Information Processing Letters 38 (1991) 71–76
North-Holland

26 April 1991

A polynomial time algorithm to decide pairwise concurrency of transitions for 1-bounded conflict-free Petri nets

Hsu-Chun Yen

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC

Communicated by T. Lengauer
Received 7 November 1990
Revised 24 January 1991

Abstract

Yen, H.-C., A polynomial time algorithm to decide pairwise concurrency of transitions for 1-bounded conflict-free Petri nets, Information Processing Letters 38 (1991) 71–76.

We show that the problem of determining whether two transitions in a 1-bounded conflict-free Petri net can become enabled simultaneously is solvable in polynomial time, answering an open question posed by M. Tiusanen in (EATCS Bulletin, Vol. 36, 1988, pp. 152–160).

Keywords: Analysis of algorithms, concurrency, conflict-free, Petri net

1. Introduction

In [8], 1-bounded conflict-free Petri nets have been proposed for modeling self-timed or self-synchronized circuits. In the study of self-timed circuits, a fundamental issue is to identify the situation in which the raising of a signal by one circuit element is in conflict with the lowering of the same signal by another element. In the modeling Petri net, this sort of conflict corresponds to the situation in which two designated transitions (representing the raising and lowering of a signal in the modeled circuit) become enabled simultaneously. An open problem was mentioned in [8] regarding the search for efficient algorithms to test pairwise concurrency of transitions in a 1-bounded conflict-free Petri net. More precisely, we are interested in determining, given a Petri net and two transitions, whether there exists a reachable marking in which the two transitions are enabled simultaneously. This problem will be referred to as the *concurrency problem* throughout the remainder of this paper.

For general Petri nets, the concurrency problem can be equated with the covering problem (of [7]); thus, it is EXPSPACE-complete (exponential space complete). Here, however, we are only interested in a "restricted" class of Petri nets, namely the class of 1-bounded conflict-free Petri nets.

The class of conflict-free Petri nets has received much attention from the viewpoint of computational complexity. Such efforts can be found in [1–6]. In particular, the boundedness problem has been shown to be PTIME-complete [4], the reachability problem has been proved to be NP-complete [2], whereas the containment and equivalence problems have been shown to be Π_2^p (the second level of the polynomial time hierarchy)

0020-0190/91/$03.50

complete [2]. If we further restrict ourselves to *bounded* conflict-free Petri nets, the upper bound of reachability can be improved to PTIME [3]. In view of the above, a natural approach to attack the concurrency problem is to relate the problem to those mentioned above. (For example, if we could show the concurrency problem to be reducible to the reachability problem, a polynomial time solution would follow immediately.) Unfortunately, conventional reduction techniques for general Petri nets do not seem to work for conflict-free Petri nets. This is mainly because of the fact that, in many cases, adding a new transition to a conflict-free Petri net could easily render the new Petri net non-conflict-free.

In this paper, we will show the concurrency problem for 1-bounded conflict-free Petri nets to be solvable in polynomial time. In solving the problem, we adopt a novel technique which may have other applications to the analysis of Petri nets.

2. Definitions

Let $\mathbb{Z}$ ($\mathbb{N}$) denote the set of (nonnegative) integers, and $\mathbb{Z}^k$ ($\mathbb{N}^k$) the set of vectors of k (nonnegative) integers. For a k-dimensional vector v, let $v(i)$, $1 \leqslant i \leqslant k$, denote the ith component of v. For a given value of k, let $\mathbf{0}$ denote the vector of k zeros (i.e., $\mathbf{0}(i) = 0$ for $i = 1, \ldots, k$).

A *Petri Net* (PN, for short) is a 4-tuple (P, T, ϕ, μ_0), where P is a finite set of *places*, T is a finite set of *transitions*, ϕ is a *flow function*,

$$\phi : (P \times T) \cup (T \times P) \to \{0, 1\},$$

and μ_0 is the *initial marking*, $\mu_0 : P \to \mathbb{N}$. A *marking* is a mapping $\mu : P \to \mathbb{N}$. A transition $t \in T$ is *enabled* at a marking μ iff for every $p \in P$, $\phi(p, t) \leqslant \mu(p)$. A transition t may *fire* at a marking μ if t is enabled at μ. We then write $\mu \to^t \mu'$, where $\mu'(p) = \mu(p) - \phi(p, t) + \phi(t, p)$ for all $p \in P$. A sequence of transitions $\sigma = t_1 \ldots t_n$ is a *firing sequence* from μ_0 (or a firing sequence of (P, T, ϕ, μ_0)) iff $\mu_0 \to^{t_1} \mu_1 \to^{t_2} \cdots \to^{t_n} \mu_n$ for some sequence of markings $\mu_1, \ldots, \mu_n$. (We also write "$\mu_0 \to^\sigma \mu_n$".) We write "$\mu_0 \to^\sigma$ " to denote that σ is enabled and can be fired from μ_0, i.e., $\mu_0 \to^\sigma$ iff there exists a marking μ such that $\mu_0 \to^\sigma \mu$.

Given a sequence of transitions σ, we define $\#_\sigma$ to be a mapping $\#_\sigma : T \to \mathbb{N}$ such that $\#_\sigma(t) =$ the number of occurrences of t in σ. Let $\mu \to^\sigma \mu'$. The *value* of σ, denoted by $\Delta(\sigma)$, is defined to be $\mu' - \mu$ ($\in \mathbb{Z}^k$, where k is the number of places in the Petri net). We let $S(\sigma)$ denote the set of transitions occurring in σ, i.e.,

$$S(\sigma) = \{t \mid t \in T, \#_\sigma(t) > 0\}.$$

Let $\mathscr{P} = (P, T, \phi, \mu_0)$ be a PN. The *reachability set* of $\mathscr{P}$ is the set

$$R(\mathscr{P}) = \{\mu \mid \mu_0 \to^\sigma \mu \text{ for some } \sigma\}.$$

$\mathscr{P}$ is said to be *k-bounded* (for some $k \in \mathbb{N}$) iff $\forall \mu \in R(\mathscr{P})$, $\forall p \in P$, $\mu(p) \leqslant k$. Given a place s, we let

$$s^\bullet = \{t \mid \phi(s, t) = 1, t \in T\}$$

and

$${}^\bullet s = \{t \mid \phi(t, s) = 1, t \in T\}.$$

A place s and a transition t are on a *self-loop* iff $t \in s^\bullet$ and $t \in {}^\bullet s$, i.e., s is both an input and output place of t. $\mathscr{P}$ is said to be *conflict-free* iff for every place s, either

(1) $|s^\bullet| \leqslant 1$, *or*

(2) $\forall t \in s^\bullet$, t and s are on a self-loop.

In words, a Petri net is conflict-free if every place which is an input of more than one transition is on a self-loop with each such transition [5,6]. In a conflict-free Petri net, once a transition becomes enabled, the only way to disable the transition is to fire the transition itself. (That is, $\forall t, t' \in T$, $t \neq t'$, $\mu \to^t \mu'$ and $\mu \to^{t'}$ implies $\mu' \to^{t'}$.) Notice that marked graphs are conflict-free, although the converse need not be true.

The *concurrency problem* for Petri nets is that of, given a Petri net $\mathscr{P} = (P, T, \phi, \mu_0)$ and two transitions u and v, determining whether there exists a reachable marking μ (i.e., $\mu \in R(\mathscr{P})$) such that both u and v are enabled in μ.

In the following section, we will show the problem to be solvable in polynomial time.

3. The complexity bound

We first show that, given two transitions u and v, if a path simultaneously enabling u and v exists, then there must exist a short "witness" which can be partitioned into "segments". We then use a lemma from [4] to construct those segments one by one in polynomial time to answer the concurrency problem. First, we present some lemmas to set the stage for our polynomial time algorithm.

Lemma 3.1. *Let $\mu_0 \to^{\sigma} \mu$ be a computation enabling transitions u and v simultaneously. Then there exist σ_1 and σ_2 such that*

(1) $\#_{\sigma} = \#_{\sigma_1\sigma_2}$,

(2) $\mu_0 \to^{\sigma_1\sigma_2} \mu$,

(3) $S(\sigma_2) \subseteq S(\sigma_1)$, *and*

(4) $\forall r$, $\#_{\sigma_1}(r) \leqslant 1$.

In words, $\sigma_1\sigma_2$ is a rearrangement of σ such that if a transition occurs in σ, it can also be found in σ_1; in addition, no transition in σ_1 appears more than one time in σ_1.

Proof. First, we claim that if $\mu_0 \to^{\delta_1\delta_2 t} \mu'$ (where δ_1, δ_2 are sequences of transitions and t is a transition) and $\forall$ transitions r, $S(\delta_2) \subseteq S(\delta_1)$, $0 \leqslant \#_{\delta_1}(r) \leqslant 1$, and $t \notin S(\delta_1)$, then $\mu_0 \to^{\delta_1 t \delta_2} \mu'$.

To prove the above claim, assume that t is not enabled in μ'', where $\mu_0 \to^{\delta_1} \mu''$. This implies one of t's input places, say p, must be empty in μ''. However, t is enabled in μ''', where $\mu_0 \to^{\delta_1\delta_2} \mu'''$, indicating the existence of a transition, say t' in δ_2, which deposits a token to p. Since $S(\delta_2) \subseteq S(\delta_1)$, t' must be in δ_1. Since the Petri net is conflict-free and $t \notin S(\delta_1)$, $\mu''(p) \neq 0$—a contradiction. So t must be enabled in μ''. Since the Petri net is conflict-free and $t \notin S(\delta_2)$ (because $S(\delta_2) \subseteq S(\delta_1)$ and $t \notin S(\delta_1)$), the firing of t in μ'' will not affect the enabledness of the subsequent transition sequence δ_2. This completes the proof of the claim. □

The next lemma indicates that if u and v can become enabled simultaneously, then there exists a short witness satisfying certain properties. More precisely, we have:

Lemma 3.2. *If $\mu_0 \to^{\sigma} \mu$ is the shortest (or one of the shortest) computation enabling transitions u and v simultaneously, then σ can be rearranged into $\sigma_1\sigma_2 \cdots \sigma_k$ such that*

(1) $\mu_0 \to^{\sigma_1\sigma_2\cdots\sigma_k} \mu$,

(2) $\forall 1 \leqslant i \leqslant k$, $\forall$ *transitions* r, $\#_{\sigma_i}(r) \leqslant 1$,

(3) $\forall 1 \leqslant i \leqslant k-1$, $S(\sigma_{i+1}) \subseteq S(\sigma_i)$, *and*

(4) $k \leqslant n$ *and* $|\sigma| \leqslant n^2$, *where n is the number of transitions of the Petri net.*

Proof. Conditions (1)–(3) can be derived easily by repeatedly applying Lemma 3.1. In what follows, we prove (4).

We first show that $k \leqslant n$. Suppose this is not the case. Then there must exist a j, $1 \leqslant j \leqslant k-1$ such that $S(\sigma_j) = S(\sigma_{j+1})$ (since $S(\sigma_1)$, $S(\sigma_2), \ldots, S(\sigma_k)$ forms a "shrinking" sequence of nonempty sets (i.e., $S(\sigma_1) \supseteq S(\sigma_2) \supseteq \cdots \supseteq S(\sigma_k)$); furthermore, there are at most n transitions in the Petri net). Consider the following three cases:

(1) $\Delta(\sigma_j)$ contains a negative component: Since the Petri net is 1-bounded and conflict-free σ_{j+1} could never be fired.

(2) $\Delta(\sigma_j) > \mathbf{0}$: In this case, "pumping" σ_j infinitely many times will render the Petri net unbounded—a contradiction.

(3) $\Delta(\sigma_j) = \mathbf{0}$: In this case, σ_j can be removed without affecting the simultaneous enabledness of u and v. This contradicts the assumption that σ be the shortest.

$|\sigma| \leqslant n^2$ follows immediately from $k \leqslant n$ and the fact that σ_i, $\forall i$, does not contain any transition more than once. □

Even though Lemma 3.2 allows us to put a polynomial upper bound on the length of the shortest path enabling u and v, a polynomial time algorithm for the concurrency problem does not follow immediately. However, we can use the "maximum" sequence of transitions enabled in the starting marking of that segment, rather than the exact σ_i stated in Lemma 3.2. It is then possible to construct a path (even though it may no longer be the shortest; it is still polynomial in length) in polynomial time to enable u and v, if any such path exists.

Lemma 3.3. *Let $\mu_0 \rightarrow^{\sigma_1\sigma_2} \mu$ be a computation enabling transitions u and v simultaneously. If there exists a sequence σ_1' such that*

(1) $\mu_0 \rightarrow^{\sigma_1'}$,

(2) $\forall$ *transitions* r, $\#_{\sigma_1}(r) \leqslant \#_{\sigma_1'}(r) \leqslant 1$,

(3) $S(\sigma_2) \subseteq S(\sigma_1)$,

(4) $u, v \notin S(\sigma_1') - S(\sigma_1)$,

then $\mu_0 \rightarrow^{\sigma_1'\sigma_2} \mu'$, for some μ', and u and v are enabled in μ'.

Proof. According to conditions (2) and (3), $S(\sigma_1') - S(\sigma_1)$ does not contain any transition belonging to σ_2. This, in conjunction with the conflict-freedom property, ensures that no transition in σ_2 will become disabled as a result of firing transitions in $S(\sigma_1') - S(\sigma_1)$. Hence, $\mu_0 \rightarrow^{\sigma_1'\sigma_2} \mu'$, for some μ'. In addition, (4) implies that u and v are enabled in μ'. □

The following lemma indicates the existence of a polynomial time algorithm to construct the maximum sequence of transitions in a given marking. In addition, the sequence consists of each of the enabled transitions exactly once.

Lemma 3.4 [4]. *Given a conflict-free Petri net $\mathscr{P} = (P, T, \phi, \mu_0)$, we can construct in polynomial time a path σ enabled in μ_0 in which no transition in σ is used more than once, such that if some transition t is not used in σ, then there is no path (emanating from μ_0) in which t is used.*

Lemma 3.5. *Given a 1-bounded conflict-free Petri net $\mathscr{P} = (P, T, \phi, \mu_0)$, let $\mu_0 \rightarrow^{\sigma_1\sigma_2\cdots\sigma_k} \mu$, $1 \leqslant k \leqslant n$, be the shortest computation guaranteed by Lemma 3.2. Let T_i be $T - (\{u, v\} - S(\sigma_i))$, and ϕ_i be the restriction of ϕ on T_i. Suppose $\sigma_1'\sigma_2'\cdots\sigma_k'$ is a sequence of transitions defined recursively as follows: $\forall 1 \leqslant i \leqslant k$, σ_i' is the sequence of transitions guaranteed by Lemma 3.4 with respect to $(P, T_i, \phi_i, \mu_{i-1})$, where $\mu_0 \rightarrow^{\sigma_1'\cdots\sigma_{i-1}'} \mu_{i-1}$. Then $\mu_0 \rightarrow^{\sigma_1'\sigma_2'\cdots\sigma_k'} \mu'$ is a computation enabling u and v simultaneously (i.e., u and v are enabled in μ').*

Proof. First consider σ_1'. It is easy to see the following:

(1) $\forall r$, $\#_{\sigma_1}(r) \leqslant \#_{\sigma_1'}(r) \leqslant 1$.

$\#_{\sigma_1}(r) \leqslant 1$ and $\#_{\sigma_1'}(r) \leqslant 1$ are due to Lemmas 3.2 and 3.4, respectively. (That is, no transition occurs more than once in σ_1 and σ_1'.) According to Lemma 3.4, if a transition is not in σ_1', then there is no path in which the transition is used. Hence, $\forall r$, $\#_{\sigma_1}(r) \leqslant \#_{\sigma_1'}(r)$.

(2) $S(\sigma_2 \cdots \sigma_k) \subseteq S(\sigma_1)$—Lemma 3.2.

(3) $u, v \notin S(\sigma_1') - S(\sigma_1)$.

Recall that $T_1 = T - (\{u, v\} - S(\sigma_1))$ and σ_1' is with respect to (P, T_1, ϕ_1, μ_0). Hence, if u (v) is not in σ_1, then it is not in σ_1' either.

Using Lemma 3.3, we immediately have that

$$\mu_0 \rightarrow^{\sigma_1'} \mu_1 \rightarrow^{\sigma_2\cdots\sigma_k} \mu'',$$

for some μ'', and u and v are enabled in μ''. Starting in μ_1 and repeatedly applying the above argument, we can easily show that $\mu_0 \rightarrow^{\sigma_1'\sigma_2'\cdots\sigma_k'} \mu'$ is a computation enabling u and v simultaneously. □

The above lemma suggests an iterative way of constructing a path to enable u and v simultaneously, if such a path exists. In each stage, we construct a sequence of maximum length which is enabled in the resulting marking of the previous stage. The only problem with this approach is that for each i, the presence or absence of u (or v) in σ_i', $1 \leqslant i \leqslant k$, must coincide with that in σ_i. Since we have no knowledge of the transitions used in σ_i, how do we know whether we should include or exclude u and v in constructing the maximum sequence? At first glance, it seems that nondeterministic steps (for guessing the presence or absence of u and v in all k, $1 \leqslant k \leqslant n$, segments) were inevitable. A careful examination, however, reveals that such nondeterminism can be avoided by taking advantage of the "shrinking property" of the sequence $\sigma_1, \sigma_2, \ldots, \sigma_k$ mentioned in Lemma 3.2. More precisely, if u (or v) does not appear in, say σ_j, then u (or v) will never occur in subsequent segments $\sigma_{j+1}, \ldots, \sigma_k$. This is exactly the key property upon which our polynomial time algorithm relies.

Theorem 3.6. *The concurrency problem for 1-bounded conflict-free Petri nets is solvable in polynomial time.*

Proof. We let *Find-Max-Seq*(P, T, ϕ, μ_0) (where (P, T, ϕ, μ_0) is a conflict-free Petri net) be a procedure that outputs a sequence of transitions σ in which no transition in σ is used more than once, and if some transition t is not used in σ, then there is no path in (P, T, ϕ, μ_0) in which t is used. The existence of such a procedure is guaranteed by Lemma 3.4.

We construct the sequence $\sigma_1'\sigma_2' \cdots \sigma_k'$ stated in Lemma 3.5. In our procedure, we use two variables, namely i and j, to represent the indices of the segments in which u and v, respectively, are *not* used for the first time. (More precisely, u (v) is assumed to be used in σ_l', $1 \leqslant l \leqslant i-1$ (σ_l', $1 \leqslant l \leqslant j-1$), but not in σ_r', $i \leqslant r \leqslant k$ (σ_r', $j \leqslant r \leqslant k$).) Since we do not know in advance the exact values of i and j in the path enabling u and v, we consider all possibilities of i and j, each of which ranges from 1 to $n+1$. (For example, $i=1$ indicates that u was never used; $i=n+1$ indicates that u is used in every segment.) The algorithm is as follows.

Procedure Concurrent-Pair$(P, T, \phi, \mu_0, u, v)$

/ * $u, v \in T$; This procedure is to determine whether transitions u and v can become enabled simultaneously in a 1-bounded conflict-free Petri net (P, T, ϕ, μ_0). * /

```
for i := 1 to n + 1 do
  for j := 1 to n + 1 do
    begin
      T' := T;
      μ := μ0;
      for l := 1 to n do
        begin
          if i = l then T' := T' − {u};
          if j = l then T' := T' − {v};
          σ := Find-Max-Seq(P, T', φ', μ);
          let μ' be the marking such that
          μ →^σ μ';
          if u and v are enabled in μ'
            then EXIT and return "YES"
            else μ := μ'
        end
    end
return "NO"
```

The procedure should be quite easy to understand. We thus have a polynomial time algorithm for the concurrency problem. □

4. Conclusion

It was mentioned as an open problem in [8] whether, given a 1-bounded conflict-free Petri net, the concurrency problem can be solved efficiently. In this article, we have answered the above question in an affirmative way by demonstrating a polynomial time algorithm for the problem. Since 1-bounded conflict-free Petri nets are known to be useful for modeling self-timed circuits, our result might have applications with respect to the verification of self-timed circuits.

Theoretically, it is of interest to know whether our polynomial time result can be extended to (bounded) conflict-free Petri nets. We must point out that our technique in this paper relies heavily on the fact that the Petri net is 1-bounded.

Acknowledgment

The author would like to thank the anonymous referees for suggestions that improved the presentation of this paper.

References

[1] S. Crespi-Reghizzi and D. Mandrioli, A decidability theorem for a class of vector addition systems, *Inform. Process. Lett.* **3** (3) (1975) 78–80.

[2] R. Howell and L. Rosier, Completeness results for conflict-free vector replacement systems, *J. Comput. System Sci.* **37** (1988) 349–366.

[3] R. Howell and L. Rosier, On questions of fairness and temporal logic for conflict-free Petri nets, in: G. Rozenberg, ed., *Advances in Petri Nets 1988*, Lecture Notes in Computer Science **340** (Springer, Berlin, 1988) 200–226.

[4] R. Howell, L. Rosier and H. Yen, An $O(n^{1.5})$ algorithm to decide boundedness for conflict-free vector replacement systems, *Inform. Process. Lett.* **25** (1987) 27–33.

[5] N. Jones, L. Landweber and Y. Lien, Complexity of some problems in Petri nets, *Theoret. Comput. Sci.* **4** (1977) 277–299.

[6] L. Landweber and E. Robertson, Properties of conflict-free and persistent Petri nets, *J. ACM* **25** (3) (1978) 352–364.

[7] C. Rackoff, The covering and boundedness problems for vector addition systems, *Theoret. Comput. Sci.* **6** (1978) 223–231.

[8] M. Tiusanen, Some unsolved problems in modeling self-timed circuits using Petri nets, EATCS Bulletin **36** (1988) 152–160.