_- let $i_p := \max\{0, -a_+ - \mathbb{E}(b)\}$ 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 $a'_+ := a_+ + \mathbb{E}(b) + i_p$. In all cases, $a'_+ - a'_- = \mathbb{E}(ab)$ and $b'_+ - b'_- = \mathbb{E}(b)$ 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 corresponding prefix $h(v)$ of w plus the additional $(i_p)_p$. We conclude that w' and only w' can fire in N', i.e. N' solves w'. □ 3

This lemma suggests that comparing the lengths of b-blocks are more important for solvability than computing their absolute lengths. Our necessary criterion, being a summary of the results of this section, establishes this intuition more formally as follows:

Theorem 1. Linear Time Necessary Criterion.
If a word $w \in \{a,b\}^*$ is solvable, it is the empty word $w = \varepsilon$ 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. E.g., *abababbabba*, *ababbababba*, *ababbabbaba*, *abbababbaba*, and *abbababababba* are all solvable while *abbabbababa* is not. One could suspect that the high number of early b's makes the latter word unsolvable. This will be made precise in the following section.

4 Characterisation of Solvable Binary Words

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

Theorem 2. Characterisation of Solvable Words.

A word $w \in \{a,b\}^*$ is solvable if and only if the following formula holds for $x = a \wedge y = b$ as well as for $x = b \wedge y = a$:

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

Proof: We need to show that for any decomposition $w = u|_s xv$ with $x \in \{a,b\}$, the other letter $y \neq x$, $y \in \{a,b\}$ is separable at s if and only if the above formula holds for all decompositions of $u = \alpha y \beta$ and $v = \gamma y \delta$. We outsource this proof to Lemma 4. Disabling a and b at the end of the word is trivially done by putting $|w|$ tokens on a new place, from which each transition takes one token upon firing. □ 2

Lemma 4. 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: "⇒": 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. $\mathbb{E}(x) > 0$.

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. $\mathbb{E}(y) < 0$.

Since y can fire at s' but not at s, tokens are consumed by $y\beta$, i.e. $\#_y(y\beta) \cdot (-\mathbb{E}(y)) > \#_x(y\beta) \cdot \mathbb{E}(x)$. From s to s'', for analogous reasons, tokens are produced on p, so $\#_x(x\gamma) \cdot \mathbb{E}(x) > \#_y(x\gamma) \cdot (-\mathbb{E}(y))$. Multiply the first inequality by $\#_x(x\gamma)$ and the second one by $\#_x(y\beta)$, then divide both by $-\mathbb{E}(y)$ to make them comparable:

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

"⇐": Let $S' := \{s' \mid \exists \alpha, \beta : w = \alpha|_{s'} y\beta|_s xv\}$ and $S'' := \{s'' \mid \exists \gamma, \delta : w = u|_s x\gamma|_{s''} y\delta\}$. 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 be zero (as there is an x directly after s and a y after s' and s''), so no ambiguos fraction $\frac{0}{0}$ can occur. If $\#_x(s', s)$ is zero, we assume the (obvious) default value of ∞ for this fraction. See Fig. 3 for a visualisation.

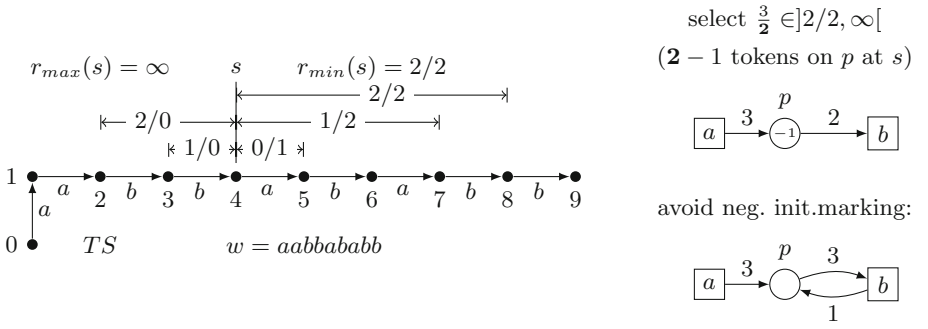

Fig. 3. TS corresponding to *aabbababb* with a state s at which b must not occur. We compute maximal/minimal b/a-ratios $r_{min}(s)/r_{max}(s)$ for words starting with b ending at s and starting at s ending in front of a b, respectively. The production/consumption ratio for a place p in a Petri net prohibiting 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.

We now show 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 $w = \alpha|_{s'} y\beta|_{s} x\gamma|_{s''} y\delta$ with $s' \in S'$ and $s'' \in S''$, we get $\#_y(y\beta) \cdot \#_x(x\gamma) > \#_x(y\beta) \cdot \#_y(x\gamma)$. 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 $S' \cup S''$. 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 obviously possible; compare Fig. 3. 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 trivially 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.

Analogously, 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 $s(p) = y_- - 1$ 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. □ 4

Proposition 7. 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 the arc weight ratio $\frac{x_+}{y_-}$ of the occurring letter x compared to the separation letter y must lie inside the open interval. 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. □ 7

Let us take a look at the word $w = aabbabab$ from Fig. 3 again. For states followed by an a we get $r_{min}(0) = \max\{\frac{0}{2}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{4}\} = 1$, $r_{max}(0) = \infty$ with the interval $]1, \infty[$, $r_{min}(1) = \max\{\frac{0}{1}, \frac{1}{1}, \frac{2}{2}, \frac{3}{3}, \frac{4}{3}\} = \frac{4}{3}$, $r_{max}(1) = \infty$ with the interval $]\frac{4}{3}, \infty[$, $r_{min}(4) = \max\{\frac{0}{1}, \frac{1}{2}, \frac{2}{2}\} = 1$, $r_{max}(4) = \min\{\frac{2}{0}, \frac{1}{0}\} = \infty$ with $]1, \infty[$, and $r_{min}(6) = \max\{\frac{0}{1}, \frac{1}{1}\} = 1$, $r_{max}(6) = \min\{\frac{3}{1}, \frac{2}{1}, \frac{1}{0}\} = 3$ with $]1, 3[$. The value $\frac{3}{2}$ lies in all open intervals, so we get one place p with $\mathbb{E}(a) = 3$ and $\mathbb{E}(b) = -2$ and at most $2-1 = 1$ tokens on it at each of the four states. Backward calculation of the initial state gives 1, -2, -1, and -2 tokens for the states 0, 1, 4, and 6, respectively. We set $b_- = 2 + 2$, $b_+ = 2$ to obtain zero tokens in the initial marking. For states followed by b we have $r_{min}(2) = \max\{\frac{0}{2}, \frac{1}{3}\} = \frac{1}{3}$, $r_{max}(2) = \min\{\frac{2}{0}, \frac{1}{0}\} = \infty$, $r_{min}(3) = \max\{\frac{0}{1}, \frac{1}{2}\} = \frac{1}{2}$, $r_{max}(3) = \min\{\frac{2}{1}, \frac{1}{1}\} = 1$, $r_{min}(5) = \max\{\frac{0}{1}\} = 0$, $r_{max}(5) = \min\{\frac{3}{2}, \frac{2}{2}, \frac{1}{0}\} = 1$, $r_{min}(7) = -\infty$, $r_{max}(7) = \min\{\frac{4}{3}, \frac{3}{3}, \frac{2}{1}, \frac{1}{0}\} = 1$, $r_{min}(8) = -\infty$, $r_{max}(8) = \min\{\frac{4}{4}, \frac{3}{4}, \frac{2}{2}, \frac{1}{1}\} = \frac{3}{4}$. We are inside all intervals if we choose $\frac{2}{3} \in]\frac{1}{2}, \frac{3}{4}[$ for a new place q with $\mathbb{E}(b) = 2$, $\mathbb{E}(a) = -3$, and at most $3 - 1 = 2$ tokens at any of these states. We compute for the initial marking 8, 6, 7, 7, 6 tokens (for the five states), so by the proof of Lemma 4 six initial tokens are enough to enable a where it occurs in w, but not anywhere else. Adding a place f with 9 tokens to prevent a and b at the end, we obtain the net in Fig. 4.

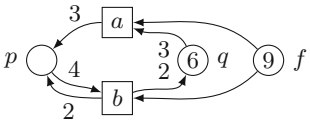

Fig. 4. A Petri net solving the word *aabbabab*

In all examples we examined so far, all the open intervals $]r_{min}(s), r_{max}(s)[$, for the same separation letter, had a common intersection. With one additional place needed to prevent a and b at the end of a word, we therefore believe:

Conjecture 1. Solutions for Binary Solvable Words need $\leq$3 Places. For any solvable word $w \in \{a, b\}^*$ there is a Petri net with at most three places solving it. □ Conjecture 1

The following algorithm for the Petri net synthesis for a finite word w is in $O(|w|^2)$:

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 $(P, \{a,b\}, F, M_0)$

Note that the first part (with the **for**-loops) is obviously quadratic, and the **while**-loop is run two times if Conj. 1 holds and at most $|w|$ times otherwise. For the choice of I, 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.

For an enumeration of all solvable words (ordered by length) without synthesising Petri nets, we would 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.

The algorithm ABSolve can be adapted for k-bounded Petri nets (where in every reachable marking every place has at most k tokens). Note that when choosing $\frac{m}{n}$, both $m \leq k$ and $n \leq k$ must hold, so the number of options does not depend on $|w|$. We need to check, though, if the created place could have more than k tokens on it (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. So, if we create one place for each interval we could do the second half of ABSolve in $O(k^2 \cdot |w|^2)$ (the first half remaining unchanged), but an optimal solution with as few places as possible is much harder to gain.

5 Cyclic Solvable Words

A word $w = t_1 \dots t_n$ (with $t_i \in T$) is *cyclic solvable* if the transition system $TS_{cyc}(w) = (\{0, \dots, n\}, \{(i-1, t_i, i) \mid 0 < i < n \wedge t_i \in T\} \cup \{(n, t_n, 0)\}, T, 0)$ is solvable. $TS_{cyc}(w)$ represents the infinite word w^ω. A Petri net solving $TS_{cyc}(w)$ reproduces its initial marking by firing w and allows for the (infinite) firing of w^ω.

Theorem 3. Characterisation of Cyclic Solvable Binary Words.
A word $w \in \{a, b\}^+$ is cyclic 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: "$\Rightarrow$": Let N be the Petri net solution for w. Due to the reproduction of the initial marking we can fire w arbitrarily often, i.e. 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 $\#_y(w) = \#_y(x\alpha) + \#_y(y\beta)$ and $\#_x(w) = \#_x(x\alpha) + \#_x(y\beta)$, the ratio of y to x ($\in \mathbb{Q} \cup \{\infty\}$) in w must lie between those of $x\alpha$ 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

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

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 add up to w. If we 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. Since s is chosen arbitrarily, the infinite word w^ω is solvable and thus w is cyclic solvable.

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 and y in w must be between those of $x\gamma$ 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 $x\alpha\tilde{w}$). Just note that $\hat{w}$ and w have the same Parikh vector, so the same number of x and y 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. This concludes the proof. □ 3

Note that 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 $\frac{\#_y(w)}{\#_x(w)}$ – independently of the separation point, as long as we prevent the same transition y. We conclude:

Proposition 8. Nets for Cyclic Solvable Words.
If $w \in \{a,b\}^+$ is cyclic 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.

Take the word $w = ababbab$ as an example. We check prefixes ending before some b first. The $\frac{a}{b}$-ratio must be better than in w, i.e. $>\frac{3}{4}$. This is true for a (∞), aba (2), $abab$ (1), and $ababba$ (1). Then, rotate the front a to the end ($babbaba$) and check again (now for the $\frac{b}{a}$-ratio $>\frac{4}{3}$, and prefixes ending before an a): b (∞), $babb$ (3), and $babbab$ (2). We continue until we end up rotating back

to w. Here, everything is ok and w is cyclic solvable. The Petri net solving it is depicted in Fig. 5. Note, however, that at this point of the development, we do not know about its initial marking; it is only the next proposition which allows us to compute it.

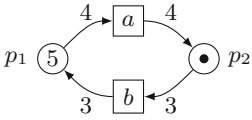

Fig. 5. A Petri net solving the word $(ababbab)^\omega$

Let us call a word $w \in \{a,b\}^+$ *minimal cyclic solvable* if it is cyclic solvable and there is no shorter word v, $|v| < |w|$, with $v^\omega = w^\omega$.

Proposition 9. Token Count for Cyclic Solvable Words.
Let $w \in \{a,b\}^+$ be minimal cyclic 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 8 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, $\forall M \in [M_0\rangle$: $M(p_1) + M(p_2) = M_0(p_1) + M_0(p_2)$. Let $w[i]$ be the ith letter of w and let M_i markings with $M_{i-1}[w[i]\rangle M_i$ for $1 \le i \le |w|$. Then, $M_0 = M_{|w|}$ and due to minimal cyclic solvability of w, $M_i \neq M_j$ for $0 \le i < j < |w|$ (otherwise, $v^\omega = w^\omega$ for some rotation v of $w[i+1]\dots w[j]$). 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. $M_0(p_1)+M_0(p_2) \ge |w|-1$. Assume, $n := M_0(p_1) + M_0(p_2) \ge |w|$. Then, markings $(\#_b(w) + k, \#_a(w) + \ell)$ with $k, \ell \ge 0$ must be unreachable as they allow firing of both transitions. The remaining possible markings $(0,n)$, …, $(\#_b(w) - 1, n - \#_b(w) + 1)$ and $(n,0)$, …, $(n - \#_a(w) + 1, \#_a(w) - 1)$ 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, $M_0(p_1) + M_0(p_2) = |w| - 1$. □ 9

Algorithm 2 for cyclic solving of a word w is obviously in $O(|w|^2)$.

Lemma 5. 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 cyclic solvable and v is a postfix of w^i for some $i \ge 1$.

Proof: "⇒": For arbitrary late parts of vw^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 $|v| \le |w^i|$), we find u, x, y with (w.l.o.g.) $v = xau$ and $w^i = ybu$,

Algorithm 2. ABCycSolve

Input: $w \in \{a,b\}^+$
Output: Petri net solving w^ω if it exists
  compute $W[0] \leftarrow \#_a(w)$, $W[1] \leftarrow \#_b(w)$
  $m_0 \leftarrow 0$, $m \leftarrow 0$ {tokens available for a}
  for $i = 0$ **to** $|w| - 1$ **do** {rotations of w}
    $v \leftarrow w[i] \dots w[|w| - 1]w[0] \dots w[i - 1]$
    if $v[0] = 'a'$ **then** $R \leftarrow 0$ **else** $R \leftarrow 1$ {fraction selector}
    $N[0] \leftarrow 0$, $N[1] \leftarrow 0$ {for counting a's and b's}
    for $j = 0$ **to** $|v| - 1$ **do** {prefixes of v}
      if $v[0] \neq v[j]$ **and** $N[R] * W[1 - R] \leq W[R] * N[1 - R]$
        then return {unsolvable}
      if $v[j] = 'a'$ **then** $N[0] \leftarrow N[0] + 1$ **else** $N[1] \leftarrow N[1] + 1$
    endfor
    if $v[0] = 'a'$ **then** $m \leftarrow m - W[1]$ **else** $m \leftarrow m + W[0]$ {fire}
    if $m < 0$ **then** $m_0 \leftarrow m_0 - m$, $m \leftarrow 0$
  endfor
  $F(p_1, a) \leftarrow \#_b(w)$, $F(a, p_2) \leftarrow \#_b(w)$, $F(p_2, b) \leftarrow \#_a(w)$, $F(b, p_1) \leftarrow \#_a(w)$
  return $(\{p_1, p_2\}, \{a, b\}, F, \{p_1 \to m_0, p_2 \to |w| - 1 - m_0\})$

and $M_0[xa\rangle M[uyb\rangle M$ for some marking M. Since 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. Thus, $M(p_1) + M(p_2) \geq F(b, p_1) + F(a, p_2) = \#_a(w) + \#_b(w) = |w|$, contradicting Proposition 9.

"$\Leftarrow$": 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 3, differing only at the initial marking. □ 5

So, words vw^ω with $\#_a(w) > 0 < \#_b(w)$ are solvable only if they can be rewritten as u^ω. Checking words in which w contains only one letter, we get:

Lemma 6. 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 trivial 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 p preventing a at s' exists with $s'(p) - s(p) = \mathbb{E}(a) < 0$. Thus, a cannot fire infinitely often at s''. □ 6

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

Theorem 4. Solvable Cyclic Binary Words with a Prefix.
A word vw^ω with $v, w \in \{a,b\}^*$ is solvable if and only if $w = \varepsilon$ or vw^ω can be rewritten as a cyclic solvable word u^ω or it has the form a^+b^ω or b^+a^ω.

6 Experimental Results

In an implementation of the general synthesis algorithm using the region-based approach, it is very likely that one of the freely available ILP solvers is employed. Since such solvers usually implement the simplex algorithm, our theoretical considerations of Sect. 2.3 are of limited practical value and need to be complemented by benchmarks. In particular, while the simplex algorithm can have exponential run times, it frequently allows a system of linear equations to be solved in linear time, which is much better than Khachiyan's worst case complexity.

To see how our algorithm fares compared to the region-based approach, we used the tools Synet and APT, 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), trying to synthesize the whole set of words. From the composite result we computed the average run time per word.

We made tests for words in $(a|b)^x$ with a fixed word length $x \in \{1, \ldots, 700\}$, i.e. 700 tests times 4000 words per tool, and again for words from $(ab|abb)^x$, where we expected a higher probability for solvable words. (Note that for $x = 1$ randomisation means we tested the words a and b each about 2000 times, but e.g. at $x = 50$ it is extremely unlikely that we tested the same word twice.) Fig. 6 (upper left) shows the results for $(a|b)^x$ 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 the upper right we normalised all curves by dividing all values of each curve by its value at $x = 350$. Due to 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 $k > 1$. The lower left part of the figure shows the results for $(ab|abb)^x$. The times are higher than for $(a|b)^x$, 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 $x = 40$ 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. The lower right part of the figure 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, at least in our version of Synet.

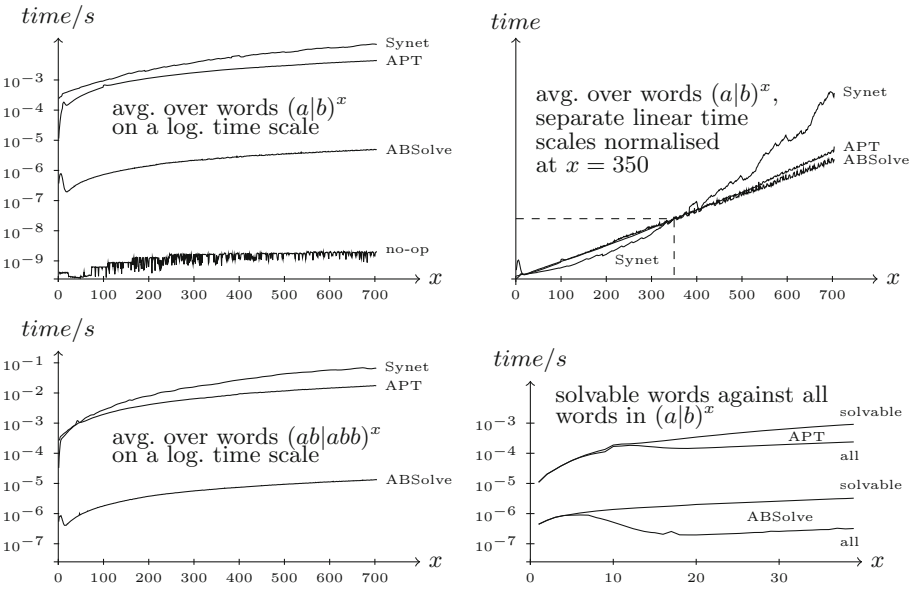

Fig. 6. Tests were done for random sets of 4000 words for each x with $1 \leq x \leq 700$ where words stem from $(ab|abb)^x$ (lower left) or $(a|b)^x$ (the other three pictures). Time scales are linear (but different for each curve) in the upper right picture and logarithmic in the others.

7 Concluding Remarks

In this paper, the class of Petri net synthesisable binary words has been studied in depth. We have presented a linear-time necessary condition for solvability representing an educated guess (Theorem 1), as well as quadratic time characterisations for finite binary words and for cyclic binary words (Theorems 2 and 3). The proof of Theorem 2 can easily be turned into a proof of a conjecture stated in [2], the main difference being that the latter is formulated for minimal unsolvable (rather than general) words. The algorithms derived from our quadratic-time characterisations allow to check solvability considerably more quickly than a general synthesis algorithm could. This has been confirmed both by the theoretical estimates contained in this paper and by experimental validation.

It would be interesting to consider extensions and ramifications. For example, we know of no results characterising PN-solvable acyclic labelled transition systems with few branching points, or with some other regular structure. The work described in [4] is an exception, a reason being that the cyclic structure of marked graph reachability graphs is particularly harmonious. Extending the results from binary words to words over a larger alphabet should also be worthwhile, and does not seem to be easy.

The present work could also be of interest in a wider context, as it might entail nontrivial necessary conditions for the solvability of an arbitrary labelled

transition system. If the latter is solvable, then finding a PN-unsolvable structure in it may have a strong impact on its structure or shape. Also, words are persistent in the sense of [8] and tractable by the method described in [3]. However, they form (in some sense) a worst case and still lead to many region inequalities. It could therefore be interesting to check more closely whether the work described here can be of any benefit in enhancing the method described in [3].

Acknowledgments. We would like to thank Raymond Devillers for his very helpful suggestions to improve this paper and Valentin Spreckels for his valuable support in computing the experimental data.

References

1. Badouel, É., Bernardinello, L., Darondeau, P.: Petri Net Synthesis, 339 p. Springer, Heidelberg (2015). ISBN 978-3-662-47966-7
2. Barylska, K., Best, E., Erofeev, E., Mikulski, Ł., Piątkowski, M.: On binary words being Petri net solvable. In: Carmona, J., Bergenthum, R., van der Aalst, W. (eds) Proceedings of the ATAED 2015, pp. 1–15 (2015). http://ceur-ws.org/Vol1371
3. Best, E., Devillers, R.: Synthesis of bounded choice-free Petri nets. In: Aceto, L., Frutos Escrig, D. (eds) Proceedings of the 26th International Conference on Concurrency Theory (CONCUR 2015), LIPICS, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, pp. 128-141 (2015). doi:10.4230/LIPIcs.CONCUR.2015.128
4. Best, E., Devillers, R.: Characterisation of the state spaces of live and bounded marked graph Petri nets. In: Dediu, A.-H., Martín-Vide, C., Sierra-Rodríguez, J.-L., Truthe, B. (eds.) LATA 2014. LNCS, vol. 8370, pp. 161–172. Springer, Heidelberg (2014)
5. Caillaud, B.: http://www.irisa.fr/s4/tools/synet/
6. Khachiyan, L.: Selected Works, Moscow Center for Mathematical Continuous Education, ISBN 978-5-94057-509-2, 519 pages (2009). (in Russian)
7. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
8. Landweber, L.H., Robertson, E.L.: Properties of conflict-free and persistent Petri nets. J. ACM **25**(3), 352–364 (1978)
9. Reisig, W.: Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies, 211 p. Springer, Heidelberg (2013). ISBN 978-3-642-33278-4
10. Schlachter, U. et al.: (2013). https://github.com/CvO-Theory/apt