Conditions for Petri Net Solvable Binary Words

Kamila Barylska[2], Eike Best[1(✉)], Evgeny Erofeev[1], Łukasz Mikulski[2], and Marcin Piątkowski[2]

[1] Parallel Systems, Department of Computing Science, Carl von Ossietzky Universität, 26111 Oldenburg, Germany
{eike.best,evgeny.erofeev}@informatik.uni-oldenburg.de
[2] Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, 87-100 Toruń, Poland
{kamila.barylska,lukasz.mikulski,marcin.piatkowski}@mat.umk.pl

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. Two conjectures providing different characterisations of this class of words are motivated and proposed. One conjecture characterises the class in terms of pattern-matching, the other in terms of letter-counting. Several results are described which amount to a partial proof of these conjectures.

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 [14]. 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 (see, e.g., [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.

In this paper, we investigate labelled transition systems which are finite and acyclic. The ultimate aim is to characterise, graph-theoretically, exactly which

K. Barylska, Ł. Mikulski and M. Piątkowski—Supported by the Polish Nat. Sci. Center (grant no. 2013/09/D/ST6/03928).
E. Best and E. Erofeev—Supported by DFG CAVER, ARS, and http://www.uni-oldenburg.de/en/scare/.

M. Koutny et al. (Eds.): ToPNoC XI, LNCS 9930, pp. 137–159, 2016.
DOI: 10.1007/978-3-662-53401-4_7

ones of them are synthesisable into an unlabelled place/transition Petri net [11]. To our knowledge, 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. That is, we study the class of binary words, and our aim is to characterise the Petri net synthesisable ones amongst them.

Region theory [1] provides an indirect characterisation of this class by means of an algorithm based on solving systems of linear inequations and synthesising a Petri net if possible. In this paper, we describe two alternative, more direct, characterisations, and provide partial proofs in support of their validity. The first condition characterises the class of Petri net synthesisable binary words in terms of a pseudo-regular expression. The second condition characterises the same class in terms of a letter-counting relationship. Both conditions seem to be more efficient to check than by using the general synthesis algorithm.

In Sect. 2 we briefly recapitulate some basic definitions about labelled transition systems, Petri nets, and regions. Sections 3 and 4 describe our two conjectures and contain proofs that they are necessary for synthesisability. In Sect. 5, we provide sufficiency proofs for special cases of these conjectures. Section 6 reduces the problem to words of a special form, and Sect. 7 describes some pertinent results about words of such forms. Section 8 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 $s[\sigma\rangle s'$. For clarity, in case of long formulas we write $|_r\, \alpha\, |_s\, \beta\, |_q$ instead of $r\ [\alpha\rangle\ s\ [\beta\rangle\ q$. 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, F is the flow function $F\colon ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}$ specifying the arc weights, and M_0 is the initial marking (where a marking is a mapping $M\colon P \rightarrow \mathbb{N}$, 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 ${}^\bullet p = \{t \in T \mid F(t,p)>0\}$. N is pure or side-place free if it has no side-places. 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 $TS(w)$.

2.2 Basic Region Theory, and an Example

Let a finite labelled transition system $TS = (S, \rightarrow, T, s_0)$ 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 $s \neq s'$, and for every such set, one needs a place that distinguishes them. Such problems are always solvable if $TS = TS(w)$ originates from a word w; for instance, we might simply introduce 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.

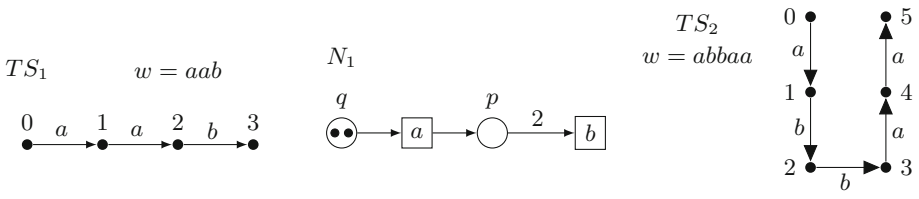

Fig. 1. TS_1 and TS_2 correspond to *aab* and *abbaa*, respectively. N_1 solves TS_1. No Petri net solution of TS_2 exists.

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

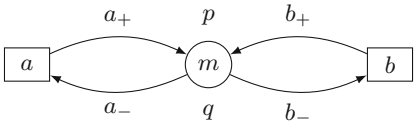

Fig. 2. A place with four arc weights a_-, a_+, b_-, b_+ and initial marking m. It is named p if used for preventing b and named q if used for preventing a.

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. 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 2. 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.

2.3 Brief Estimation of the Complexity of the General Algorithm

In a word of length n, the equation system for a single event/state separation problem comprises $n + 1$ inequations, n for the states $0, \ldots, n - 1$, which guarantee that the corresponding transition is enabled, and one for the event/state separation itself. 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 Karmarkar's algorithm [10]. It seems that, solving $O(n)$ systems of inequalitites, we may roughly expect a running time of $O(n^3 \cdot L(n))$, i.e., cubic with a logarithmic factor $L(n) = \log(n) \cdot \log(\log(n))$.

For the remainder of the paper, we fix $T = \{a, b\}$.

3 A Pattern-Matching Condition

3.1 Minimal Unsolvable Words

If w is PN-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. For this reason, the notion of a *minimal unsolvable word* is well-defined, namely, as an unsolvable word all of whose proper subwords are solvable. A complete list of minimal unsolvable words up to length 110 can be found, amongst some other lists, in [13]. Observe that in this list, every word starts and ends with the same letter. This is a consequence of (the contraposition of) the next proposition.

Proposition 1. 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 $w = b^k$ (for $k \in \mathbb{N}$) then $awb = ab^{k+1}$ is obviously solvable, hence we assume that w contains at least one a. 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 gives a net which is synchronised at a and b, and which allows all (and only) sequences allowed by both N_1 and N_2. We modify N_1 and N_2 before forming their union, as follows:

(i) In N_1, for each place p in ${}^\bullet b \cap P_1$, add another $F_1(p, b)$ tokens; and if p in ${}^\bullet a \cap P_1$, then add the quantity $F_1(p, b)$ both to $F_1(p, a)$ and to $F_1(a, p)$; otherwise, keep the arc weights unchanged. This allows an additional b in the end of the word awb. Since the last b in awb could have enabled a at the final state, we add a counting place q_a which is an input place for a with a unit arc weight and has $\#_a(aw)$ tokens on it initially. Thus, a remains disabled in awb exactly at states in which it was disabled before the modification and becomes permanently disabled after aw.

(ii) Modify N_2 by adding to each place q in ${}^\bullet a \cap P_2$ another $F_2(q, a)$ tokens (this allows an additional a). Further, for each place p in $a^\bullet \cap P_2 \cap {}^\bullet b$, add the quantity $F_2(a, p)$ both to $F_2(p, b)$ and to $F_2(b, p)$. 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 in the previous step). 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. To prevent this, add a new place p' to 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 disables b at the beginning of awb and does not influence the behaviour of N_2 after the first a.

Define N as the union of the two nets thus modified, and see Fig. 3 for an example. (The added tokens are drawn as hollow circles.) In general, N solves

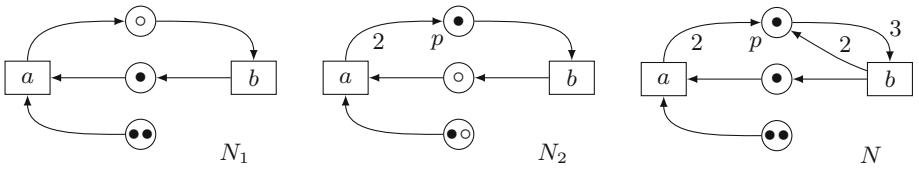

Fig. 3. N_1 (black tokens) solves $aw{=}abab$. N_2 (black tokens) solves $wb{=}babb$. N (redundant places omitted) solves $awb{=}ababb$. Arc weight change due to $p{\in}a^\bullet{\cap}P_2{\cap}{}^\bullet b$.

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 final additional a is allowed because N_1 does not allow it. All intermediate occurrences of a are regulated by the modification of N_1, and the same of b by the modification of N_2. □

This proposition can be used for a remark on word reversal. If both aw, wb and their reversals are solvable, then both awb and its reversal are solvable. This follows directly from the previous proof. If one of the reversals of aw and wb is not solvable, however, then the reversal of awb is not necessarily solvable. Consider, for instance, $w = abba$, $aw = aabba$, $wb = abbab$, and $awb = aabbab$. Here, aw is solvable, but its reversal is $abbaa$, which is a subword of the reversal of awb.

3.2 A Pseudo-regular Expression for Unsolvable Words

Studying the list [13], it can first be observed that all words starting and ending with b are just mirror 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 be of the following general form:

$$\boxed{(\ a\ b\ \alpha\ )\ b^*\ (\ b\ a\ \alpha\ )^+\ a\ , \qquad \text{with } \alpha \in T^*} \tag{1}$$

with a not being separated 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 (1) with $\alpha = \varepsilon$, the star * being repeated zero times, and the plus $^+$ being repeated once. Such words are generally PN-unsolvable:

Proposition 2. Sufficient condition for the unsolvability of a word
If a word over $\{a, b\}$ has a subword of the form (1), then it is not PN-solvable.

Proof: Let s_0 be the state before the first a in (1), 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 as in Fig. 2 exists which separates a at s. Abbreviate $\mathbb{E}(ab\alpha)$ to E and $\mathbb{E}(b)$ to E_b. For q, we have the following inequalities:

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

(0) is true because s_0 enables a. (s') is true because s' enables a. (r) is true because r enables a; and $\ell > 0$ because of the $^+$. Finally, (s) is true because q disables a at state s. Adding (s')+(s) gives $1 \leq E_b$. Adding (0)+(s) gives $1 \leq -E - k{\cdot}E_b$, and using also $1 \leq E_b$ 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 separating a at s exists. □

3.3 Converses of Proposition 2, and Complexity Estimation

All words of the form (1) are unsolvable, but there exist unsolvable words which are not of this form. Nevertheless, it turns out that all *minimal* unsolvable words not only conform to (1), but are of an even simpler shape, as expressed in the following conjectures (and as will be elaborated in later parts of this paper; the facts from Sects. 5 and 6 are the justification of their partial correctness).

Conjecture 1. First converse of Proposition 2
Suppose a word over $\{a, b\}$ is non-PN-solvable and minimal with that property. Then it is (modulo swapping a and b) of the form given in (1). □

Basing on computer experiments partially supported by Proposition 5 (the fact on the existence of subwords aa and bb inside solvable words proven in Subsect. 6.2) we also feel that, without loss of generality, one can restrict (1) to α containing only letters b if the b^* part is not empty. More precisely:

Conjecture 1a. Strengthened converse of Proposition 2
Each minimal unsolvable word over $\{a, b\}$ conforms to one of the forms

$$\boxed{\left[\, ab\underbrace{b^j}_{\alpha} b^k ba \underbrace{b^j}_{\alpha} a \text{ with } j \geq 0, k \geq 1 \,\right] \quad \text{or} \quad \left[\, ab\alpha(ba\alpha)^\ell a \text{ with } \ell \geq 1 \,\right]} \qquad (2)$$

(again, modulo swapping a and b). □

Using Conjecture 1, the problem of deciding the PN-solvability of a word v of length n can be reduced to a pattern-matching problem. Namely, we need to verify whether v contains a subword w of the form (1). Using an algorithm

based on the Knuth-Morris-Pratt algorithm [8] (utilizing strict border arrays to search for the repetitions by processing all suffixes of v), this can be done in time $O(n^2 \log n)$. Using Conjecture 1a, subwords of the form $ab\alpha(ba\alpha)^\ell a$ with $\ell \geq 1$ can be recognised using the same technique (KMP-like algorithm). Let us notice that in this case the partial matched subword u_1 and the repeating subword $(u_2)^\ell$ are not separated by a block of the form b^k. Subwords of the form $abb^j b^k bab^j a$ $(j \geq 0, k \geq 1)$ can be recognised in time $O(n)$ by counting distances between consecutive occurrences of a (at any moment we have to remember only the positions of two preceding occurrences of a). In contrast to the general case, using Conjecture 1a do not need any additional preprocessing or memory, and the solution takes time at most $O(n^2)$.

4 A Counting Condition

4.1 An Arithmetic Criterion for Unsolvable Words

Proposition 3. 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

$$\boxed{\#_a(\beta)\cdot\#_b(\alpha) \;\geq\; \#_a(\alpha)\cdot\#_b(\beta)} \tag{3}$$

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:

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

If a place q separates a at s and has marking m at s_0, then for $E_\alpha = \mathbb{E}(\alpha) = \#_a(\alpha)\cdot E_a + \#_b(\alpha)\cdot E_b$ and $E_\beta = \mathbb{E}(\beta) = \#_a(\beta)\cdot E_a + \#_b(\beta)\cdot E_b$ we have:

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

Adding (0)+(s) yields $1 \leq -E_\alpha$, hence (A): $-(\#_a(\alpha)E_a + \#_b(\alpha)E_b) \geq 1$.
Adding (r)+(s) yields $1 \leq E_\beta$, hence (B): $(\#_a(\beta)E_a + \#_b(\beta)E_b) \geq 1$.
Also, $E_b \geq 1$ because q prevents a at s, but a becomes enabled after one or more firings of b. Then,

$$\begin{array}{rll} -\#_a(\beta) \;\geq\; \#_a(\beta)\#_a(\alpha)E_a + \#_a(\beta)\#_b(\alpha)E_b & & (\text{multiplying (A) by } \#_a(\beta)) \\ \geq\; \#_a(\beta)\#_a(\alpha)E_a + \#_a(\alpha)\#_b(\beta)E_b & & (\text{using (3) and } E_b \geq 1) \\ \geq\; \#_a(\alpha) & & (\text{multiplying (B) by } \#_a(\alpha)) \end{array}$$

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. □

4.2 Converses of Proposition 3, and complexity estimation

Conjecture 2. First converse of Proposition 3
If a word is of the form $w = \alpha\beta a$ where α starts with a and β starts with b, and if w is minimal non-PN-solvable, then inequation (3) holds. □

We also believe that for sufficiently long words that are of a special (as it turns out, interesting) shape, the $\geq$ in (3) can be strengthened to equality. More precisely,

Conjecture 2a. Strengthened converse of Proposition 3
If $w = \alpha\beta a$ is of the form

$$w \;=\; \underbrace{ab^{x_1}a\ldots ab^{x_k-1}}_{\alpha} \;|_s\; \underbrace{ba\ldots ab^{x_n}}_{\beta}\, a \quad \text{with } n \geq 3 \text{ and } x_i \geq 1 \tag{4}$$

then a is not separable at state s iff $\#_a(\beta)\cdot\#_b(\alpha) = \#_a(\alpha)\cdot\#_b(\beta)$. □

The arithmetic criterion in Conjecture 2a tells us nothing about minimality. The word to be checked is assumed to start and end with the same letter. In a bad case, e.g., for $w = ab\ldots ab$, checking needs to be repeated two times. Each maximal subword, starting and ending with the same letter, can be divided into α and β at most in $n-3$ different ways. For every such devision we need to go through the subword of length $n-2$ once, in order to check the criterion. This amounts to time approximately $2(n-3)(n-2)$ and thus, time $O(n^2)$ in total. A solvability algorithm based on Conjecture 2 was recently (end of October 2015) implemented by Harro Wimmel and compared with the general synthesis APT algorithm [6,15]. It was briefly tested on 1024 words of length 1990. The special algorithm took about a minute to check solvability, while the general algorithm takes much longer (being general-purpose and actually constructing a solution if one exists). To our knowledge, testing only solvability with the general algorithm, without necessarily finding a solution, is only faster in the degree of the number of variables, which is constant for the separation problems.

A reasonable and possibly beneficial approach could be to use the algorithms described in Sects. 2.3, 3.3 and 4.2 in combination, depending on a particular task: The general algorithm yielding a Petri net solution if the given word is solvable; the pattern-matching algorithm checking minimal unsolvability (and possibly combinable with other efficient methods); and the counting algorithm checking solvability or unsolvability (but requiring, for minimality, several repetitions for subwords).

5 Special Cases of the Two Conjectures

In this section, we substantiate Conjectures 1 and 2 by providing partial proofs for the converses of Propositions 2 and 3. First, we prove the minimal unsolvability of words corresponding to the following two patterns, as special instances of (1):

$$abb^xb^kbab^xa \quad \text{and} \quad abb^xb(ab^xb)^dab^xa \quad \text{with } x \geq 0, k \geq 1, d \geq 0 \tag{5}$$

The first pattern satisfies (1) with $\alpha = b^x$, the star * being repeated k times, and the plus $^+$ being repeated only once, while the second pattern satisfies (1) with $\alpha = b^x$, the star * being repeated zero times, and the plus $^+$ being repeated $d + 1$ times. Due to Proposition 2, all binary words of one of the forms in (5) are unsolvable. To prove that they are minimal with this property, we provide Petri nets (with initial markings) solving maximal proper prefixes and maximal proper suffixes of these words.

The Petri net N_1 on the left-hand side of Fig. 4, with appropriate values of parameters in the arc weights and initial marking, is a possible solution for a maximal prefix $abb^x b^k bab^x$ of the first form in (5). Place p_1 prevents b at the beginning, and p_2 restricts the total number of b's. Place q prevents a when it is necessary. This place has enough tokens on it for the initial a and for one more a after the block $bb^x b^k b$, and it does not enable a afterwards.

The maximal proper suffix $bb^x b^k bab^x a$ can be executed by the net N_2 on the right-hand side of Fig. 4. Initially only $x + k + 2$ firings of b are possible, which brings enough tokens on place q for a to occur. This first a adds x tokens on place p_1, which enables b again. The total number of b's is controlled by place p_2. When there is no tokens on p_2, a is enabled once more, and this last occurrence of a ends the execution of the suffix. Hence, words of the first form in (5) are minimally unsolvable.

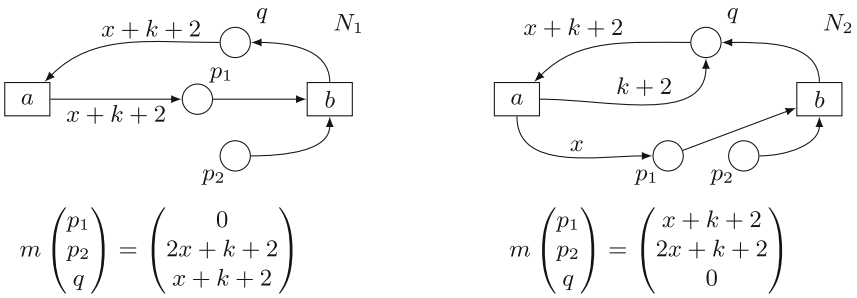

Fig. 4. N_1 solves the prefix $abb^x b^k bab^x$. N_2 solves the suffix $bb^x b^k bab^x a$.

The maximal proper prefix $abb^x b(ab^x b)^d ab^x$ of the second form in (5) can be solved by the net N_1 in Fig. 5. Place q in this net enables the initial a, and then disables it unless b has been fired $x + 2$ times. After the execution of block $bb^x b$ there are d tokens more than a needs to fire on place q. These surplus tokens allow a to be fired after each sequence $b^x b$, but not earlier. Place p_1 has initially 1 token on it, which is necessary for block $bb^x b$ after the first a, and this place has only $x + 1$ tokens after each next a, preventing b at states where a must occur. Places p_2 and p_3 prevent undesirable occurrences of b at the very beginning and at the very end of the prefix, respectively.

For the general form of suffix $bb^x b(ab^x b)^d ab^x a$ of the second form in (5), one can consider the Petri net N_2 on the right-hand side of Fig. 5 as a possible

solution. Indeed, place q_1 prevents premature occurrences of a in the first block bb^xb, and enables a only after this and each next block b^xb. Doing so, it collects one additional token after each b^xb, which allows this place to enable the very last a after sequence b^x. The initial marking allows to execute the sequence bb^xb in the beginning, and at most $x+1$ b's in a row after that, thanks to place p_1. Place p_2 restricts the total number of b's allowing only block b^x at the end. Place q_2 serves for bounding the total number of occurrences of a, and it is necessary if $x = 0$ and $d = 0$. Thus we deduce that any word of the form $abb^xb(ab^xb)^dab^xa$ with $x, d \geq 0$ is minimally unsolvable.

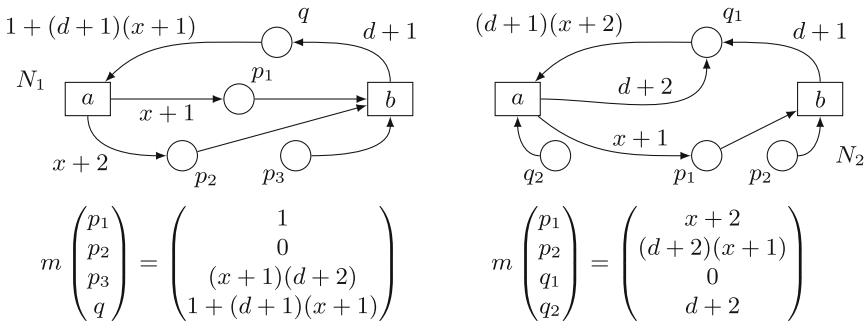

Fig. 5. N_1 solves the prefix $abb^xb(ab^xb)^dab^x$. N_2 solves the suffix $bb^xb(ab^xb)^dab^xa$.

Words of two forms in (5) correspond to two classes of minimally unsolvable words that were described in Conjecture 1a, the strengthened variant of Conjecture 1. Moreover, while the form $abb^xb(ab^xb)^dab^xa$ is only a partial instance (for $\alpha = b^x$) of the more general form $ab\alpha(ba\alpha)^\ell a$ with $\ell \geq 1$ (see Conjecture 1a), pattern $abb^xb^kbab^xa$ coincides entirely with $abb^jb^{kba}b^ja$, where $j \geq 0, k \geq 1$ (cf. (2)).

In support of Conjectures 2 and 2a, assume a minimally unsolvable word

$$w_1 \;=\; abb^x \,|_{s_1}\, b \,\ldots\, b^x \,|_{s_j}\, b \,\ldots\, b^x \,|_{s_{d+1}}\, bab^xa$$

of the second form in (5) to be given, with some fixed non-negative x and d. For any $1 \leq j \leq d+1$ and state s_j, in $w_1 = \underbrace{ab\ldots b^x}_{\alpha}|_{s_j}\underbrace{b\ldots ab^x}_{\beta}a$ we have

$$\#_a(\beta)\cdot\#_b(\alpha) = (d+2-j)\cdot((x+1)\cdot j) = j\cdot((d+1-j)\cdot(x+1)+1+x) = \#_a(\alpha)\cdot\#_b(\beta)$$

By Proposition 3, a is not separated at such states s_j. On the other hand, expression (3) is fulfilled in w_1 as an equality, which corresponds to the strong variant of Conjecture 2.

The requirement $n \geq 3$ in (4) is important. In a minimally unsolvable word $w_2 = \underbrace{abb^xb^k}_{\alpha}|_r\underbrace{bab^x}_{\beta}a$ of the first form in (5), with $x \geq 0$ and $k \geq 1$, we have

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

According to Proposition 3, a is not separated at r, but (3) is satisfied as a strict inequality.

6 Limiting the Occurrence of Factors *aa* or *bb*

In this section, we show that the problem of characterising minimal unsolvable words w can be reduced to two cases, $w = b^{x_1}a \dots ab^{x_n}$ or $w = ab^{x_1}a \dots ab^{x_n}a$ (both with $x_1 \geq 1$). Observe that Conjecture 2a concerns the second case.

Since words in which a and b strictly alternate are easy to solve, it stands to reason to investigate the situations in which a letter occurs twice in a row. We show that in a minimal unsolvable word, the factors aa and bb are essentially limited to occur in some particular ways.

6.1 Factors *aa* or *bb* Starting an Unsolvable Word

If a word av is unsolvable and if av is minimal unsolvable, then, as a consequence of the next proposition, v definitely starts with a letter b. That is, no minimal unsolvable word can start with aa (nor with bb, for that matter).

Proposition 4. 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 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 (cf. Fig. 6).

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 a after the last occurrence of a. 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 ℓ is the number of a before the first b in av, and $M_0(q_a) = k$, $F(q_a, a) = 1$, where k is the number of a in av. (All other F values $= 0$.)

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 M_0. Note that neither p_b nor q_a are in $NUF(a)$. Note also that for every $p \in NUF(a)$, $F(p, a) \leq M_0(p) < F(a, p)$ – the first because a is initially enabled, the second by $p \in NUF(a)$. That is, a has a positive effect on p. Without loss of generality, b has a negative effect on p (otherwise, thanks to the normalising place p_b, p could be deleted without changing the behaviour of N).

For every $p \in NUF(a)$ we add the quantity $F(a, p)$ uniformly to $M_0(p)$, to $F(p, b)$, and to $F(b, p)$, eventually obtaining $N' = (P', \{a, b\}, F', M'_0)$, and we show that N' also solves av. 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 by definition, the latter by 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 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,

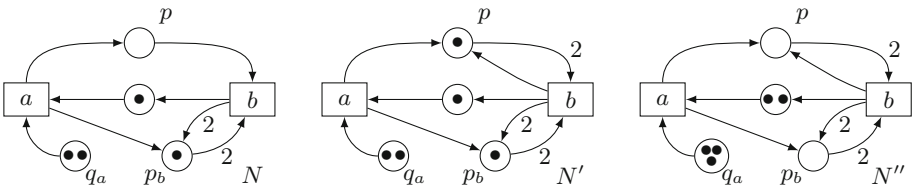

Fig. 6. N is normalised and solves $abab$. N' solves $abab$ as well. N'' solves $aabab$.

$q \notin NUF(a)$, 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. By construction, aav is a firing sequence of N''. Furthermore, M_0'' does not enable b because of p_b. □

6.2 Factors *aa* or *bb* Inside a Minimal Unsolvable Word

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 5. No aa and bb inside a minimal unsolvable word
If a minimal non-PN-solvable 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 group of a's which goes after a group of b's. Let a^m and b^n be such groups, assume that a^m goes after b^n and that there are no groups of a or of b 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, $(ab)b^{n-2}(ba)^{k+1}a$ is a proper subword of u. But it has the form $(abw)b^*(baw)^+a$, with $w = \varepsilon$, which implies its unsolvability by Proposition 2, 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 4. 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

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

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 $w = \varepsilon$, implying its non-PN-solvability, due to Proposition 2 with inverted a and b. This again contradicts the minimality of u. □

For these reasons, we are particularly interested in words of the following form:

$$\boxed{\text{either}\quad ab^{x_1}a \ldots ab^{x_n}a \quad\text{or}\quad b^{x_1}a \ldots ab^{x_n} \quad\text{where } x_i \geq 1 \text{ and } n > 1} \tag{6}$$

In the first form, there are no factors aa. If factors bb are excluded and the word starts and ends with an a, then we get words that are of the second form, except for swapping a and b.

7 Some Results About Words of the Form $b^{x_1}a \ldots ab^{x_n}$

Let $w = b^{x_1}a \ldots ab^{x_n}$ be a word with $n > 1$ and $x_i \geq 1$ for every $1 \leq i \leq n$, consisting of groups of letters b separated by single a's, and starting and ending with b. With a view to (6), it seems important to understand conditions

- for transforming solutions of w into solutions of aw,
- and for transforming solutions of w (or aw) into solutions of wa (or awa).

In the present section, we address the first of these tasks. The aim is to modify an existing solution of w to yield a solution of aw. Similar constructions in the previous sections were typically done by transforming the places of an existing Petri net into places of a new net. The proof technique employed in this section allows to create new regions from old ones by transforming a given solution involving quantities such as m, a_-, etc., into new quantities such as m', a'_-, etc. This is useful as there is not always a direct intuitive (pictorial) relationship between the new and the old places.

7.1 Side-Places in Words of the Form $b^{x_1}a \ldots ab^{x_n}$

If a word $w = b^{x_1}a \ldots ab^{x_n}$ can be solved, then side-places may be necessary to do it. For instance, *bbabbababab* cannot be solved side-place-freely. (More precisely: a side-place is needed in order to separate a at state 6.) 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 $x_1 \leq \min\{x_2, \ldots, x_{n-1}\}$. For example, *babbababab* can be solved without any side-places. The "smallness" of x_1 is sufficient but not necessary. For instance, *bbabbabab* has a side-place-free solution, even though $x_1 \not\leq \min\{x_2, \ldots, x_{n-1}\}$.

In the following, we assume w to be of the following form (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.

$$w = b^{x_1-1}\,|_{r_1}\,b\,|_{s_1}\,a\,b^{x_2-1}\,|_{r_2}\,b\,|_{s_2}\,a\;\ldots\;|_{s_{k-1}}\,a\,b^{x_k-1}\,|_{r_k}\,b\,|_{s_k}\,a\;\ldots\;|_{s_{n-1}}\,a\,b^{x_n} \tag{7}$$

Proposition 6. Side-place-free solvability with few initial b's
If $w = b^{x_1}ab^{x_2}a\ldots ab^{x_n}$ 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 $w = b^{x_1}ab^{x_2}a\ldots ab^{x_n}$ is solvable and $x_1 \leq \min\{x_2, \ldots, x_{n-1}\}$, then w is solvable side-place-freely.

Proof: The first claim follows from Lemmata 1 and 2 below. The second claim follows from Lemma 3. □

Lemma 1. side-place-freeness around b
If $w = b^{x_1}a\ldots ab^{x_n}$ 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 (7) of w, we have

$$\begin{array}{rcl} b_+ & \leq & m + x_1(b_+ - b_-) \\ b_+ & \leq & m + (x_1 + x_2)(b_+ - b_-) + (a_+ - a_-) \\ \cdots & & \\ b_+ & \leq & m + (x_1 + \ldots + x_n)(b_+ - b_-) + (n-1)(a_+ - a_-) \\ 0 & \leq & -m - (x_1 + \ldots + x_k)(b_+ - b_-) - (k-1)(a_+ - a_-) + b_- - 1 \end{array} \tag{8}$$

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 (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 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. n'th) group of b's, a can easily be prevented side-place-freely. For place q with initial marking m, we have

$$\begin{array}{rcl} a_+ & \leq & m + x_1(b_+ - b_-) + (a_+ - a_-) \\ a_+ & \leq & m + (x_1 + x_2)(b_+ - b_-) + 2(a_+ - a_-) \\ \cdots & & \\ a_+ & \leq & m + (x_1 + \ldots + x_{n-1})(b_+ - b_-) + (n-1)(a_+ - a_-) \\ 0 & \leq & -m - (x_1 + \ldots + x_k - 1)(b_+ - b_-) - (k-1)(a_+ - a_-) + a_- - 1 \end{array} \tag{9}$$

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 (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 (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 2. SIDE-PLACE-FREENESS AROUND a, PREVENTING b
Suppose $w = b^{x_1}ab^{x_2}a \dots ab^{x_n}$. 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 (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. □

If some place q prevents transition a, then it may be a side-place between q and a. It may not always be possible to remove such a side-place. For instance, the word $w = bbabbababab$ is of the form (7), and any net solving it necessarily contains a side-place around transition a. The next lemma shows that the presence of a side-place around a may be due to there being "many" initial b's.

Lemma 3. SIDE-PLACE-FREENESS AROUND a, PREVENTING a
Suppose $w = b^{x_1}ab^{x_2}a \dots ab^{x_n}$. If $x_1 \leq \min\{x_2, \dots, x_{n-1}\}$ 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 1, 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 (9), except that it is simplified by $b_- = 0$):

$$\begin{array}{rl} a_+ & \leq m + x_1(b_+) + (a_+ - a_-) \\ a_+ & \leq m + (x_1 + x_2)(b_+) + 2(a_+ - a_-) \\ \dots & \\ a_+ & \leq m + (x_1 + \dots + x_{n-1})(b_+) + (n-1)(a_+ - a_-) \\ 0 & \leq -m - (x_1 + \dots + x_k - 1)(b_+) - (k-1)(a_+ - a_-) + a_- - 1 \end{array} \tag{10}$$

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, (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 (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 (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:

$$\widetilde{m}(q) \quad \geq \quad \widetilde{m}(q') \quad \geq \quad 0 \tag{11}$$

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 (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 (11), let $x = \min\{x_2, \ldots, x_{n-1}\}$. Then

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

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. □

7.2 Solving Words aw from Words of the Form $w = b^{x_1}a \ldots ab^{x_n}$

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 7. SIDE-PLACE-FREE SOLVABILITY OF $b^{x_1}ab^{x_2}a \ldots ab^{x_n}$
$w = b^{x_1}ab^{x_2}a \ldots ab^{x_n}$ is solvable side-place-freely iff aw is solvable.

Proof: Lemmata 4 and 5 for ($\Rightarrow$), and Lemma 6 for ($\Leftarrow$). □

Lemma 4. 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 (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 k'th group of b's) and a correspondingly modified system as follows:

$$\boxed{\begin{array}{rcl} 0 & \leq & m' + (x_1 + \ldots + x_i) \cdot (b'_+) + (i+1) \cdot (-a'_-) \quad \text{for all } 0 \leq i \leq n-1 \\ 0 & \leq & -m' - (x_1 + \ldots + x_k - 1) \cdot (b'_+) - k \cdot (-a'_-) + a'_- - 1 \end{array}} \tag{12}$$

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 (12), provided m, b_+ and a_- (together with $a_+ = 0$ and $b_- = 0$) satisfy (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 (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 (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 (9)). □

Note 1: In order to disable a at r_k, q could be replaced by a place q' obtained by duplicating q and changing the initial marking m to $m' = m + a_-$. Intuitively, this means that m' is computed from m by "unfiring" a once.
Note 2: Place q should not be removed as soon as q' is added, because q could also be preventing a at some other r_k. In that case, a new place q'' must be computed from q for this different value of k. We may forget about q only after all the relevant indices k have been processed.

Lemma 4 does not, by itself, imply that aw is solvable. We still need to consider the separations of b. Thus, consider an input place p of b in a side-place-free solution of w and suppose that p prevents b at state s_k. Suppose that we want to solve aw. If p is not also an output place of a, then it can simply be retained unchanged, and with the same marking, prevent b at corresponding states in aw and in w. However, if p is also an output place of a, "unfiring" a in the initial marking may lead to negative tokens on p. This is illustrated by the word *babbabb* which has a side-place-free solution, as shown on the left-hand side of Fig. 7.

The places q_1, q_2 can be treated as in the above proof, that is, by changing their markings by "unfiring" a, yielding new places q'_1, q'_2 with marking $\{(q'_1, 3), (q'_2, 3)\}$. If we allowed negative markings, then a new place p' with initial marking $(p', -1)$ (and otherwise duplicating p) would do the job of solving *ababbabb* (as in the middle of the figure). However, we shall need a more delicate argument in order to avoid negative markings.

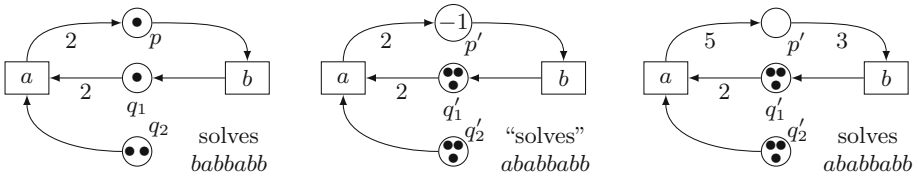

Fig. 7. Solving *babbabb* (l.h.s.), (almost) *ababbabb* (middle), and *ababbabb* (r.h.s.).

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 inequation system:

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

This system has to be compared with a restricted form of (8) (setting $b_+ = a_- = 0$, since the solution of w is pure). Doing this by line-wise comparison, we get the following inequation system for the new value differences:

$$\boxed{\begin{array}{lcl} \mu & \geq & -m, \ \beta_+ \geq -b_+, \ \beta_- \geq -b_-, \ \alpha_+ \geq -a_+, \ \alpha_- \geq -a_- \\ \beta_+ & \leq & \mu + (x_1 + \ldots + x_i) \cdot (\beta_+ - \beta_-) + i \cdot (\alpha_+ - \alpha_-) + a_+ \\ 0 & \leq & -\mu - (x_1 + \ldots + x_k) \cdot (\beta_+ - \beta_-) - k \cdot (\alpha_+ - \alpha_-) - a_+ + \beta_- \end{array}} \qquad (13)$$

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 5. SOLVING aw FROM w
Suppose $w = b^{x_1} a b^{x_2} a \ldots a b^{x_n}$ 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 (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 (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, (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. □

For instance, in the solution of *babbabb* shown on the left-hand side of Fig. 7, the place p from a to b satisfies m=1, b_-=1, b_+=0, a_-=0 and a_+=2. (13) is solved by $\mu = -1$, $\beta_- = 2$, $\beta_+ = 0$, $\alpha_- = 0$ and $\alpha_+ = 3$. Hence with m'=$m - 1$, b'_-=$b_- + 2$, b'_+=b_+, a'_-=a_- and a'_+=$a_+ + 3$, the net shown on the right-hand side of Fig. 7 is a pure solution of *ababbabb*. (Place p' prevents b not only in states s_1 and s_2 but also in the initial state and in the final state.) There exist words such as *bbabbababab*, however, which can be solved but for which aw is not solvable. We have a converse of Lemma 5:

Lemma 6. 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. □

8 Concluding Remarks

In this paper, the class of Petri net synthesisable binary words has been studied in depth. We have motivated, presented, and substantiated two conditions stating how such words could be characterised and how different algorithms could be devised for them. These algorithms can 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.

Several other facts are known about the class of two-letter PN-synthesisable words. It is easily seen that if a word is solvable side-place-freely, then so is the reverse word. Also, if a binary word is solvable, then it is solvable using places having exactly one outgoing transition. (This property is not shared by words with three or more letters, a counterexample being *abcbaa*.)

Moreover, PN-solvable words are balanced in the following sense. Referring to $w = b^{x_1}ab^{x_2}a \ldots ab^{x_n}$, call w balanced if there is some x such that $x_i \in \{x, x+1\}$ for all $2 \leq i \leq n-1$. We can prove that if $w = b^{x_1}ab^{x_2}a \ldots ab^{x_n}$ is PN-solvable, then w is balanced, and moreover, $x_n \leq x+1$. Presenting these, and other, properties of PN-solvability must however be left to future publications.

Once the conjectures are (hopefully) proved, it would be interesting to consider extensions and ramifications. For example, we know of no results characterising PN-solvable simple cycles, or 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.

The present work could well 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 [12] 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, Thomas Hujsa, Uli Schlachter and Harro Wimmel for valuable comments. We also thank the anonymous reviewers for their remarks which allowed to improve the presentation of the paper.

Note added in proof. This paper extends [2] by Sect. 5 and a few other enhancements. At the time of revision (May 2016), the conjectures stated in Sects. 3.3 and 4.2 have been proved correct. These proofs are contained in [5,9].

References

1. Badouel, É., 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/Vol-1371
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 2015 (2015). doi:10.4230/LIPIcs.CONCUR.2015.128, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, 128–141
4. Best, E., Devillers, R.: Characterisation of the state spaces of marked graph Petri nets. In: Dediu, A.H., et al. (eds) Selected Extended Papers of LATA 2014, To appear in Information and Computation, 20 p. (2015)
5. Best, E., Erofeev, E., Schlachter, U., Wimmel, H.: Characterising Petri net solvable binary words. In: Kordon, F., Moldt, D. (eds.) PETRI NETS 2016. LNCS, vol. 9698, pp. 39–58. Springer, Heidelberg (2016). doi:10.1007/978-3-319-39086-4_4

6. Best, E., Schlachter, U.: Analysis of Petri nets and transition systems. In: Knight, S., Lanese, I., Lafuente, A.L., Vieira, H.T. (eds.) Proceedings of the 8th Interaction and Concurrency Experience, Electronic Proceedings in Theoretical Computer Science, vol. 189, pp. 53–67, June 2015. http://eptcs.web.cse.unsw.edu.au/paper.cgi?ICE2015.6
7. Commoner, F., Holt, A.W., Even, S., Pnueli, A.: Marked directed graphs. J. Comput. Syst. Sci. **5**(5), 511–523 (1971)
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. The MIT Press, Cambridge (2009)
9. Erofeev, E., Barylska, K., Mikulski, Ł., Piątkowski, M.: Generating all minimal Petri net unsolvable binary words (2016). http://folco.mat.umk.pl/papers/generating-binary-muws.pdf
10. Karmarkar, N.: https://en.wikipedia.org/wiki/Karmarkar's_algorithm
11. Murata, T.: Petri Nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
12. Landweber, L.H., Robertson, E.L.: Properties of conflict-free and persistent Petri nets. JACM **25**(3), 352–364 (1978)
13. Piątkowski, M., et al.: http://folco.mat.umk.pl/unsolvable-words (2015)
14. Reisig, W.: Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies, 211 p. Springer, Hiedelberg (2013). ISBN 978-3-642-33278-4
15. Schlachter, U., et al.: https://github.com/CvO-Theory/apt (2013)