Fundamenta Informaticae XX (2004) 1–23
IOS Press

A Polynomial-Time Algorithm for Checking Consistency of Free-Choice Signal Transition Graphs

Javier Esparza[*]
Institute for Formal Methods in Computer Science
University of Stuttgart
esparza@informatik.uni-stuttgart.de

Abstract. Signal Transition Graphs (STGs) are one of the most popular models for the specification of asynchronous circuits. A STG can be implemented if it admits a so-called consistent and complete binary encoding. Deciding this is EXPSPACE-hard for arbitrary STGs, and so a lot of attention has been devoted to the subclass of free-choice STGs, which offers a good compromise between expressive power and analizability. In the last years, polynomial time synthesis techniques have been developed for free-choice STGs, but they assume that the STG has a consistent binary encoding. This paper presents the first polynomial algorithm for checking consistency.

1. Introduction

Asynchronous circuit design is attracting increasing interest due to some important potential advantages, like absence of clock skew problems and low power consumption [2, 7]. Signal Transition Graphs (STGs) [5, 4, 20] are one of the most popular specifi cation models for asynchronous circuits. They are Petri nets in which the fi ring of a transition is interpreted as rising or falling of a signal in the circuit. Transitions corresponding to rising (falling) of signal a are labelled by a^+ (a^-). Signals are partitioned into input and output signals, which are supposed to be controlled by the environment and the circuit, respectively.

Given a STG, the synthesis problem consists in producing an asynchronous circuit exhibiting the same behavior. Such a circuit is known to exist if the state graph of the STG admits a *consistent* and *complete binary encoding*. A binary encoding is a function that assigns to each reachable state a vector of booleans, one for each signal, indicating if the signal is up (1) or down (0). Loosely speaking, an encoding is consistent if it respects the semantics of transition labels: e.g., if the fi ring of a transition labelled by a^+ leads from state s to state s', then signal a must have value 0 in s and value 1 in s'. In

[*]This work was mostly done while the author was at the University of Edinburgh.

particular, occurrences of a^+ and a^- must alternate along every firing sequence. Completeness requires the binary code of a state to contain enough information to determine the behavior of the circuit; if two states have the same code, then they have to enable exactly the same output signals.

Early techniques for the synthesis of asynchronous circuits from STG specifications proceeded by first constructing the state graph of the STG, then computing a consistent and complete binary encoding (if it exists), and then synthesizing the circuit from the encoding. This approach is fairly well understood (see e.g. [16, 22, 12]), and there exists good tool support, e.g. like that provided by the Petrify tool [6]). If these techniques are used, checking consistency and completeness are minor problems, since the check can be performed in linear time in the size of the reachability graph.

However, these approaches suffer strongly from the state explosion problem: the number of states of the state graph can grow super-exponentially in the size of the STG, or even be infinite. As synthesis tools for asynchronous systems start to mature, the size of STGs increases and techniques based on the state graph become obsolete. Therefore, much effort is being devoted to synthesis techniques that avoid the construction of the state graph. In this new setting consistency and completeness become a major problem. Checking them is EXPSPACE-hard in the size of the STG, even if its state graph is known to be finite, a result that follows easily from the EXPSPACE-hardness of the reachability problem for Petri nets [17, 11]. For 1-bounded STGs, in which a place can contain at most 1 token (the case most common in practice), checking consistency and completeness is still PSPACE-complete. PSPACE-hardness follows easily from the PSPACE-hardness of the reachability problem for 1-bounded Petri nets [11], while membership in PSPACE is trivial.

In order to cope with this problem, syntactically defined subclasses of STGs have been studied. The class with the best compromise between expressive power and analizability are *free-choice* STGs [5], a class that allows to model both nondeterminism (necessary for modelling the environment) and concurrency (essential for asynchronous modelling), but restricts their interplay. As a matter of fact, STGs were originally defined in [5] as free-choice nets, and many papers still identify STGs with free-choice STGs. Loosely speaking, a STG is free-choice if for every place p, whenever some output transition of p is enabled, all output transitions of p are enabled, and so it is always possible "to freely choose" which of them fires. This is an adequate model of the behavior of the environment, which should be able to freely produce any input signal to the circuit. Many asynchronous circuits can be naturally specified using free-choice nets, although they are not powerful enough to model arbiters.

Free-choice STGs and subclasses thereof have been studied in numerous papers (see e.g. [5, 3, 18, 21]). A number of techniques exist for the automatic implementation of STGs with consistent and complete encodings. In [3] it is also shown how to transform a free-choice STG having a consistent encoding into another one which admits a consistent and complete encoding. The problem of checking consistency has been studied in [18], where two polynomially checkable conditions for consistency, one sufficient and the other necessary, are presented. However, the exact computational complexity of checking consistency and completeness is still unknown.

In this paper we attack the consistency problem with results of the theory of free-choice Petri nets obtained in the early 90s [9]. We show that consistency can be checked in polynomial time for free-choice STGs known to be bounded, deadlock-free, and cyclic (meaning that the initial marking can be reached from any other reachable marking). These are standard conditions used by all papers on the subject since Chu's work [5], and in particular by [3, 18]. These conditions hold[1] for many real specifications, they

[1]Or can be artificially made to hold, for instance by adding transitions that 'restore' the initial marking.

simplify the synthesis procedures, and can be checked in polynomial time [9].

The paper is organized as follows. Section 2 contains basic definitions. Section 3 describes the checking procedure and proves its correctness. The resulting algorithm is presented in Section 4, and its complexity is analyzed. Finally, Section 5 contains some brief conclusions.

2. Basic definitions

A *net* is a triple (P, T, F), where P and T are disjoint sets of *places* and *transitions*, respectively, and F is a function $(P \times T) \cup (T \times P) \to \{0, 1\}$. Places and transitions are generically called *nodes*. Places are graphically represented as circles, and transitions as boxes. If $F(x, y) = 1$ then we say that there is an *arc* from x to y. The *preset* of a node x, denoted by ${}^\bullet x$, is the set of its input nodes, i.e., the set $\{y \in P \cup T \mid F(y, x) = 1\}$. The *postset* of x, denoted by $x^\bullet$, contains its output nodes, i.e., the set $\{y \in P \cup T \mid F(x, y) = 1\}$.

A *marking* M of a net (P, T, F) is a mapping $P \to \mathbb{N}$ (where $\mathbb{N}$ denotes the natural numbers including 0). Graphically, a marking is represented by drawing $M(p)$ tokens on the circle representing the place p. A marking M *enables* a transition t if it puts at least one token on each place $p \in {}^\bullet t$, i.e. if $M(p) > 0$ for each $p \in {}^\bullet t$. If t is enabled at M, then it can *fire* or *occur*, and its occurrence *leads to* a new marking L, obtained by removing a token from each place in the preset of t, and adding a token to each place in its postset; formally, $L(p) = M(p) + F(t, p) - F(p, t)$ for every place p. $M \xrightarrow{t} L$ denotes that t is enabled at M and that its occurrence leads to L.

A *Petri net* is a pair (N, M_0) where N is a net and M_0 is a marking of N, called the *initial marking*. A sequence of transitions $\sigma = t_1 t_2 \ldots t_n$, $n \geq 0$, is an *occurrence sequence from M to M'* if there exist markings $M_1, M_2, \ldots, M_{n-1}$ such that

$$M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots M_{n-1} \xrightarrow{t_n} M'$$

We usually omit the intermediate markings and write $M \xrightarrow{\sigma} M'$ for an occurrence sequence; $M \xrightarrow{\sigma}$ denotes that σ can be fired from M. The *Parikh mapping* of $\sigma \in T^*$, denoted by $\vec{\sigma}$, is the mapping $T \to \mathbb{N}$ that assigns to each transition the number of times it occurs in σ. A marking M is *reachable* if there exists an occurrence sequence from M_0 to M. The *reachability graph* of a Petri net is a labelled graph having the set of reachable markings as nodes, and the restriction of the $\xrightarrow{t}$ relations to the set of reachable markings as edges.

Signal transition graphs. We fix a finite alphabet $A = \{a_1, \ldots, a_n\}$ of *signals* partitioned into *input* and *output* signals. (All our results can be immediately extended to STGs with dummy transitions, we omit them for clarity.) Rising and falling of a signal a is denoted by a^+ and a^-, respectively. We call an element of $\mathcal{L} = A \times \{+, -\}$ a *label*. Loosely speaking, a signal transition graph is a Petri net whose transitions are labelled with elements of $\mathcal{L}$. Formally, a *signal transition graph* (STG) is a triple $S = (N, M_0, \ell)$, where (N, M_0) is a Petri net and ℓ is a surjective *labelling function* that assigns to each transition of N a label in $\mathcal{L}$.

The *state graph* of a STG $S = (N, M_0, \ell)$ has the reachable markings of (N, M_0) as nodes. If we have $M \xrightarrow{t} L$ in the reachability graph of (N, M_0), then the state graph contains an edge $M \xrightarrow{\ell(t)} L$.

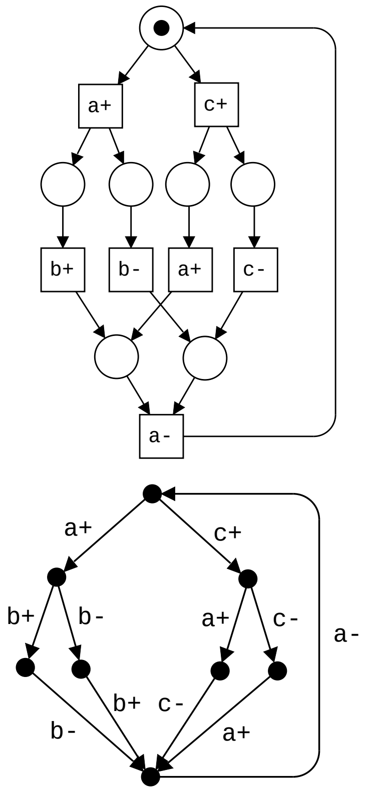

Figure 1. A free-choice STG and its state graph

Figure 1 shows a STG and its state graph. The node at the top of the state graph corresponds to the initial marking of the Petri net.

A STG is a specification of the behavior of the circuit under some assumptions on the environment. A STG $S = (N, M_0, \ell)$ with k places is implementable if there exists a *state coding mapping* λ that associates to each reachable marking M of (N, M_0) a vector of *signal values* $\lambda(M) \in \{0, 1\}^k$ satisfying the following two properties:

- *Consistency*. If $M \xrightarrow{t} L$ and t is labelled by a_i^+, then the i-th components of $\lambda(M)$ and $\lambda(L)$ are 0 and 1, respectively, and all other components have the same value in $\lambda(M)$ and $\lambda(L)$. If t is labelled by a_i^-, then the i-th components of $\lambda(M)$ and $\lambda(L)$ are 1 and 0, respectively, and all other components have the same value in $\lambda(M)$ and $\lambda(L)$.

- *Completeness*: if two different reachable markings M, L satisfy $\lambda(M) = \lambda(L)$, then they enable exactly the same output labels.

Consistency is obviously necessary for implementability. Completeness is necessary because the state of an implementation is completely determined by the signal values of all signals. Therefore, if some output signal is enabled at M but not at L, M and L must correspond to different states of the implementation, and so they must differ in the value of at least one signal.

We say that a STG is consistent or complete if it has a consistent or complete binary encoding, respectively. The STG of Figure 1 is not consistent. To see why, let M be the marking reached by firing a^+. Since M enables both b^+ and b^-, a consistent encoding λ must satisfy both $\lambda(M)(b) = 0$ and $\lambda(M)(b) = 1$, and so it cannot exist. If the transition at the top labelled by a^+ and the transition labelled by b^- swap their labels, then the STG becomes consistent.

A marking M of a STG is *n-bounded* if $M(p) \leq n$ for every place p. A STG is n-bounded if all its reachable markings are n-bounded. Since the circuit implementation of a STG can be seen as a finite object with at most 2^n states, where n is the number of signals, STGs used in practice are bounded (even though in principle unbounded STGs could make sense), and in fact most of them are even 1-bounded. A STG is *deadlock-free* if every reachable marking enables some transition. It is *cyclic* if for every reachable marking there is a sequence σ such that $M \xrightarrow{\sigma} M_0$, where M_0 is the initial marking. STGs specifying circuits which continuously interact with their environment are deadlock-free. Many are also cyclic, since after each round of interaction the circuit usually returns, or at least may return, to a home state. A STG is *well formed* if it is deadlock-free, bounded, and cyclic. The STG of Figure 1 is well formed. If we change the initial marking to the one that puts one token on the input places of the transitions labelled by b^+ and c^-, and no tokens elsewhere, then the STG is still live and bounded, but no longer cyclic.

Checking well formedness of STGs or consistency of well formed STGs is EXPSPACE-hard. Therefore we introduce a syntactically defined subclass of STGs. A STG is *free-choice* if its underlying net (P, T, F) satisfies the following property: for each place p and every transition t, if $F(p,t) = 1$ then $F(p', t') = 1$ for every $p' \in {}^\bullet t$, $t' \in p^\bullet$. In a free-choice STG, if some output transition of a place is enabled at a marking, then all its output transitions are enabled, and it is possible to "freely" choose among them.

An important property of well formed free-choice STGs is that they are live (see for instance Theorem 4.31 of [9]). A STG is *live* if for every reachable marking M and each transition t $M \xrightarrow{\sigma t}$ holds for some sequence σ of transitions. Moreover, well formedness of free-choice STGs can be checked in polynomial time (see e.g Corollary 6.18 and Theorem 8.12 of [9]).

We say that a STG is *alternating* if the signs of all signals alternate in all paths of its state graph. The STG of Figure 1 is not alternating: its state graph contains a path in which b^- occurs twice without any occurrence of b^+ in-between.

We conclude this section with a simple result showing that for live and cyclic systems consistency is equivalent to alternation.

Lemma 2.1. (1) Consistent STGs are alternating.

(2) Live, cyclic, and alternating STGs are consistent.

Proof:
(1) follows directly from the definition of consistency. For (2), let S be a live, cyclic, and alternating STG, and assign to each reachable marking M a binary coding $\lambda(M)$ as follows: $\lambda(M)(a) = 0$ $(= 1)$ if the state graph contains a path starting at M in which a^+ occurs before a^- (a^- before a^+). We prove that λ is well defined and consistent. Consistency follows immediately from the definition. To show that λ is well defined, let M be an arbitrary reachable marking. Since S is live and its labelling function is surjective, its state graph contains a path $M \xrightarrow{\sigma t}$ where t is labelled by a^+ or a^-. Assume now that

there are two paths π^+ and π^- such that a^+ occurs before a^- in π^+, and a^- before a^+ in π^-. By cyclicity, π^+ can be extended to a cycle from M to M (which may visit markings more than once). Since signals alternate along all paths, a^- occurs last in this cycle. Consider the path that starts with this last occurrence of a^-, reaches M, and then continues with π^-. In this path, the label a^- occurs twice without any intermediate occurrence of a^+, contradicting the assumption. □

3. Checking consistency

The next three sections present a three-step procedure for checking consistency of well formed free-choice STGs. We will make implicit use of Lemma 2.1 along the way.

The first two steps check non-autoconcurrency and balancedness, two necessary conditions for consistency. The third step is a divide-and-conquer procedure which checks consistency of STGs that pass the first two tests.

3.1. Checking non-autoconcurrency

A marking M *concurrently enables* two transitions t and u of a net if it puts at least one token in all input places of t and u, and at least two tokens on every input place of both t and u. It follows easily from the firing rule that if M concurrently enables t and u then there is a marking L such that both $M \xrightarrow{tu} L$ and $M \xrightarrow{ut} L$ are occurrence sequences. A STG is *non-autoconcurrent* if no reachable marking concurrently enables two transitions labelled by the same signal (even if the labels carry opposite signs). We have the following result:

Lemma 3.1. Consistent STGs are non-autoconcurrent.

Proof:
Assume that two transitions t and u are concurrently enabled at a reachable marking M. Then we have $M \xrightarrow{tu}$ and $M \xrightarrow{ut}$. Assume that both t and u are labelled by a signal a. If they are both labelled by a^+ or a^-, then alternation is violated. If, say, t is labelled by a^+ and u by a^-, then no consistent coding can exist: signal a cannot be assigned a code in M, because it can both rise and fall from M. □

By Lemma 3.1, if a well formed free-choice STG does not pass the non-autoconcurrency test, we can already dismiss it as non-consistent. The STG of Figure 1 does not pass the test, because b^+ and b^- can be concurrently enabled. Checking non-autoconcurrency is EXPSPACE-complete for arbitrary STGs (this can be easily proved using the techniques of [11]). Fortunately, for well formed free-choice STGs we can do much better: It is proved in [14] that non-autoconcurrency can be checked in polynomial time. For the sake of completeness, we include here a summary of this result.

The concurrency relation is usually defined as the set of pairs of transitions that can be concurrently enabled. We generalize the definition a bit. Given a node x of a STG S, we define the marking M_x as follows:

- if x is a place, then M_x is the marking that puts one token on x, and no tokens elsewhere;
- if x is a transition, then M_x is the marking that puts one token on every input place of x, and no tokens elsewhere.

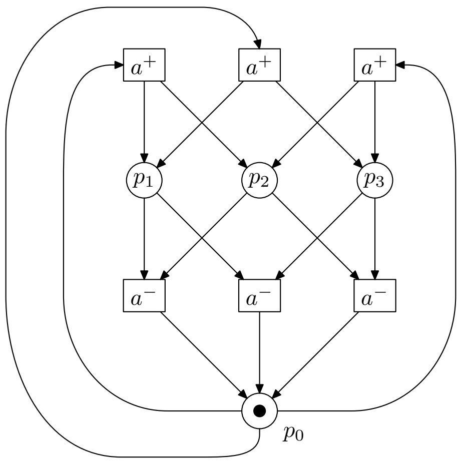

Figure 2. A well formed STG for which $\|\ \subset\ \|^s$

The *concurrency relation* $\|$ contains the pairs of nodes (x_1, x_2) such that $M \geq M_{x_1} + M_{x_2}$ for some reachable marking M. In particular, a pair (t_1, t_2) of transitions belongs to the concurrency relation if t_1, t_2 can occur concurrently from some reachable marking.

The *structural concurrency relation,* first presented in [13], is the smallest symmetric relation $\|^s$ on the nodes of S such that:

(i) for all places p, p', if $M_0 \geq M_p + M_{p'}$, where M_0 is the initial marking of S, then $(p, p') \in \|^s$;

(ii) for all transitions t, if $({}^\bullet t \times {}^\bullet t) \setminus id_P \subseteq \|^s$, then $(t^\bullet \times t^\bullet) \setminus id_P \subseteq \|^s$, where id_P denotes the identity relation on the places of S;

(iii) for all nodes x and for all transitions t, if $\{x\} \times {}^\bullet t \subseteq \|^s$, then $(x, t) \in \|^s$ and $\{x\} \times t^\bullet \subseteq \|^s$.

Loosely speaking, condition (i) states that any two places marked at the initial marking are structurally concurrent (actually, this is the case for a pair (p, p) only if M_0 puts at least two tokens on p). Condition (ii) states that if all the input places of a transition are structurally concurrent, then so are its output places. Condition (iii) states that if a node is structurally concurrent with all the input places of a transition, then it is also structurally concurrent with all its output places. It is easy to see that $\|\ \subseteq\ \|^s$. Figure 2 (taken from [13]) shows a STG for which $\|\ \subset\ \|^s$. The reader can check that $(p_1, p_0) \notin \|$. However, since $(p_1, p_2) \in \|^s$ and $(p_1, p_3) \in \|^s$, we have $(p_1, p_0) \in \|^s$.

So, in general, the structural concurrency relation provides only a sufficient, but not necessary condition for non-autoconcurrency. Fortunately, in the free-choice case (the STG of Figure 2 is non-free-choice), they coincide:

Theorem 3.1. ([14]) For live and bounded free-choice STGs, $\|\ =\ \|^s$.

The structural concurrency relation can be easily computed as a least fixed-point. In [14], two algorithms are presented which compute $\|^s$ in $O(|P|^2 \cdot |T| \cdot (|P| + |T|))$ time for arbitrary Petri nets and in $O(|P| \cdot (|P| + |T|)^2)$ time for free-choice Petri nets. Cortés [8] has observed that these algorithms are incorrect because they forget to maintain the $\|^s$ relation symmetric. Corrected versions kindly provided by him are shown in an Appendix.

3.2. Checking balancedness

A STG is *balanced* if for every signal a, every cycle of the state graph contains the same number of occurrences of a^+ and a^-. Clearly, balancedness is a necessary condition for alternation and hence for consistency. We show how to check balancedness.

Let $S = (N, M_0, \ell)$ be a STG, where $N = (P, T, F)$. A mapping $J \colon T \to \mathbf{Q}$, where $\mathbf{Q}$ denotes the rational numbers, is a *T-invariant* of N if $\sum_{t \in {}^\bullet p} J(t) = \sum_{t \in p^\bullet} J(t)$ for every place p. Notice that T-invariants form a vector space. A T-invariant J is *positive* (*semi-positive*) if $J(t) > 0$ ($J(t) \geq 0$) for all $t \in T$.

It is easy to show that if $M \xrightarrow{\sigma} M$ for some reachable marking M, then $\vec{\sigma}$ is a semi-positive T-invariant. However, given a semi-positive (integer) T-invariant J, it may not be possible to *activate* it, i.e., there may be no cycle $M \xrightarrow{\sigma} M$ in the reachability graph of (N, M_0) such that $J = \vec{\sigma}$. Fortunately, for well formed free-choice STGs a limited version of the converse holds. In order to state this result we need some definitions. Let the *support* of a semi-positive T-invariant be the set of transitions t such that $J(t) > 0$. A semi-positive T-invariant is *minimal* if there is no other semi-positive T-invariant with strictly smaller support.

Theorem 3.2. ([9], Theorems 5.17 and 5.20)
Let S be a well formed free-choice STG. All minimal T-invariants of N can be activated.

We use this result to check balancedness. Given a mapping $J \colon T \to \mathbf{Q}$ and a signal a we write $\#(J, a^+)$ ($\#(J, a^-)$) to denote the sum of $J(t)$ over all transitions t labelled by a^+ (a^-), and define $bal(J) \colon A \to \mathbb{N}$ by $bal(J)(a) = \#(J, a^+) - \#(J, a^-)$ for every signal a.

Theorem 3.3. Let S be a well formed free-choice STG. S is balanced iff $bal(J) = \mathbf{0}$ for every T-invariant J of S.

Proof:
($\Rightarrow$): We first prove $bal(J) = \mathbf{0}$ for semi-positive T-invariants. Let J be a semi-positive T-invariant. It follows easily from the definition of minimal T-invariant that there are minimal T-invariants $J_1, \ldots, J_k$ such that $J = J_1 + \ldots + J_k$. By Theorem 3.2, all these T-invariants can be activated and so, since S is balanced, we have $bal(J_1) = \ldots = bal(J_k) = \mathbf{0}$. Since $bal(J) = bal(J_1) + \ldots + bal(J_k)$, we get $bal(J) = \mathbf{0}$.
Let J now be an arbitrary T-invariant. By Theorem 2.38 of [9], S has a positive T-invariant J'. Let $J'' = J + kJ'$, where k is large enough to make J'' semi-positive. Since $bal(J) = bal(J'') - k \cdot bal(J')$ and $bal(J'') = bal(J') = \mathbf{0}$, we have $bal(J) = \mathbf{0}$.
($\Leftarrow$): Let c be an arbitrary cycle of the state graph of S. This cycle corresponds to a cycle $M \xrightarrow{\sigma} M$ of the reachability graph. Then $\vec{\sigma}$ is a T-invariant of S, and therefore $bal(\vec{\sigma}) = \mathbf{0}$, which means that c contains the same number of occurrences of a^+ and a^- for every signal a. □

The STG of Figure 3 (taken from [19] with some modifications) shows that Theorem 3.3 does not hold for arbitrary well formed STGs. The STG is balanced, but the mapping J that assigns 0 to the two transitions in the middle of the picture and 1 to the others is a T-invariant for which $bal(J) \neq \mathbf{0}$. Notice that this T-invariant cannot be activated.

T-invariants can be computed using linear algebra. The *incidence matrix* of a net N, denoted by $\mathbf{N}$, is a $|P| \times |T|$ matrix given by $\mathbf{N}(p, t) = F(t, p) - F(p, t)$. It is easy to see that $M \xrightarrow{\sigma} L$ implies

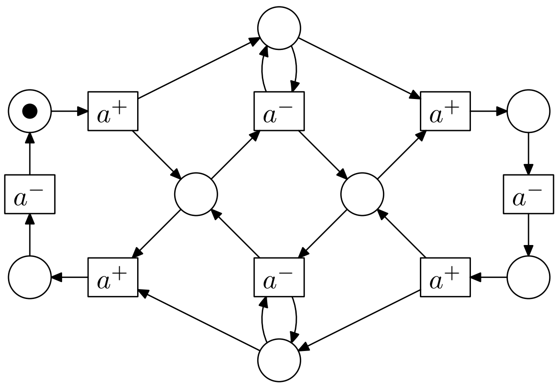

Figure 3. A well formed and balanced STG

$L = M + \mathbf{N} \cdot \vec{\sigma}$. (For $\sigma = t_1 \ldots t_n$, this is nothing but the equalities $L(p) = M(p) + F(t_1, p) - F(p, t_1) + \ldots + F(t_n, p) - F(p, t_n)$ written in matrix form.) A little calculation shows that J is a T-invariant if and only if $\mathbf{N} \cdot J = \mathbf{0}$. So we have the following corollary of Theorem 3.3:

Corollary 3.1. Let S be a well formed free-choice STG. S is balanced iff every rational solution of the linear system of equations $\mathbf{N} \cdot X = \mathbf{0}$ satisfies $bal(X) = \mathbf{0}$.

This corollary shows that balancedness can be decided by computing a basis of the solutions of $\mathbf{N} \cdot X = \mathbf{0}$ and checking that each element J of the basis satisfies $bal(J) = \mathbf{0}$.

3.3. A divide-and-conquer procedure

In the rest of the section we work with well formed free-choice STGs which are both balanced and non-autoconcurrent. We call these STGs *very well formed*.

The third step of our procedure to check alternation is a divide-and-conquer algorithm for checking alternation of very well formed free-choice STGs (i.e., of those well formed free-choice STGs which have passed both the non-autoconcurrency and the balancedness tests): we decompose the STG into two parts and show that checking alternation of the whole reduces to checking alternation of each part. Recall that, by Lemma 2.1, very well formed STGs are consistent if and only if they are alternating.

The divide-and-conquer procedure is best explained by considering the special case of STGs in which every transition has exactly one input and one output place. In what follows, these are called *state machine* STGs[2]. Figure 4 shows a very well formed state machine STG. This figure and the next ones use two drawing conventions. First, only the label of a transition is drawn (its surrounding box is omitted). Second, places having only one input and one output transition are also omitted. Therefore, an edge of the form, say, $a^+ \rightarrow b^-$, indicates that the STG contains an arc from a transition labelled by a^+ to a place, another arc from this place to a transition labelled by b^-, and that the place has no other input or output transitions. Tokens on the intermediate place are just drawn on the arc.

[2]Checking consistency of state machine STGs is a trivial problem. We use them in order to present some of the ideas of the free-choice case in a simple setting.

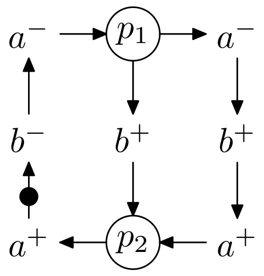

Figure 4. A very well formed state machine STG

Let S be a very well formed state machine STG. Let a *handle* be a simple path of (the underlying net of) S satisfying the following conditions: it starts and ends at a place, all its intermediate nodes have exactly one input and one output node, and after removing these intermediate nodes together with their adjacent arcs, the remaining STG is still strongly connected. In Figure 4, the path on the right that visits p_1, a^-, b^+, a^+, p_2 is a handle.

Without loss of generality, we assume that the intermediate places of S contain no tokens at the initial marking (if this is not the case, we can fi re transitions to remove those tokens; by cyclicity, alternation holds for the new STG if and only if it holds for the old one). Let H and R be STGs whose underlying nets are the handle and the remaining STG, respectively; the initial marking of H puts one token on its fi rst place and no tokens elsewhere, while the initial marking of R is the projection of the initial marking of S onto R. For the example of Figure 4, the STGs H and R are shown in Figure 5.

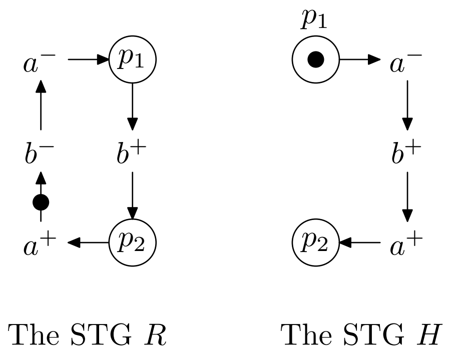

Figure 5. Dividing the STG of Figure 4

We would like to have that S is alternating if and only if H and R are alternating. However, Figure 4 shows that this is not true. In this example, S is not alternating because of the two consecutive a^- at the top of the fi gure, even though both H and R are alternating.

Our solution consists of modifying R to make sure that, if S is not alternating and H is alternating, then the new R is not alternating.

In order to see how to modify R, we look at the path of R visiting p_1, b^+, p_2. This is a 'mirror image' of H in R, i.e., a path of R with the same start and end nodes as H. Let α and β be the sequences of labels corresponding to H and its mirror image, i.e., $\alpha = a^-b^+a^+$ and $\beta = b^+$. Intuitively, the modifi cation of R should guarantee that if we can use α to produce a non-alternating sequence, then we can also use β, i.e., that α and β are 'equivalent' with respect to their potential for producing non-alternating sequences.

But how could a formal definition of 'equivalent' look like? It is not difficult to see that we could choose this one: for each signal a, if the projection of α on a is nonempty, then the projections of α and β on a begin with the same label and end with the same label. However, we have not been able to transform this definition into a polynomial divide-and-conquer procedure, and so we choose another one which also works: (1) $\vec{\alpha}$ and $\vec{\beta}$ have the same balance, (2) for each signal a, if a occurs in α then it also occurs in β, and (3) if a occurs in α and β then the first occurrence of a has the same sign in both sequences. In Figure 4, α and β are not equivalent because condition (2) is violated.

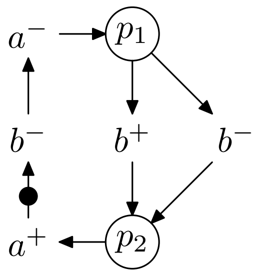

Figure 6. A state machine STG

So our modification of R should guarantee that α and β satisfy conditions (1), (2) and (3). Fortunately, it is easy to see that if (1) is violated then the STG is not very well formed. Figure 6 shows an example. We have $\alpha = b^-$ and $\beta = b^+$, and so condition (1) is violated; the STG contains a circuit with two occurrences of b^- and none of b^+, and so it is unbalanced. Since we assume that our initial STG is very well formed, we do not have to worry about ensuring condition (1): it holds automatically.

In order to guarantee (2) and (3), we insert a 'witness' in R, as shown in Figure 7 for the STG R of Figure 5. The witness 'records' that a occurs in α, and that the first occurrence of a has negative sign. Notice that, after introducing the witness, R is no longer alternating. The two consecutive a^- that were possible in S are now possible in the modified R as well.

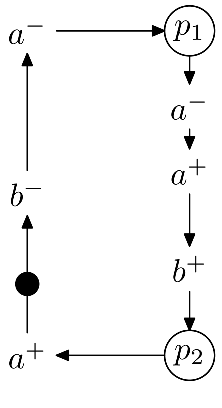

Figure 7. Inserting a witness in R

In the rest of the section we show how to extend these ideas to free-choice STGs. In particular, handles can no longer be just paths, due to the presence of concurrency. Section 3.3.1 introduces CP-subnets[3], which are a generalization of handles. Section 3.3.2 formalizes the insertion of witnesses. Section 3.3.3 shows the correctness of the divide-and-conquer steps: alternation holds for the original STG if and only if it holds for the parts. Finally, Section 3.3.4 shows how to directly check consistency of STGs that do not contain any CP-subnets.

3.3.1. Handles generalize to CP-subnets

A *transition-generated subnet* of a net $N = (P, T, F)$ is a net $N' = (P', T', F')$ such that $T' \subseteq T$, $P' = {}^\bullet T' \cup T'^\bullet$ (i.e., P' contains all input and output places of T' in the net N), and $F' = F \cap ((P' \times T') \cup (T' \times P'))$. Transition-generated subnets are just called subnets in what follows. A subnet $N' = (S', T', F')$ is *weakly connected* if for every partitioning of $P' \cup T'$ into two disjoint, nonempty sets X_1, X_2, there is $x_1 \in X_1$ and $x_2 \in X_2$ such that $(x_1, x_2) \in F$ or $(x_2, x_1) \in F$. It is *strongly connected* if $(x, y) \in (F')^*$ for every $x, y \in P' \cup T'$, where $(F')^*$ denotes the reflexive and transitive closure of F. Given a subnet N', we define $N \setminus N'$ as the subnet having $T \setminus T'$ as set of transitions. A place p of N' is called *entry* (*exit*) place if some transition of ${}^\bullet p$ ($p^\bullet$) belongs to $N \setminus N'$. All other places of N' are called *internal*. The output transitions of the entry places are called *entry transitions*. All other transitions of N' are called *internal*.

N' is a *CP-subnet* of N if

(i) it is nonempty and weakly connected,

(ii) every internal place has exactly one input and one output transition, and

(iii) the net $N \setminus N'$ is strongly connected.

The handle of Figure 5 is an example of CP-subnet. A more interesting example is the subnet of Figure 1 generated by the three transitions of the left half of the net labelled by a^+, b^+, and b^-. Finally, Figure 8 shows a free-choice STG, which in Figure 9 is divided into a CP-subnet, on the right, and the rest, on the left. (The fact that the initial markings of Figure 8 and Figure 9 are different will be explained later).

In what follows, $\overline{N}$ denotes a CP-subnet of a net N. Given a STG $S = (N, M_0, \ell)$, we denote by $S \setminus \overline{N}$ the STG whose underlying net is $N \setminus \overline{N}$, and whose initial marking and labelling function are the projections of M_0 and ℓ onto $N \setminus \overline{N}$. We will use the following result of [9]:

Lemma 3.2. ([9]) CP-subnets of well formed free-choice STGs have a unique entry transition.

A *flushing sequence of* $\overline{N}$ *in* S is an occurrence sequence $M \xrightarrow{\sigma} L$ of S satisfying the following two properties:

- M and L are reachable markings of S that enable no internal transitions of $\overline{N}$, and
- $\sigma = \overline{t}_e \tau$, where $\overline{t}_e$ is the unique entry transition of $\overline{N}$, and τ contains no occurrences of $\overline{t}_e$.

Intuitively, in a flushing sequence the entry transition of $\overline{N}$ fires once, and then the internal transitions of $\overline{N}$ fire for as long as possible. In the case of handles, a flushing sequence lets a token run along the handle, from its first to its last place, and so handles have one single flushing sequence. This is no longer

[3]This strange name is due to historical reasons.

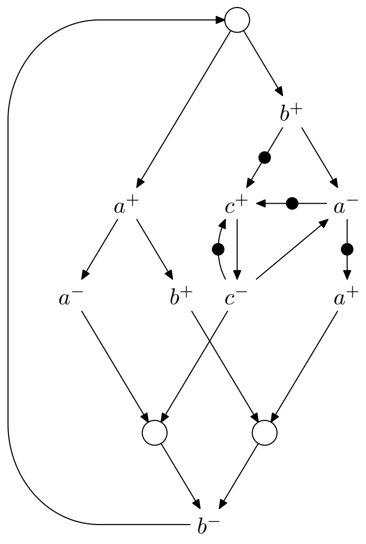

Figure 8. A free-choice STG

the case for CP-subnets, because concurrency among the transitions of $\overline{N}$ is possible, and so we can get a new flushing sequence by changing the order in which concurrently enabled transitions fi re. For instance, in the CP-subnet of Figure 9, there are three possible flushing sequences, corresponding to the sequences of labels $b^+a^-a^+c^+c^-$, $b^+a^-c^+a^+c^-$, and $b^+a^-c^+c^-a^+$. However, for very well formed STGs we can prove that CP-subnets enjoy many of the properties of handles:

Proposition 3.1. Let S be a very well formed free-choice STG, let $\overline{N}$ be a CP-subnet of S, and let $M \xrightarrow{\sigma} M'$ and $L \xrightarrow{\tau} L'$ be arbitrary flushing sequences of $\overline{N}$ in S. Then

(1) M, M', L, L' coincide on all internal places of $\overline{N}$,

(2) the projections of $\ell(\sigma)$ and $\ell(\tau)$ on each signal coincide, and

(3) every transition of $\overline{N}$ occurs exactly once in σ and τ.

Proof:
(1) This is shown in Lemma 9.2 of [9].
(2) We prove a stronger result, namely that the projections of σ and τ on the transitions labelled by a^+ or a^- coincide for every signal a. We start with some preliminaries. Since both M and L enable the entry transition of $\overline{N}$ and S is non-autoconcurrent, they put one token in all the entry places of $\overline{N}$. So, by (1), M and L coincide on all input places of all transitions of $\overline{N}$. Let K be the common projection of M and L on these places. Then $K \xrightarrow{\sigma}$ and $K \xrightarrow{\tau}$ are occurrence sequences of $\overline{N}$. We can now make use of a well-know property of persistent nets, a class that contains all nets in which places have at most one output transition [15]: if $K \xrightarrow{\upsilon_1}$ and $K \xrightarrow{\upsilon_2}$, then $K \xrightarrow{\upsilon}$ for some sequence υ such that $\vec{\upsilon} = \max\{\vec{\upsilon}_1, \vec{\upsilon}_2\}$, where $\max$ is defi ned componentwise.
After these preliminaries, assume that for some signal a the projections of $\ell(\sigma)$ and $\ell(\tau)$ on the transitions labelled by a^+ or a^- do not coincide. Let σ' and τ' be the longest prefi xes of σ and τ (possibly the empty

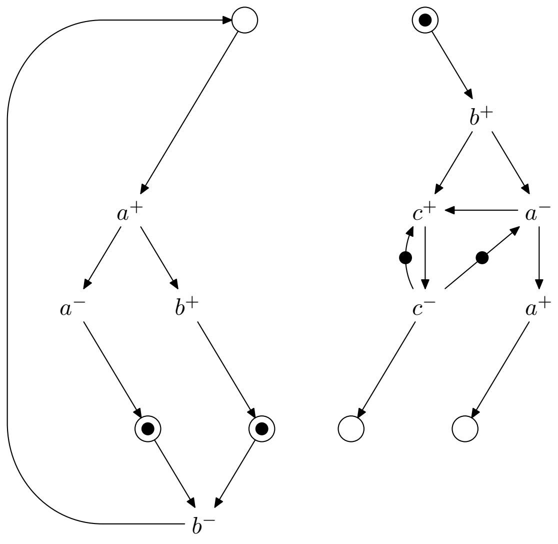

Figure 9. Dividing the STG of Figure 8

word) for which they coincide. Then $\sigma = \sigma'\sigma''t\sigma'''$ and $\tau = \tau'\tau''u\tau'''$, where σ'' and τ'' do not contain a-labelled transitions, $\ell(t), \ell(u) \in \{a^+, a^-\}$, and $\ell(t) \neq \ell(u)$. Let $X = \max\{\vec{\sigma'}, \vec{\tau'}\}$, $X_t = \max\{\vec{\sigma' t}, \vec{\tau'}\}$, and $X_u = \max\{\vec{\sigma'}, \vec{\tau' u}\}$. Let K', K_t and K_u be the markings reached by the sequences having X, X_t, and X_u as Parikh vector. Then, since $K_t = K' + N \cdot \vec{t}$ and $K_u = K' + N \cdot \vec{u}$, the marking K' enables both t and u. Since places of $\overline{N}$ have at most one output transition, t and u do not share any input place. So t and u are concurrently enabled at K', contradicting that S is very well formed.

(3) By liveness of S, every circuit of $\overline{N}$ is marked at M (otherwise the transitions of the circuit can never occur, because, since $\overline{N}$ is a CP-subnet, the circuit can never get marked). Moreover, since S is non-autoconcurrent and free-choice, M puts exactly one token on the entry places of $\overline{N}$ (otherwise any output transition of these places can occur concurrently with itself). Now, let t be an internal transition of $\overline{N}$. Since t is not enabled at M, it has an input place unmarked at M. If the input transition of this place is an internal transition, then it is again not enabled by M, and so in this way we can construct a path whose places are unmarked at M. Since this path cannot run into a circuit, because all circuits of $\overline{N}$ are marked, it must end at the entry place. Since the path contains only one token, t can occur at most once in the flushing sequence, because after its first occurrence the path becomes empty. That t occurs at least once follows from liveness. $\square$

So, even if in a very well formed free-choice STG a CP-subnet may have many different flushing sequences, they all start and end at markings that coincide on all the internal places of $\overline{N}$ (the markings may differ elsewhere), each transition occurs in them exactly once, and, for every signal a, their projections onto the labels a^+, a^- coincide.

Proposition 3.1 justifies the following definition. Given an arbitrary flushing sequence $M \xrightarrow{\sigma} L$ of $\overline{N}$ in S, we define the *characteristic marking* of $\overline{N}$, denoted by $\overline{M}$, as the marking that puts one token on each entry place of $\overline{N}$, coincides with M (and L) on the internal places, and puts no tokens on the exit places (unless they are also entry places, in which case they get one token). The *characteristic sequence* of $\overline{N}$ with respect to signal a, denoted by $\overline{\sigma}_a$, is the projection of $\ell(\sigma)$ onto a^+ and a^-. The characteristic marking of the CP-subnet of Figure 9 is the one shown in the fi gure, and we have $\overline{\sigma}_a = a^- a^+$, $\overline{\sigma}_b = b^+$, and $\overline{\sigma}_c = c^+ c^-$.

In the case of state machine STGs we divide the STG into a handle H and the rest R. In the free-choice case, the rôle of H is played by the STG S_{CP} having $(\overline{N}, \overline{M})$ as underlying Petri net. By Proposition 3.1(2), checking consistency of S_{CP} is very simple: if suffi ces to construct *any* flushing sequence (all of them have linear length), and check if they are alternating.

3.3.2. Inserting witnesses

In this section we defi ne the STG playing the rôle of R. Let S be a very well formed free-choice STG with a CP-subnet $\overline{N}$. Without loss of generality, we assume that the initial marking M_0 of S does not enable any internal transition of $\overline{N}$. If this were not the case, we would let the internal transitions of $\overline{N}$ occur for as long as possible. For instance, in the STG of Figure 8 we let the transitions labelled by c^+, c^-, and a^+ occur. Since S is cyclic, the resulting STG is alternating if and only if S is alternating.

Let A' be the set of signals a such that $\overline{\sigma}_a$ is of the form $a^+ \alpha a^-$ or $a^- \alpha a^+$ for some $\alpha \in \mathcal{L}^*$. The procedure to construct S_R is best described in an informal but hopefully precise way, since a formal defi nition would be diffi cult to read.

- Start with the STG $S \setminus \overline{N}$.
- Let P_e be the set of entry places of $\overline{N}$ (these are the input places of the unique entry transition $\overline{t}_e$), which belong to $S \setminus \overline{N}$. Let $T' = P_e^\bullet \setminus \{\overline{t}_e\}$. Remove all the arcs (p, t) of $N \setminus \overline{N}$ such that $p \in P_e$ and $t \in T'$.
- Construct a net consisting of one single path, starting at a transition and ending at a place. The path, which we call the *witness* of a, contains two transitions t_a, u_a for each label $a \in A'$, with t_a preceding u_a. If $\overline{\sigma}_a$ is of the form $a^+ \alpha a^-$, then t_a is labelled by a^+, and u_a is labelled by a^-. If it is of the form $a^- \alpha a^+$, then t_a is labelled by a^- and u_a is labelled by a^+. The order between transitions corresponding to different labels can be chosen arbitrarily. Let t_f be the fi rst transition of the path, and let p_l be its last place.
- Add arcs $\{(p, t_f) \mid p \in P_e\}$ and $\{(p_l, t) \mid t \in T'\}$.

(At this point, the reader may ask why do we introduce witnesses only for the actions of the set A'. While we could safely introduce witnesses for all signals whose characteristic sequence is nonempty, it turns out to be superfluous.)

In the case of Figure 9 the set A' contains the signals a and c. The STG S_R is shown in Figure 10.

Notice that if alternation holds for S, then it also holds for S_R. The converse, however, is not true, as shown by Figure 4.

3.3.3. Correctness of the divide-and-conquer strategy

The goal of this section is to prove the two essential correctness properties of the divide-and-conquer procedure. First, we have to show that alternation holds for S (and for all signals) if and only if it holds for both S_{CP} and S_R, as defi ned in Sections 3.3.1 and 3.3.2. Second, since we assume that S is a very well formed free-choice STG, we have to prove that S_R is also very well-formed, so that the procedure can be iterated with S_R. The reader not interested in these proofs can go directly to Section 3.3.4.

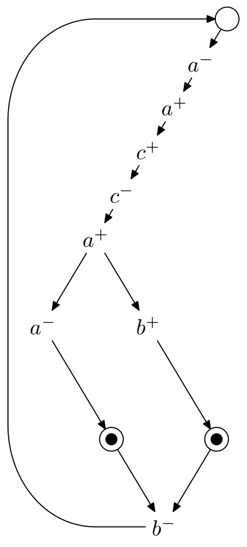

Figure 10. The STG S_R

We start by proving the second point:

Proposition 3.2. S_R is a very well formed free-choice STG.

Proof:
Notice that S_R is live, bounded, cyclic, and balanced if and only if $S \setminus \overline{N}$ is. Liveness and boundedness of $S \setminus \overline{N}$ is proved in Proposition 7.8 of [9]. According to Theorem 8.11 of the same reference, cyclicity requires to prove that all so-called traps of $S \setminus \overline{N}$ are marked. This follows easily from the fact that all traps of S are marked, and the fact that the initial marking of $S \setminus \overline{N}$ does not enable any internal transition of $\overline{N}$. Balancedness of $S \setminus \overline{N}$ follows from the fact that, due to the definition of CP-subnet, every T-invariant of $S \setminus \overline{N}$ is also a T-invariant of S.

It remains to show non-autoconcurrency of transitions of S_R. Notice first that if two transitions of S cannot be concurrently enabled, then they cannot be concurrently enabled in in $S \setminus \overline{N}$ either, since all the behaviors of $S \setminus \overline{N}$ are also possible in S. Therefore, $S \setminus \overline{N}$ is non-autoconcurrent. Furthermore, it is easy to see that no two transitions introduced as witnesses can be autoconcurrent. If they were, then, since they lie in a path starting at some entry place of $\overline{N}$, and in this path all transitions have one single input place, some reachable marking of S would put at least two tokens on the entry places of $\overline{N}$, and so the entry transition of $\overline{N}$ could be concurrently fired with itself, contradicting the non-autoconcurrency of S. Finally, we have to show that a witness transition t labelled by a^+ or a^- cannot be concurrently enabled with any a-labelled transition of $S \setminus \overline{N}$. For this, assume that t can be concurrently enabled with some transition u of $S \setminus \overline{N}$. Then, by the way witnesses are inserted, there is a reachable marking that enables u and puts a token on each place of P_e. From this marking the flushing sequence of $\overline{N}$ can be executed, and since $\overline{N}$ contains at least one a-labelled transition, say v, we have that t and v can be concurrently enabled. This contradicts the non-autoconcurrency of S. $\square$

Let us now consider the first point, i.e., to show that alternation holds for S (and for all signals) if and only if it holds for both S_{CP} and S_R. When introducing the procedure for the case of state machine STGs we spoke of the 'mirror sequences' of the flushing sequence of a handle. Recall that the goal of introducing witnesses was to ensure that the flushing sequence and its mirror image are 'equivalent', meaning that they have the same balance, contain the same signals, and, for each signal a, their projections on a start by the same letter.

In the free-choice case even the existence of 'mirror sequences' is nontrivial. The next proposition shows that they indeed exist, and that they satisfy the required properties.(Loosely speaking, proving the existence of mirror sequences amounts to showing that if we can reach L from M in S by executing a flushing sequence of $\overline{N}$, then we can also reach L from M *without using transitions of* $\overline{N}$.)

Proposition 3.3. Let S be a very well formed free-choice STG, let $\overline{N}$ be a CP-subnet of S, and let $M \xrightarrow{\sigma} L$ be a flushing sequence of $\overline{N}$ in S.

(1) S_R has an occurrence sequence $M' \xrightarrow{\tau} L'$, where M' and L' are the projections of the markings M and L onto the places of S_R.

(2) $bal(\sigma) = bal(\tau)$.

(3) Let $\sigma|_a$ and $\tau|_a$ be the projections of σ and τ onto a^+, a^-. If σ and τ are alternating and $\sigma|_a$ is nonempty, then $\sigma|_a$ and $\tau|_a$ start and end with the same label.

Proof:
(1) By Lemma 3.1, M and L coincide on the internal places of $\overline{N}$, and every transition of $\overline{N}$ occurs exactly once in τ. So we can add a new transition t to $N \setminus \overline{N}$ which removes one token from the entry place of $\overline{N}$, and adds one token to each of its exit places (the places having no output transition in $\overline{N}$). We then have $L' \xrightarrow{t} K'$.

Let $\mathbf{C}$ and $\mathbf{C_t}$ be the incidence matrices of $N \setminus \overline{N}$, and of the result of adding t to $N \setminus \overline{N}$, respectively. By Theorem 6.7 of [10] (see also Theorem 7.13 of [9]), there is a vector $X \geq \mathbf{0}$ such that $\mathbf{C} \cdot X = \mathbf{C_t} \cdot \vec{t}$. Since $L' \xrightarrow{t} K'$, we have $L' = K' + \mathbf{C_t} \cdot \vec{t}$, and so $K' = L' + \mathbf{C} \cdot X$. By Theorem 9.6 and 9.7 of [9], K' is reachable from L' in $S \setminus \overline{N}$, and we are done.

(2) By cyclicity, there is a sequence $L \xrightarrow{\upsilon} M$. So the reachability graph of S contains the circuits $M \xrightarrow{\sigma} L \xrightarrow{\upsilon} M$ and $M \xrightarrow{\tau} L \xrightarrow{\upsilon} M$. By balancedness, we have $bal(\sigma\upsilon) = bal(\tau\upsilon) = \mathbf{0}$, and so $bal(\sigma) = bal(\tau)$.

(3) Consider two cases:

- $bal(\sigma) = \mathbf{0}$.
  Since σ is alternating, we can assume without loss of generality that $\sigma|_a = a^+\alpha a^-$ for some α. Since S_R contains a witness for signal a, the projection $\tau|_a$ also starts with a^+. Now we have: τ is alternating, $\tau|_a$ starts with a^+, and $bal(\tau) = bal(\sigma) = \mathbf{0}$. It follows that $\tau|_a$ ends with a^-.

- $bal(\sigma) \neq \mathbf{0}$.
  Since σ is alternating, we can assume without loss of generality that $\sigma|_a = a^+\alpha a^+$ for some α. Since τ is alternating and $bal(\tau) = bal(\sigma)$, we have $\tau|_a = a^+\beta a^+$ for some β.

$\square$

The next lemma allows us to obtain a canonical form for the non-alternating sequences of S (if they exist).

Lemma 3.3. Let S be a very well formed free-choice STG with initial marking M_0, and let $\overline{N}$ be a CP-subnet of S. If S is non-alternating, then it has a non-alternating sequence of the form

$$M_0 \xrightarrow{\sigma_0} L_0 \xrightarrow{\tau_0} M_1 \ldots M_n \xrightarrow{\sigma_n} L_n \xrightarrow{\tau_n} M_{n+1}$$

such that:

- the σ_i sequences only contain transitions of S_R,
- the τ_i sequences only contain transitions of S_{CP}, and
- $L_0 \xrightarrow{\tau_0} M_1, \ldots, L_n \xrightarrow{\tau_n} M_{n+1}$ are flushing sequences of $\overline{N}$ in S.

Proof:
Assume that S is non-alternating. Then it has a non-alternating occurrence sequence $M_0 \xrightarrow{\sigma} M$.

Let t and u be an arbitrary internal transition of $\overline{N}$ and a transition of $N \setminus \overline{N}$, respectively. By the defi nition of CP-subnet, t and u have a disjoint preset. So for every occurrence sequence $L \xrightarrow{\tau t u \tau'} L'$, we have that $L \xrightarrow{\tau u t \tau'} L'$ is also an occurrence sequence. Moreover, since S is non-autoconcurrent, $L \xrightarrow{\tau t u \tau'} L'$ is alternating if and only if $L \xrightarrow{\tau u t \tau'} L'$ is alternating.

By exhaustively swapping internal transitions of $\overline{N}$ and transitions of $N \setminus \overline{N}$, we reorganize $M_0 \xrightarrow{\sigma} M'$ into an occurrence sequence $M_0 \xrightarrow{\sigma_0} L_0 \xrightarrow{\tau_0} M_1 \ldots M_n \xrightarrow{\sigma_n} L_n \xrightarrow{\tau_n} M_{n+1}$, where all $L_i \xrightarrow{\tau_i} M_{i+1}$ but possibly $L_n \xrightarrow{\tau_n} M_{n+1}$ are flushing sequences. If $L_n \xrightarrow{\tau_n} M_{n+1}$ is not a flushing sequence, we extend it to one by letting internal transitions of M_n occur as long as possible. □

We are now ready to prove the main result:

Theorem 3.4. Let S be a very well formed free-choice STG, and let $\overline{N}$ be a CP-subnet of S. S is alternating if and only if S_{CP} and S_R are alternating.

Proof:
(⇒): Easy.
(⇐): Assume that S is non-alternating. Let $M_0 \xrightarrow{\sigma_0} L_0 \xrightarrow{\tau_0} M_1 \ldots M_n \xrightarrow{\sigma_n} L_n \xrightarrow{\tau_n} M_{n+1}$ be the non-alternating sequence with respect to some signal a, whose existence is guaranteed by Lemma 3.3. If some $L_i \xrightarrow{\tau_i} M_{i+1}$ is non-alternating, then S_{CP} is non-alternating, and we are done. So assume that all τ_i's are alternating sequences. Let M_i', L_i' be the projections of M_i, L_i onto $N \setminus \overline{N}$. By repeated applications of Proposition 3.3, there exist sequences $\upsilon_0, \ldots \upsilon_{n-1}$ such that

$$M_0' \xrightarrow{\sigma_0} L_0' \xrightarrow{\upsilon_0} M_1' \ldots M_n' \xrightarrow{\sigma_n} L_n' \xrightarrow{\upsilon_n} M_{n+1}'$$

Observe that the projection of all the M_i, L_i onto the non-entry places of $\overline{N}$ is equal to $\overline{M}$, since they are all initial or fi nal markings of a flushing sequence (Lemma 3.1). Therefore,

$$M_0 \xrightarrow{\sigma_0} L_0 \xrightarrow{\upsilon_0} M_1 \ldots M_n \xrightarrow{\sigma_n} L_n \xrightarrow{\upsilon_n} M_{n+1}$$

is an occurrence sequence of S.

Consider two cases:
(1) Some σ_i or υ_i is non-alternating.
Then $S \setminus \overline{N}$ is non-alternating, and so S_R is non-alternating.
(2) All σ_i and υ_i are alternating.
We show that in this case S_R is non-alternating. Assume without loss of generality that $(\sigma_0\tau_0 \ldots \sigma_n\tau_n|_)a$ contains the word a^+a^+. Since all σ_i and τ_i are alternating, none of them contain a^+a^+. Therefore, the first and the second a^+ must be the last and the first letter, respectively, of two different elements of $\{\sigma_0, \tau_0, \ldots, \sigma_n, \tau_n\}$. There are four possible cases, of which we consider only one, the other three being similar:

• $\sigma_i|_a$ ends with a^+, $\tau_{i+k}|_a$ starts with a^+, and $\tau_i|_a\sigma_{i+1}|_a \ldots \sigma_{i+k}|_a$ is the empty word.
By Proposition 3.3(3), $\upsilon_{i+k}|_a$ also starts with a^+. By Proposition 3.3(2), the word $\upsilon_i|_a \ldots \upsilon_{i+k-1}|_a$ has balance **0**. Since the $\upsilon_i's$ are alternating, this word is either empty, starts with a^+, or ends with a^+. In all cases, $\sigma_i|_a\upsilon_i|_a \ldots \upsilon_{i+k}|_a$ contains the word a^+a^+. □

3.3.4. The base case

Theorem 3.4 and Proposition 3.2 allow to iteratively remove CP-subnets from S until either consistency is disproved, or no CP-subnets can be found. It is easy to show that one of the two will eventually be the case: every time a CP-subnet is removed, the sum over all places p of $|p^\bullet| - 1$ strictly decreases. If this sum becomes 0, then every place has exactly one output transition, and so no CP-subnets can exist.

It is shown in [9] (Proposition 7.11) that if no CP-subnets exist, then each place has exactly one output *and* exactly one input transition. Such nets are called T-nets or marked graphs, and we call the corresponding STGs *marked graph STGs*.

The rest of the section contains only folklore results, but we include them for the sake of completeness. In order to check consistency of very well formed marked graph STGs, we reuse a result that we proved for CP-subnets. It can be extended without effort to all non-autoconcurrent marked graph STGs.

Lemma 3.4. Let S be a non-autoconcurrent marked graph STG, and let $M \xrightarrow{\sigma}$ and $M \xrightarrow{\tau}$ be two occurrence sequences of S. For each signal a, $\ell(\sigma)|_a$ is a prefix of $\ell(\tau)|_a$, or vice versa.

Proof:
The proof of Lemma 3.1 works with minor modifications. We construct a sequence υ such that $\vec{\upsilon} = \max\{\vec{\sigma}, \vec{\tau}\}$, and show that if the desired property does not hold then S is not autoconcurrent. □

It is well known that well formed marked graphs have the following property: there exists an occurrence sequence $M_0 \xrightarrow{\sigma} M_0$ in which all transitions occur exactly once. This sequence can obviously be iterated an arbitrary number of times. We have the following result:

Theorem 3.5. Let S be a very well formed marked graph STG with initial marking M_0. Let $M_0 \xrightarrow{\sigma} M_0$ be the sequence in which each transition occurs exactly once. S is consistent if and only if σ is alternating.

Proof:
The only if direction is trivial. For the if direction, assume S is not consistent. Since S is very well

formed, it is live and cyclic. By Lemma 2.1 there is a non-alternating sequence $M_0 \xrightarrow{\tau}$. Choose a number n such that σ^n is longer than τ. By Lemma 3.4, $M_0 \xrightarrow{\sigma^n}$ is non-alternating. Since S is balanced and $\vec{\sigma}$ is a T-invariant, we have $bal(\vec{\sigma}) = \mathbf{0}$. But then σ must be non-alternating. □

4. The algorithm

In this section we present the complete algorithm for checking consistency.
Input: a well formed free-choice STG S.
Output: 'consistent' or 'not consistent'.

(1) Check if S is non-autoconcurrent. If it is autoconcurrent, return 'not consistent', otherwise go to step 2.

(2) Check if S is balanced. If not, return 'non consistent', otherwise go to step 3.

(3) If S is a marked graph STG, construct a sequence $M_0 \xrightarrow{\sigma} M_0$ in which every transition occurs exactly once; if σ is alternating for every signal then return 'consistent', otherwise return 'not consistent'.
If S is not a marked graph, go to step 4.

(4) Select a CP-subnet $\overline{N}$ of S, and let its transitions occur as long as possible (each transition can occur at most once). Call this new STG (new because of the different initial marking) S'.

(5) Consider $\overline{N}$ with the following initial marking: one token on each entry place, as many tokens on each internal place as in the initial marking of the STG S', and no tokens on exit places. Execute a flushing sequence in which each transition occurs exactly once. Check if all signals alternate. If not, return 'not consistent', otherwise go to step 6.

(6) Remove from S' all internal places and transitions of $\overline{N}$. Let $S := S'_R$ as defined in Section 3.3.2 (inserting witnesses where necessary), and go to step 3.

We show that the algorithm runs in $O((|P| + |T|)^3)$ time:

- The algorithm of [14] for Step (1) takes $O(|P| \cdot (|P| + |T|)^2)$ time.

- Step (2) requires to compute a basis of the solutions of the system $\mathbf{N} \cdot X = \mathbf{0}$, where $\mathbf{N}$ has dimension $|P| \times |T|$. This can be done in time $O(\mathcal{M}(\max\{|P|, |T|\}))$, where $\mathcal{M}(n)$ denotes the complexity of matrix multiplication, and so certainly in time $O((|P| + |T|)^3)$.

- The loop of the divide-and-conquer procedure can run for at most $|T|$ iterations, since each iteration reduces the number of output transitions of one place by 1. Moreover, at each step the STG S'_R has at most as many nodes as S (the inserted path of witnesses contains at most as many nodes as the removed CP-subnet).

- Steps (3) and (5) take $O((|P| + |T|)^2)$ time, even if we assume that it takes linear time to find an enabled transition.

- Step (4) takes $O((|P| + |T|)^2)$ time: there are at most $|T|$ candidates to be the entry transition of a CP-subnet, and we can check in $O(|P| + |T|)$ time for each of them if they indeed are (using Tarjan's algorithm for checking strong connectedness).

- Step (6) takes $O(|P| + |T|)$ time.

So the execution of all the iterations of the loop's body take together $O(|T| \cdot (|P| + |T|)^2)$ time, and the whole algorithm runs in $O((|P| + |T|)^3)$ time.

5. Conclusions

We have presented the first polynomial algorithm for checking consistency of well formed free-choice STGs. The correctness proof uses many results from Petri net theory. Almost all of the main theorems of the monograph [9] are directly or indirectly used. There could be a much simpler proof, although experience shows that results about free-choice nets often need long arguments.

Together with the techniques of [3] and [18], our algorithm can be used to produce free-choice STGs with consistent and complete encodings without having to construct their state graph, which could be exponentially larger than the STG itself.

We suspect that it is possible to get rid of the balancedness check at the price of a more complicated divide-and-conquer procedure. Exploring this is left for future research.

The consistency problem is an instance of the more general problem of determining *synchronic distances* between sets of transitions of a Petri net (how often can transitions in one set occur without the others occurring?). These problems appear also in other applications of free-choice nets in the are of workflow processes [1]. We think that our results will also be useful there.

6. Acknowledgements

Many thanks to Jordi Cortadella and Alex Yakovlev for many helpful comments, pointers to the literature, and discussions. The author is very indebted to the anonymous referees for their careful reading and very helpful comments.

References

[1] van der Aalst, W.: The Application of Petri Nets to Workflow Management, *The Journal of Circuits, Systems and Computers*, **8**(1), 1998, 21–66.

[2] van Berkel, C., Josephs, M., Nowick, S.: Scanning the Technology: Applications of Asynchronous Circuits, *Proceedings of the IEEE*, **87**(2), 1999, 234–242.

[3] Carmona, J., Cortadella, J., Pastor, E.: A structural encoding technique for the synthesis of asynchronous circuits, *Proc. Int. Conf. on Application of Concurrency Theory to System Design*, IEEE Computer Society, 2001.

[4] Chu, T.-A.: On the models for designing VLSI asynchronous digital systems, *Integration: the VLSI journal*, **4**, 1986, 99–113.

[5] Chu, T.-A.: *Synthesis of Self-Timed VLSI Circuits from Graph-theoretic Specifications*, Ph.D. Thesis, MIT, 1987.

[6] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers, *IEICE Trans. on Information and Systems*, **E80-D**(3), 1997, 315–325.

[7] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: *Logic synthesis of asynchronous controllers and interfaces*, Springer-Verlag, 2002.

[8] Cortés, L.: Private communication, 2002.

[9] Desel, J., Esparza, J.: *Free Choice Petri Nets*, vol. 40 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, 1995.

[10] Esparza, J.: Reduction and Synthesis of Live and Bounded Free Choice Petri Nets, *Information and Computation*, **114**(1), 1994, 50–87.

[11] Esparza, J.: Decidability and complexity of Petri net problems - an introduction, in: *Lectures on Petri Nets I: Basic Models. Advances in Petri Nets*, vol. 1491 of *Lecture Notes in Computer Science*, Springer-Verlag, 1998, 374–428.

[12] Kondratyev, A., Cortadella, J., Kishinevsky, M., Pastor, E., Roig, O., Yakovlev, A.: Checking Signal Transition Graph implementability by Symbolic BDD Traversal, *Proc. European Design and Test Conference*, 1995.

[13] Kovalyov, A.: Concurrency Relations and the Safety Problem for Petri Nets, in: *Proc. 13th Int. Conf. on Application and Theory of Petri Nets*, vol. 616 of *Lecture Notes in Computer Science*, 1992, 299–309.

[14] Kovalyov, A., Esparza, J.: A Polynomial Algorithm to Compute the Concurrency Relation of Free-choice Signal Transition Graphs, *Proc. Int. Workshop on Discrete Event Systems*, IEE Society, 1996.

[15] Landweber, L., Robertson, E.: Properties of Conflict-Free and Persistent Petri Nets, *Journal of the ACM*, **25**(3), 1978, 352–364.

[16] Lavagno, L., Moon, C., Brayton, R., Sangiovanni-Vincentelli, A.: Solving the state assignment problem for signal transition graphs, *Proc. 29th IEEE/ACM Design Automation Conference*, IEEE Computer Society Press, 1992.

[17] Lipton, R.: *The Reachability Problem Requires Exponential Space*, Technical Report 62, Yale University, 1976.

[18] Pastor, E., Cortadella, J., Kondratyev, A., Roig, O.: Structural Methods for the Synthesis of Speed Independent Circuits, *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, **17**(11), 1998, 1108–1129.

[19] Reisig, W.: *Petri Nets: An Introduction*, vol. 4 of *EATCS Monographs in Computer Science*, Springer-Verlag, 1985.

[20] Rosenblum, L., Yakovlev, A.: Signal Graphs: from self-timed to timed ones, *Proc. Int. Workshop on Timed Petri nets*, IEEE Computer Society, 1985.

[21] Vanbekbergen, P., Goosens, G., Catthoor, F., Man, H. D.: Optimized Synthesis of Asynchronous Control Circuits from Graph-Theoretic Specifications, *IEEE Trans. on Computer Aided Design*, **11**(11), 1992, 1426–1438.

[22] Vanbekbergen, P., Lin, B., Goosens, G., Man, H. D.: A generalized state assignment theory for transformations on signal transition graphs, *Proc. Int. Conf. on Computer Aided Design*, 1992.

7. Appendix: Algorithms for the concurrency relation

The following algorithm closely follows the definition of $\|^s$. R contains the pairs that are known to belong to $\|^s$. E contains the pairs of R that have not been explored yet in order to check if they can generate new pairs.

```
Input:    A live Petri net $(N, M_0)$, where $N = (P, T, F)$.
Output:   The structural concurrency relation $R \subseteq X \times X$, where $X = P \cup T$.
begin
  $R := \{(p, p') \mid M_0 \geq M_p + M_{p'}\} \cup \ (\bigcup_{t \in T}(t^\bullet \times t^\bullet) \setminus id_P)$;
  $E := R$;
  while $E \neq \emptyset$ do
    select $(x, p) \in E$; $E := E \setminus \{(x, p)\}$;
    for every $t \in p^\bullet$ do
      if $\{x\} \times {}^\bullet t \subseteq R$ then
        $Aux := (\{x\} \times t^\bullet) \cup (t^\bullet \times \{x\}) \cup \{(x, t), (t, x)\}$;
        $E := E \cup ((Aux \cap (X \times P)) \setminus R)$;
        $R := R \cup Aux$
end
```

If a subset $R \subseteq X \times X$ is encoded as a two dimensional bitarray, the algorithm needs $O(|X|^2)$ space and $O(|P|^2 \cdot |T| \cdot |X|)$ time.

It is possible to give a faster algorithm for free-choice systems, because they satisfy the following property: ${}^\bullet t_1 = {}^\bullet t_2$ for every two output transitions t_1, t_2 of a place p. In this case, the condition of the **if** instruction holds either for all or for none of the transitions $t \in p^\bullet$ which are examined during the **for** loop. So, instead of checking the condition for each transition of $p^\bullet$, we just check it for one of them. If the condition holds, we add to R the whole set $\{x\} \times (p^\bullet)^\bullet$ (this is the purpose of the set A). We get the following algorithm:

```
Input:    A live free-choice system $(N, M_0)$, where $N = (P, T, F)$.
Output:   The structural concurrency relation $R \subseteq X \times X$, where $X = P \cup T$.
begin
  $R := \{(p, p') \mid M_0 \geq M_p + M_{p'}\} \cup \ (\bigcup_{t \in T}(t^\bullet \times t^\bullet) \setminus id_P)$;
  $A := \{(p, p') \mid p' \in (p^\bullet)^\bullet\}$;
  $E := R$;
  while $E \neq \emptyset$ do
    select $(x, p) \in E$; $E := E \setminus \{(x, p)\}$;
    select $t \in p^\bullet$;
    if $\{x\} \times {}^\bullet t \subseteq R$ then
      $Aux := \{(x, p'), (p', x) \mid (p, p') \in A\} \cup \{(x, u), (u, x) \mid u \in p^\bullet\}$;
      $E := E \cup ((Aux \cap (X \times P)) \setminus R)$;
      $R := R \cup Aux$
end
```

This algorithm runs in $O(|X|^2)$ space and $O(|P| \cdot |X|^2)$ time.