On the Complexity of Consistency and Complete State Coding for Signal Transition Graphs

Javier Esparza
Institute for Formal Methods in Computer Science
Univ. Stuttgart, Germany
esparza@informatik.uni-stuttgart.de

Petr Jančar*
Center of Applied Cybernetics
Dept of CS, TU Ostrava, Czechia
Petr.Jancar@vsb.cz

Alexander Miller
Institute for Formal Methods in Computer Science
Univ. Stuttgart, Germany
Alexander.Miller@informatik.uni-stuttgart.de

Abstract

Signal Transition Graphs (STGs) are a popular formalism for the specification of asynchronous circuits. A necessary condition for the implementability of an STG is the existence of a consistent and complete state encoding. For an important subclass of STGs, the marked graph STGs, we show that checking consistency is polynomial, but checking the existence of a complete state coding is co-NP-complete. In fact, co-NP-completeness already holds for acyclic and 1-bounded marked graph STGs and for live and 1-bounded marked graph STGs. We add some relevant results for free-choice, bounded, and general STGs.

1. Introduction

Signal transition graphs (STGs) are a popular formalism for specifying asynchronous circuits [3, 10]. They are Petri nets in which the firing of a transition is interpreted as rising or falling of a signal in the circuit. Not every STG can be implemented as a physical circuit. A central question related to implementability of an STG is whether it admits a so-called *consistent* and *complete state coding*. Most papers in the literature consider only the completeness part, assuming that the STG is already consistent, and call the existence of a complete state coding the *CSC property*. This property, and the stronger unique state coding property (*USC property* for short) have been studied in many papers (see e.g. [1, 7, 8, 9, 11, 13, 14]).

*This author is supported by the Czech Ministry of Education, Grant No. 1M0567

In this paper we reason about the computational complexity of deciding if a given STG has a consistent and complete state coding, viewing the consistency problem separately. We obtain new results for STGs whose underlying nets are marked graphs and free-choice nets; for completeness, we also sketch some straightforward results for STGs whose underlying nets are more general—bounded or even arbitrary.

We first explore the consistency problem for marked graph STGs. In [6] a polynomial algorithm was given to check consistency of live, bounded, and cyclic free-choice STGs, which include live and bounded marked graph STGs as a subclass. Here we show that consistency is polynomial for arbitrary marked graph STGs by means of a new algorithm based on linear programming.

A natural question is whether these polynomiality results also hold for the CSC or USC problems (i.e., the problems of checking the CSC or USC properties), at least for the class of live and 1-bounded marked graph STGs. Our main result shows that both problems are co-NP-complete, and so that polynomial algorithms are unlikely. This result explains why the algorithms of [1, 7, 8, 9, 11, 13, 14] have exponential runtime or can only decide some necessary or sufficient conditions for the CSC or USC properties to hold. These algorithms are discussed in detail in the final section.

Our co-NP-completeness result is rather robust. We prove that the CSC and USC problems remain co-NP-hard for 1-bounded and acyclic marked graph STGs, and that they remain in co-NP for arbitrary marked graph STGs and for live and bounded free-choice STGs.

Moving to more general classes, we show that the consistency, CSC and USC problems are PSPACE-complete for 1-bounded STGs, and that the consistency problem remains

PSPACE-hard in the free-choice case. Finally, we clarify the relation between the consistency, USC and CSC problems for general STGs, and the fireability and reachability problems for general Petri nets.

The paper is structured as follows. Section 2 presents basic definitions and a characterization of consistency. Section 3 presents the results about marked-graph STGs; it is the core of the paper. Section 4 deals with free-choice and Section 5 with general STGs. Section 6 contains conclusions and discusses related work.

Remark. The full version of this paper, containing all proofs, is available at
http://www.informatik.uni-stuttgart.de/fmi/szs/publications

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*; we also note that a net can be viewed as a (bipartite) graph. Places are graphically represented as circles; transitions are usually drawn like boxes, but we use just their labels in the figures. 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 set of 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 $M(p) \geq 1$ for each $p \in {}^\bullet t$. If t is enabled at M, then it can *fire* (or *occur*) and its firing (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. $M \xrightarrow{t}$ denotes that t is enabled at M, and $M \xrightarrow{t} M'$ moreover denotes that firing t leads to M'.

The notation $M \xrightarrow{\sigma}$, $M \xrightarrow{\sigma} M'$ is extended to finite sequences $\sigma \in T^*$ in the natural way. When $M \xrightarrow{\sigma} M'$, for $\sigma = t_1 t_2 \cdots t_n$, we sometimes speak about an *occurrence sequence from M to M'*, meaning the sequence

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

By the *Parikh vector* of $\sigma \in T^*$, denoted by $\vec{\sigma}$ or $P(\sigma)$, we mean the mapping $T \to \mathbb{N}$ such that $\vec{\sigma}(t)$ is the number of occurrences of t in σ.

The *incidence matrix* of N is the matrix $C_N \colon P \times T \to \{-1, 0, +1\}$ given by $C_N(p, t) = F(t, p) - F(p, t)$. We note that if $M \xrightarrow{\sigma} M'$ then $M + C_N \cdot \vec{\sigma} = M'$. (We naturally identify the mapping $\vec{\sigma}$ with a (nonnegative integer) vector; that's why we use the term 'Parikh vector'.)

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 marking M is called *reachable* if there exists an occurrence sequence from M_0 to M; we also denote this by $M_0 \to^* M$. We call

$$M_0 + C_N \cdot X \geq 0$$

the *marking inequation*. We note that $M_0 \xrightarrow{\sigma} M$ implies $M_0 + C_N \cdot \vec{\sigma} = M$; $\vec{\sigma}$ is thus a (nonnegative integer) solution of the marking inequation.

A marking M of a net N is *n-bounded* if $M(p) \leq n$ for every place p. A Petri net (N, M_0) is n-bounded if all its reachable markings are n-bounded.

A transition t is *fireable* in (N, M_0) if there is σ such that $M_0 \xrightarrow{\sigma} M$ and $M \xrightarrow{t}$. A Petri net (N, M_0) is *live* if each transition t is fireable in (N, M) for each M reachable from M_0. A transition is *dead* at a marking M if t is not fireable in (N, M).

A net N is called a *marked graph* if every place has at most one input and at most one output transition. $N = (P, T, F)$ is a *free-choice* net if: 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 net, 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.

Signal transition graphs. Let $A = \{a_1, \ldots, a_n\}$ be a set (alphabet) of *signals* partitioned into *input* and *output* signals. Rising and falling of a signal a is denoted by a^+ and a^-, respectively. (In some proofs we also use the notation $+a$ and $-a$, which is more convenient for using sub- and superscripts.) We call an element of $\mathcal{L} = A \times \{+, -\}$ a *label*. 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 *labelling function* that assigns to each transition of N a label in $\mathcal{L}$.

A signal transition graph is a specification of the behaviour of the circuit under some assumptions on the environment. An STG S is implementable if there exists a *state coding mapping* λ (we also use the term *binary encoding*) that associates to each reachable marking M a vector of *signal values* $\lambda(M) \in \{0, 1\}^n$ satisfying the following two properties:

(1) *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 $\lambda(L)$.

(2) *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 define the *consistency problem* as the problem of deciding if a given STG is consistent, i.e., if it admits a binary encoding λ satisfying (1). The Complete State Coding problem, *CSC problem* for short, is the problem to decide if a given STG (usually already assumed consistent) has the *CSC property*, i.e., admits a binary encoding satisfying (1) and (2). A stricter version is the *USC problem* (unique state coding) where we ask if a given STG has the *USC property*, i.e., admits an *injective* binary encoding λ satisfying (1) (thus $\lambda(M) \neq \lambda(L)$ for any two different reachable M, L).

STGs naturally inherit many notions from their underlying (Petri) nets. We already used this when speaking about 'enabling a label', e.g. $M \xrightarrow{a^+}$ (meaning that M enables a transition with label a^+). Thus we will freely speak about n-bounded, live, marked graph, or free-choice STGs, etc. We can also use notions like a is dead at M (meaning that each transition with label a^+ or a^- is dead at M).

We also freely use notation like $M \xrightarrow{u} M'$ for sequences of labels (meaning that there is a transition sequence $\sigma = t_1 t_2 \cdots t_m$ such that $M \xrightarrow{\sigma} M'$ and $u = \ell(t_1)\ell(t_2)\cdots\ell(t_m)$). We can occasionally even mix, and consider u as a sequence of transitions and labels, when this should not cause confusion. We also use expressions like u is a-free, meaning that there is no a^+ nor a^- in u; and if u contains transitions, we mean that those transitions do not have labels a^+, a^-. Recall that $P(u)$ denotes the Parikh vector of u; We denote by $P(u)(a^+)$ the number of transitions with label a^+ in u.

Finally we note that since the circuit implementation of an 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, most of them are even 1-bounded; but in principle unbounded STGs can make sense.

We finish the section by a characterization of consistency, i.e., we look in more detail on when an STG is inconsistent. (The proof is straightforward, and can be found in the full version.)

Proposition 2.1 *An STG $S = (N, M_0, \ell)$ is inconsistent (i.e., it admits no consistent binary encoding) iff there is*

a pair (M, a)

where $M_0 \rightarrow^* M$ and a is a signal

such that one of the following conditions holds:

(1) M enables ua^+ and va^- for some a-free sequences u, v,

(2) M enables a^+ua^+ or a^-ua^- for some a-free sequence u,

(3) M is reachable by w_1a^+u and by w_2a^-v for some a-free sequences u, v (and some w_1, w_2).

3. Marked graphs

In this section we show that consistency can be decided in polynomial time for all marked graph STGs and that both the CSC problem and the USC problem are co-NP-complete for them, even in the case of 1-bounded acyclic marked graphs and in the case of live 1-bounded marked graphs.

3.1. Consistency

In [6] it is shown that consistency of live, bounded, and cyclic free-choice STGs can be decided in polynomial time. (A Petri net is *cyclic* if the initial marking is reachable from every reachable marking, i.e., if it is always possible to return to the initial marking). Since live and bounded marked graphs are always cyclic (see for instance [4]), and marked graphs are a special case of free-choice nets, [6] provides a polynomial algorithm deciding consistency of live and bounded marked graph STGs. We now show a polynomial algorithm for all marked graph STGs.

We start by recalling some simple properties of marked graphs and derive a simpler variant of Proposition 2.1, valid for marked graphs. (Proofs are in the full version.) One such property is that if M enables a sequence with n occurrences of t and $M \xrightarrow{t'} M'$ for $t' \neq t$ then M' enables a sequence with n occurrences of t as well; if $t' = t$ then M' enables a sequence with $n-1$ occurrences of t.

By $P(u)(t)$ we denote the number of occurrences of t in a transition sequence u (P stands for the Parikh vector).

Claim 3.1 *Let M be a marking of a marked graph. If $M \xrightarrow{u} M_1$ and $M \xrightarrow{v} M_2$ then $M \xrightarrow{w} M'$ for some w and M' such that*

$$\forall t : P(w)(t) = max\{\, P(u)(t), P(v)(t) \,\}.$$

Moreover, if $M_1 \xrightarrow{t}$ and $P(v)(t) \leq P(u)(t)$ then $M' \xrightarrow{t}$.

Proposition 3.1 *A marked graph STG $S = (N, M_0, \ell)$ is inconsistent iff one of the following conditions holds:*

(1') *there is a reachable M ($M_0 \rightarrow^* M$) such that $M \xrightarrow{a^+}$ and $M \xrightarrow{a^-}$ for some signal a,*

(2') *there is a reachable M such that $M \xrightarrow{a^+ u a^+}$ or $M \xrightarrow{a^- u a^-}$ for some signal a and some a-free sequence u.*

It is now sufficient to show that conditions (1'), (2') of Proposition 3.1 can be checked in polynomial time.

To this aim, we recall further useful observations about marked graphs. We note that, given a marked graph STG $S = (N, M_0, \ell)$, we can check in polynomial time if there is a circuit of N which is not marked at M_0 (i.e., its places have no tokens in M_0). The places of such a circuit can be safely removed, since no transition in the circuit can ever occur.

We call a marked graph (N, M_0) *normalized* if every circuit of N is marked at M_0.

Claim 3.2 *Let (N, M_0) be a normalized marked graph, and consider the inequation $M_0 + C_N \cdot X \geq 0$, where C_N is the incidence matrix of N. An integer vector $X_0 \geq 0$ is a solution of this inequation if and only if $M_0 \xrightarrow{\sigma}$ for a transition sequence σ whose Parikh vector is X_0. For any linear objective function $f(X)$, the optimal solution of the inequation (if it exists) is integer, and can be computed in polynomial time.*
Moreover, if $M_0 \xrightarrow{\sigma} M$ then $M_0 + C_N \cdot X_0 = M$.

Now we come to the polynomiality claims, which can be quickly established by using linear programming (which is a well-known polynomial problem). The complete proofs are in the full version.

Proposition 3.2 *For normalized marked graph STGs, checking (1') of Proposition 3.1 can be done in polynomial time.*

Proof: Follows easily from Claim 3.2. ■

Proposition 3.3 *For normalized marked graph STGs not satisfying (1'), checking (2') can be done in polynomial time.*

Proof: Let $S = (N, M_0, \ell)$ be a normalized marked graph STG which does not satisfy (1'); i.e., no reachable M can enable both a^+ and a^-. From this we can derive that (M_0, a) does not satisfy (1) of Proposition 2.1. Therefore, in every occurrence sequence containing occurrences of the signal a, the first occurrence of a always has the same sign. Which sign this is, $+$ or $-$, can be determined very efficiently, e.g. by firing any maximal transition sequence in which each transition of S occurs at most once (such a sequence contains all transitions that can ever be enabled).

Consider signal a, and assume we have found that a^+ is fireable as the first of a^+, a^-. (The case with a^- being the first is similar.)

Let us now solve the linear programming problems

$$\begin{array}{ll} \text{maximize} & f(X) \\ \text{subject to} & X \geq 0,\ M_0 + C_N \cdot X \geq 0 \end{array}$$

$$\begin{array}{ll} \text{minimize} & f(Y) \\ \text{subject to} & Y \geq 0,\ M_0 + C_N \cdot Y \geq 0 \end{array}$$

where

$$f(X) = \sum_{t \in \ell^{-1}(a^+)} X(t) - \sum_{t \in \ell^{-1}(a^-)} X(t)$$

If we find that it is NOT the case that both problems have optimal solutions X_{op}, Y_{op} with $f(X_{op}) = 1$ and $f(Y_{op}) = 0$ then we claim '(2') holds'.

To check (2'), we run the above procedure for each signal a separately, and claim that (2') holds when one signal gives rise to this claim, otherwise we claim that (2') does not hold. The overall time of this algorithm is surely polynomial. Its correctness follows from Claim 3.2 (a more detailed proof can be found in the full version.) ■

Theorem 3.1 *Consistency of marked graph STGs can be decided in polynomial time.*

Proof: The polynomial algorithm first normalizes the STG and then uses the algorithms guaranteed by Propositions 3.2 and 3.3 to check if one of the conditions (1'), (2') of Proposition 3.1 holds. ■

3.2. Complete state coding

In this subsection we show the announced co-NP-completeness results for the CSC problem and the USC problem on (consistent) marked graph STGs.

The next lemma is the main technical result of the paper. We say that an occurrence sequence is *balanced* if for every signal a the sequence contains the same number of occurrences of transitions labelled by a^+ and of transitions labelled by a^-.

Lemma 3.1 *The following problem is NP-complete:*
Instance: *a (consistent) STG $S = (N, M_0, \ell)$ such that (N, M_0) is a 1-bounded, acyclic marked graph.*
Question: *is there an occurrence sequence $M_0 \xrightarrow{\sigma} M_1 \xrightarrow{\tau} M_2$ of S such that τ is nonempty and balanced?*

Proof: Membership in NP is clear: In any net (N, M_0) which is 1-bounded and acyclic, each transition can appear at most once in any occurrence sequence. So a nondeterministic algorithm can just guess a sequence $\sigma\tau$ of pairwise distinct transitions and verify that it is performable from M_0 and that τ is nonempty and balanced.

The main point is NP-hardness, which we show by a reduction from CNF-SAT. Let φ be a boolean formula in conjunctive normal form

- with m clauses $c_1, \ldots, c_m$,
- and n variables $x_1, \ldots, x_n$.

(E.g., formula $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_4})$ has 2 clauses and 4 variables.)

Our aim is to show a polynomial construction of a certain STG $S_\varphi = (N, M_0, \ell)$, with (N, M_0) being a 1-bounded acyclic marked graph, so that φ is satisfiable iff S_φ admits $M_0 \xrightarrow{\sigma} M_1 \xrightarrow{\tau} M_2$ for some sequence σ and some nonempty balanced sequence τ.

The construction is based on the fact that there is a truth assignment

$$\mathcal{A} : \{x_1, x_2, \ldots, x_n\} \to \{0, 1\}$$

satisfying φ if and only if there is a *consistent choice of literals*, by which we mean a mapping

$$l : \{c_1, c_2, \ldots, c_m\} \to \{x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_n, \overline{x_n}\}$$

attaching to each clause c_i one of its literals, denoted $l(c_i)$, in such a way that $l(c_i) \neq \overline{l(c_j)}$ for all i, j (i.e., it is forbidden that one clause 'chooses' x while another clause 'chooses' $\overline{x}$).

We can easily observe that any consistent choice of literals l naturally provides a satisfying truth assignment $\mathcal{A}$ (which can be specified arbitrarily for variables not appearing in the range of l); and any satisfying truth assignment enables to define (maybe several) consistent choices of literals.

We now describe the STG S_φ, providing also informal comments which will ease the later correctness proof. Figure 1 shows the overall structure of S_φ.

We need a few remarks about the notation. We construct $S_\varphi = (N, M_0, \ell)$ where N is an acyclic marked graph. All the minimal elements with respect to the flow relation will be places, and precisely those places will be initially marked (i.e., each will carry one token). We say that there is *an arc from transition t_1 to transition t_2* when there is an (intermediate) place p (initially unmarked) and arcs $t_1 \to p$, $p \to t_2$. (This is, in fact, a usual convention which we also use for drawing marked graphs.)

Each symbol of Figure 1 (i.e., each $V_T^1, \ldots, C_m^P$) stands for an acyclic marked graph. The arrow $V_T^1 \to N_\$$ has

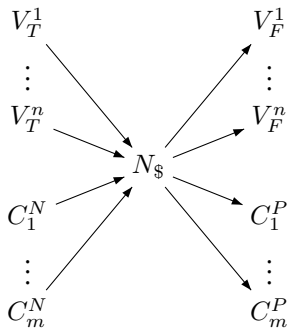

Figure 1. The overall structure of S_φ

the following meaning: V_T^1 has a transition t which is the unique maximal element in V_T^1 (w.r.t. the order induced by the flow relation), $N_\$$ has a transition u which is the unique minimal element in $N_\$$, and the (overall) net N contains an arc leading from t to u (with an intermediate place—using our convention). The meaning of the other arrows in the structure is analogous.

It will be clear (after we finish the construction) that any complete behaviour of S_φ can be divided into three phases:

I. first, all transitions in $V_T^1, \ldots, V_T^n, C_1^N, \ldots, C_m^N$ occur,

II. then all transitions of $N_\$$ follow,

III. and finally all transitions in $V_F^1, \ldots, V_F^n, C_1^P, \ldots, C_m^P$ occur.

The complete behaviours of S_φ differ only in the order in which transitions occur in the phases I and III. We proceed to describe the marked graphs corresponding to $N_\$, V_T^1, \ldots, V_T^n, C_1^N, \ldots, C_m^N$. Since we need to use both sub- and superscripts, we change the notation and write $+a$ and $-a$ instead of a^+ and a^-. The net $N_\$$, enabled after the whole phase I is finished, has one single (complete) behaviour, shown in Figure 2.

$$+\$\ -x^1\ -x^2\ \cdots\ -x^n\ -c_1\ -c_2\ \cdots\ -c_m\ -\$$$

Figure 2. (Linear) behaviour of $N_\$$

This means that the signal set of S_φ contains (among others):

- a signal c_i for every clause ($1 \leq i \leq m$);
- a signal x^j for every variable ($1 \leq j \leq n$);
- a (special) signal \$.

Signal \$ will not appear anywhere else but in $N_\$$. It will be the case that any nonempty balanced sequence must include all transitions, of $N_\$$, and so such a sequence will necessarily contain the whole phase II.

For the rest of the proof let bal denote any non-empty and balanced sequence such that $M_0 \xrightarrow{\sigma} M_1 \xrightarrow{bal} M_2$. In bal, each falling $-x^j$ ($1 \leq j \leq n$) must be compensated by a raising $+x^j$; the label $+x^j$ will appear just on the maximal (i.e., the last) transition of V_T^j (cf. Figure 3) and on the minimal (i.e., the first) transition of V_F^j (cf. Figure 4). So precisely one of the subnets V_T^j, V_F^j will contribute to bal. We interpret this as 'choosing' a truth assignment $\mathcal{A}$.

Similarly, each falling $-c_i$ ($1 \leq i \leq m$) must be compensated by a raising $+c_i$; the label $+c_i$ will appear just once in C_i^N and once in C_i^P, now 'almost' as the last transition and 'almost' as the first transition, respectively. Again, exactly one of the subnets C_i^N, C_i^P will contribute to bal.

Now we continue with the details of our construction. We extend the signal set used so far by

- a signal p_i^j for each pair i, j ($1 \leq i \leq m$, $1 \leq j \leq n$) such that clause c_i contains literal x_j (p stands for 'positive');
- a signal n_i^j for each pair i, j ($1 \leq i \leq m$, $1 \leq j \leq n$) such that clause c_i contains literal $\overline{x_j}$ (n stands for 'negative').

(As usual, we can assume that no clause c_i of formula φ contains a complementary pair of literals.)

Given j ($1 \leq j \leq n$), let $\{c_{i_1}, c_{i_2}, \ldots, c_{i_a}\}$ be the set of clauses containing (positive) literal x_j. The (sub)net V_T^j (representing setting x_j to 'true') is depicted in Figure 3. Thus V_T^j 'emits' labels $+p_{i_1}^j, +p_{i_2}^j, \ldots, +p_{i_a}^j$ in any order,

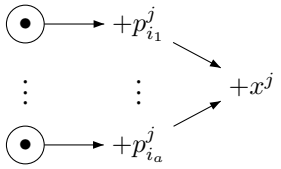

Figure 3. The net V_T^j

and then finishes by $+x^j$.

Now let $\{c_{k_1}, c_{k_2}, \ldots, c_{k_b}\}$ be the set of clauses containing (negative) literal $\overline{x_j}$. The (sub)net V_F^j (representing setting x_j to 'false') is depicted in Figure 4. Thus, after the label $-\$$ of $N_\$$ occurs, V_F^j 'emits' label $+x^j$ and then labels $-n_{k_1}^j, -n_{k_2}^j, \ldots, -n_{k_b}^j$ in any order.

We now define the subnets C_i^N, C_i^P. Recall that the sequence bal will contain either transitions of C_i^N or C_i^P, but not of both. This corresponds to 'choosing' either a positive

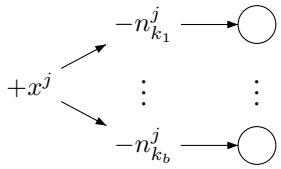

Figure 4. The net V_F^j

or a negative literal $l(c_i)$ from c_i. Which literal is chosen will depend on which transitions of the corresponding net occur in bal, and is explained later.

The nets C_i^N and C_i^P have no concurrency. They use additional 'parenthetical' signals. More precisely, we enhance the signal set by

- a signal $\square_i^j$ for each pair i, j, $1 \leq i \leq m$ and $1 \leq j \leq n_i$, where n_i is the number of negative literals in c_i;
- a signal $\triangle_i^j$ for each pair i, j, $1 \leq i \leq m$ and $1 \leq j \leq p_i$, where p_i is the number of positive literals in c_i.

Given i ($1 \leq i \leq m$), let $\{\overline{x_{j_1}}, \overline{x_{j_2}}, \ldots, \overline{x_{j_a}}\}$ be the set of negative literals of the clause c_i.

The (sub)net C_i^N has a (marked) place as the least element (w.r.t. the flow relation). And the only (complete) behaviour of C_i^N is the sequence of labels shown in Figure 5. The key observation is that if the label $+c_i$ of C_i^N belongs

$$+\square_i^1\ +n_i^{j_1}\ -n_i^{j_2}\ -\square_i^1\ +\square_i^2\ +n_i^{j_2}\ -n_i^{j_3}\ -\square_i^2\ \cdots$$

$$\cdots\ +\square_i^{a-1}\ +n_i^{j_{a-1}}\ -n_i^{j_a}\ -\square_i^{a-1}\ +\square_i^a\ +n_i^{j_a}\ +c_i\ -\square_i^a$$

Figure 5. (Linear) behaviour of C_i^N

to the balanced sequence bal, then bal must also contain $-\square_i^a$, and thus, by balancedness, also $+\square_i^a$. But then bal also contains $+n_i^{j_a}$, and so it must also contain $-n_i^{j_a}$. If we add the label $-n_i^{j_a}$ of C_i^N to bal, then we are forced to add $-\square_i^{a-1}$ as well, and thus also $+\square_i^{a-1}$ and $+n_i^{j_{a-1}}$; etc. So if labels of C_i^N occur in bal, then bal contains an occurrence of some $+n_i^j$, where $\overline{x_j}$ is a literal of c_i, such that the 'balancing' occurrence of $-n_i^j$ does not come from C_i^N, and so it must come from V_F^j. We interpret this as 'choosing' the literal $\overline{x_j}$ of c_i, i.e., as setting $l(c_i) = \overline{x_j}$.

The (sub)net C_i^P is similar. We let $\{x_{k_1}, x_{k_2}, \ldots, x_{k_b}\}$ be the set of positive literals of the clause c_i. The least element of C_i^P (w.r.t. the flow relation) is a transition labelled by $\triangle_i^1$; it follows from the overall structure that this transition is enabled after $-\$$ occurs. The only (complete)

behaviour of C_i^P (after being enabled) is the sequence of labels shown in Figure 6. And we reason similarly as above.

$$+\triangle_i^1\ +c_i\ -p_i^{k_1}\ -\triangle_i^1\ +\triangle_i^2\ +p_i^{k_1}\ -p_i^{k_2}\ -\triangle_i^2\ \cdots$$
$$\cdots\ +\triangle_i^b\ +p_i^{k_{b-1}}\ -p_i^{k_b}\ -\triangle_i^b$$

Figure 6. (Linear) behaviour of C_i^P

If the label $+c_i$ from C_i^P belongs to bal, then bal must also contain $+\triangle_i^1$, and thus also $-\triangle_i^1$ etc. So if labels of C_i^P occur in bal, then bal contains an occurrence of some $-p_i^j$, where x_j is a literal of c_i, such that the 'balancing' occurrence $+p_i^j$ does not come from C_i^N, and so it must come from V_T^j. We interpret this as 'choosing' the literal x_j of c_i, i.e., as setting $l(c_i) = x_j$.

We have thus completed the (obviously polynomial) construction of S_φ, and we can easily check that S_φ is a consistent 1-bounded acyclic marked graph. If φ is satisfiable, then we 'choose' a satisfying truth assignment $\mathcal{A}$ and for each clause c_i we 'choose' a literal l_i such that $\mathcal{A}$ makes l_i true, where 'choose' has the meaning described above. This leads to a balanced sequence bal. On the other hand, if a balanced sequence bal can be found, then the corresponding 'choice' of literals must be consistent (and so φ is satisfiable): if both x_j and $\overline{x}_j$ are 'chosen', then both $+n_i^j$ and $-p_k^j$ appear in bal, and both V_T^j an V_F^j must contribute to bal, which, as we have seen, is not possible. A more detailed argument can be found in the full version of the paper. ■

The previous lemma is now used to derive the desired co-NP-hardness results.

Proposition 3.4 *Both the CSC problem and the USC problem are co-NP-hard for (consistent) STGs whose underlying nets are 1-bounded acyclic marked graphs.*

Proof: We use the STG S_φ constructed in the proof of Lemma 3.1, recalling that it is a consistent 1-bounded acyclic marked graph; let us denote its (unique) consistent binary encoding by b.

Assume now that S_φ does not have the USC property. This means that there are occurrence sequences

$$M_0 \xrightarrow{\sigma_1} M_1,\ \ M_0 \xrightarrow{\sigma_2} M_2$$

such that

- $M_1 \neq M_2$
  (i.e., σ_1 and σ_2 do not contain the same transitions),
- $b(M_1) = b(M_2)$.

In the full version of the paper we show that this is the case iff there is a (nonempty) τ such that $M_0 \rightarrow^* M_1 \xrightarrow{\tau} M_2$; necessarily, τ is balanced. Moreover, such M_1, M_2 (with $b(M_1) = b(M_2)$) enable different sets of signals, so the CSC property is violated – when viewing all signals as output signals. Therefore recalling Lemma 3.1 finishes the proof. ■

Proposition 3.5 *Both the CSC problem and the USC problem are co-NP hard for live 1-bounded marked graph STGs.*

Proof: Consider the USC problem. We reuse the Petri net S_φ from the proof of Lemma 3.1. We note that the behaviour obtained by firing all transitions of S_φ is not balanced; i.e., $b(M_0)$ and $b(M_f)$, where b is the consistent boolean encoding and M_f is the final marking, differ on some signals.

> Remark. For concreteness, these unbalanced signals are x^j, c_i, $n_i^{j_d}$ (for $d = 2, 3, \ldots, a$), $p_i^{k_d}$ (for $d = 1, 2, \ldots, b-1$).

We define a new STG S'_φ by adding a 'final segment' to S_φ: we add a fresh signal f and construct a 'linear' net N_f with the behaviour

$$+f\ \ell_1\ \ell_2\ \cdots\ \ell_k\ -f$$

where ℓ_i are the labels compensating the unbalance of S_φ; they include $-x^j$, $-c_i$, $+n_i^{j_2}$, etc.; we note that each nonempty sequence of transitions of N_f is unbalanced. The net N_f will be prompted in S'_φ after all transitions of S_φ occur; the final transition of N_f will then restore the initial marking M_0.

Hence S'_φ is an STG whose underlying net is a live and 1-bounded marked graph. It is easy to see that any sequence containing precisely one occurrence of each transition of S'_φ is balanced. Let b' be the unique consistent boolean encoding of S'_φ.

We show that S_φ has the USC property iff S'_φ has the USC property, which proves the second part of the proposition.

It is trivial that if S_φ does not have the USC property, then S'_φ does not have it either. For the other direction, assume that S'_φ does not have the USC property. Then there is a *witness of the USC-violation*, i.e. two occurrence sequences

$$M_0 \xrightarrow{\sigma_1} M_1,\ \ M_0 \xrightarrow{\sigma_2} M_2$$

as in the proof of Proposition 3.4.

Let us assume that the witness is *minimal* in the sense that neither σ_1 nor σ_2 can be shortened. We prove that this minimal witness also corresponds to a USC-violation in the Petri net S_φ. It suffices to show that neither σ_1 nor σ_2 contain a transition labelled by the signal f.

Assume that one of σ_1 and σ_2, say σ_2, contains an occurrence of the signal f. Since $b'(M_1) = b'(M_2)$, we can easily check that the assumption $b'(M_2)(f) = 1$ would force $M_1 = M_2$, a contradiction. So $b'(M_2)(f) = 0$, which means that the last occurrence of f in σ_2 is $-f$. But then σ_2 can be (rearranged and) written as $\sigma_2 = \sigma_2'\sigma$ where σ contains precisely one occurrence of each transition of S'_φ. This implies $M_0 \xrightarrow{\sigma_2'} M_2$, which contradicts our minimality assumption.

Consider now the CSC property. Assume that all signals are output signals. We show that S_φ has the CSC property iff S'_φ has the CSC property. As in the USC case, it is trivial that if S_φ does not have the CSC property, then S'_φ does not have it either. For the other direction, assume S_φ has the CSC property. We have shown in Lemma 3.1 that in this case S_φ has the USC property as well. So, by the first part of this proof concerning the USC property, S'_φ has the USC property. Since USC implies CSC, S'_φ has the CSC property, and we are done. ■

We now show the upper bound, a lemma which was already (implicitly) proved in [1].

Lemma 3.2 *Both the CSC problem and the USC problem are in co-NP for (bounded or unbounded) marked graph STGs.*

Proof: Let $S = (N, M_0, \ell)$ be a normalized and consistent marked graph STG. (We recall that consistency of S can be checked in polynomial time.) It is sufficient to deal with the CSC problem; the claim for the USC problem will follow easily.

We observe that S does not have the CSC property if and only if there are sequences u_1, u_2 such that

- $M_0 \xrightarrow{u_1} M_1$, $M_0 \xrightarrow{u_2} M_2$,
- $M_1 \neq M_2$,
- for each signal a:

  $$P(u_1)(a^+) - P(u_1)(a^-) = P(u_2)(a^+) - P(u_2)(a^-)$$

- M_1, M_2 enable different output signals

To check that there is such a 'CSC-violation', a nondeterministic (polynomial) algorithm guesses a place p such that $M_1(p) \neq M_2(p)$, and guesses further whether $M_1(p) > M_2(p)$ or $M_2(p) > M_1(p)$ holds. The algorithm proceeds to guess an output signal a, and which of M_1, M_2 enables a. Assume w.l.o.g. the guess is that M_1 enables a and M_2 does not. The algorithm guesses which places of M_1 carry at least one token (including all the input places of some transition labeled by a) and which places of M_2 carry no token (including at least one input place of each transition labeled by a). The algorithm translates all these guesses into a system of linear inequalities, guesses an integer solution of polynomial size, and checks in polynomial time that it is indeed a solution. (Variables for transition sequences are replaced by variables for their Parikh vectors, and Claim 3.2 is used.) ■

Putting together Propositions 3.4 and 3.5 and Lemma 3.2 we obtain:

Theorem 3.2 *The CSC problem and the USC problem are co-NP-complete for marked graph STGs, and stay co-NP-hard for live and 1-bounded marked graph STGs as well as for 1-bounded acyclic marked graph STGs.*

Remark. Notice that in the marked graphs produced by the reduction from the proof of Lemma 3.1 there are different transitions carrying the same label. The case with injective labelling (each transition has its unique label) might well admit a polynomial algorithm but we leave this problem open here.

4. Live and bounded free-choice nets

As already mentioned, [6] shows that consistency can be decided in polynomial time for live and bounded free-choice STGs that are moreover cyclic, meaning that the initial marking is reachable from every reachable marking. It is not known whether the polynomiality result still holds if the ciclicity condition is removed, and we leave this problem open.

We now show co-NP-completeness of the CSC problem and of the USC problem for live and bounded free-choice STGs. Since live and bounded marked graphs are cyclic, Theorem 3.2 gives co-NP-hardness even for *cyclic* live and bounded free-choice STGs. So we just need to show that the complementary problem is in NP. We proceed similarly as in the marked graph case, first recalling a known result analogous to Claim 3.2; for this we use the following notation:

For a net $N = (P, T, F)$ and $X : T \to \mathbb{N}$, we denote by $N_X = (P_X, T_X, F_X)$ the subnet of N defined as follows: T_X is the set of transitions of T for which $X(t) \geq 1$, $P_X = {}^\bullet T_X \cup T_X^\bullet$, and F_X is the projection of F on $(P_X \times T_X) \cup (T_X \times P_X)$. We also recall that $Q \subseteq P$ is a *trap* in $N = (P, T, F)$ if $Q^\bullet \subseteq {}^\bullet Q$. (If a trap is marked, i.e., has at least one token, it cannot be unmarked). Here we consider only nonempty traps $Q \neq \emptyset$.

Lemma 4.1 ([12]) *Let (N, M_0) be a live and bounded free-choice Petri net, and let C_N be its incidence matrix.*

An integer vector $X_0 \geq 0$ is the Parikh vector of a transition sequence enabled at M_0 if and only if

1. $M_0 + C_N \cdot X_0 \geq 0$, *and*

2. $M = M_0 + C_N \cdot X_0$ *marks all traps of* N_{X_0}.

Theorem 4.1 *The CSC problem and the USC problem are co-NP-complete for live and bounded free-choice STGs.*

Proof: As mentioned above, co-NP-hardness follows from Theorem 3.2 (even when the Petri nets are also cyclic).

A nondeterministic polynomial algorithm for showing that a given (consistent) live and bounded free-choice STG does not have the CSC property (or the USC property) can be constructed as in the proof of Lemma 3.2, using Lemma 4.1 instead of Claim 3.2.

A little difficulty is the fact that a (nonnegative integer) solution of $M_0 + C_N \cdot X \geq 0$ may not be the Parikh vector of an occurrence sequence. The algorithm handles this problem by guessing (and requiring in the system of inequalities) which components of X are positive and which are zero; then it guesses a subset P' of places of N_X, verifies that P' does not contain a trap in N_X (which can be easily done in polynomial time) and requires (in the constructed system of inequalities) that $M_0 + C_N \cdot X$ is positive for all places of N_X outside P'. ■

In the next section we show the importance of the assumption of liveness.

5. More general nets

We study the complexity of the consistency, CSC, and USC problems for more general classes of STGs. The proofs of the results can be found in the full version.

By a straightforward use of standard techniques of Petri net theory (using the reachability problem for k-bounded nets) we can show:

Proposition 5.1 *The consistency problem, the CSC problem and the USC problem are PSPACE-complete for k-bounded nets (for any fixed k).*

An arbitrary 1-bounded STG can be transformed into a 1-bounded free-choice STG by means of the operation illustrated in Figure 7 [1] while preserving consistency. This leads to the following result:

Proposition 5.2 *The consistency problem for 1-bounded free-choice STGs (not necessarily live) is PSPACE-complete.*

Using reductions from and to the reachability problem of general Petri nets, we can show

[1] This operation is closely related to the "releasing arcs"-technique, see e.g. [4]

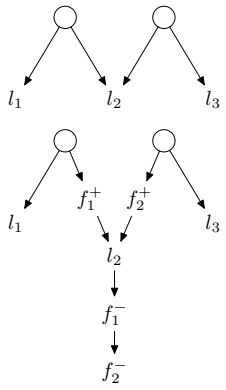

Figure 7. Transforming a 1-bounded STG into a 1-bounded free-choice STG

Proposition 5.3 *The consistency problem and the CSC problem for general STGs are decidable but EXPSPACE-hard.*

6. Conclusions and related work

We have explored the complexity of the consistency and the CSC problem for several classes of STGs. The main result shows that deciding the CSC property is co-NP-complete even for 1-bounded and acyclic marked graph STG and for 1-bounded and live marked graph STGs. The same result holds for the USC property. This result explains why none of the existing approaches for checking the USC or the CSC property in marked graph STGs is polynomial or complete.

In [11] the USC property was studied for live and 1-bounded marked graph STGs with injective labeling (i.e., one up-transition and one down-transition per signal). A sufficient condition for the USC property to hold is presented, and it is shown that it can be checked in polynomial time. The condition is conjectured to be also necessary, which would imply that checking the USC property is polynomial. The reduction used in our NP-completeness result transforms a formula into an STG in which several signals have two up- and two down-transitions, and so it does not apply to this case. The complexity of the USC property for this particular case is left for future research.

In [13] the result of [11] is extended to the case in which the STG may have several up- and down-transitions per signal. The paper presents a generalization of the sufficient condition of [11]. Our NP-completeness result shows that if P≠NP then the condition is not necessary or it cannot be checked in polynomial time, or both. In fact, we conjecture that the condition is neither necessary, nor can be checked in polynomial time (it requires to establish a property for a

potentially exponential number of objects).

In [14] it is shown that a live and 1-bounded marked-graph STGs violates the USC property iff the STG has a so-called complementary path. The paper proposes an algorithm that searches for such paths. The worst-case complexity of the algorithm is exponential, and by our result this is unavoidable unless P≠NP.

In [9] a polynomial algorithm is presented that detects all violations of the CSC property in a live and bounded free-choice STG. However, the algorithm may also give false positives, i.e., it may detect false violations. Our result shows that if P≠NP then every polynomial algorithm must produce false positives or false negatives.

In [1] a procedure is described that, given a marked-graph STG, constructs in polynomial time an Integer Linear Programming (ILP) problem such that the STG violates the CSC property if and only if the problem has a solution. Our result shows that, unless P≠NP, ILP is necessary, and cannot be replaced by ordinary Linear Programming (recall that Linear Programming problems can be solved in polynomial time).

In [7, 8] it is shown how to check the CSC property for arbitrary bounded STGs using net unfoldings and ILP-solvers or SAT-solvers. Given a bounded STG S, an object is constructed called the unfolding of S. This unfolding is used to generate an ILP problem (a boolean formula) such that S violates the CSC property iff the ILP problem has a solution (iff the formula is satisfiable). If S is a live and 1-bounded marked graph, then the unfolding of S has polynomial size in S ([5], Theorem 4.14). This shows that, even for marked graphs, ILP-solvers or SAT-solvers are unlikely to be replaceable by other tools with polynomial running time: if P≠NP, then no polynomial algorithm taking the unfolding of S as input can decide the CSC or the USC property.

Finally, it could be argued that the important problem in practice is not to decide whether a given STG satisfies the CSC property, but to transform an STG that does not satisfy the CSC property into another one that does. In [2] an automatic, very efficient procedure for such a transformation is presented. Unfortunately, the procedure adds many additional signals (one per place of the STG), and so in most cases its output is only useful as a first approximation to the design. The optimization of this first approximation has to be carried out by a (possibly automatic) trial and error procedure in which a candidate for an optimized STG is guessed. The candidate must be checked for the CSC property, which brings us back to the problem discussed in this paper.

Acknowledgments. The first author thanks Jordi Cortadella and José Carmona for helpful discussions.

References

[1] J. Carmona and J. Cortadella. ILP models for the synthesis of asynchronous control circuits. In *2003 International Conference on Computer-Aided Design (ICCAD'03), November 9-13, 2003, San Jose, CA, USA*, pages 818–826. IEEE Computer Society / ACM, 2003.

[2] J. Carmona, J. Cortadella, and E. Pastor. A structural encoding technique for the synthesis of asynchronous circuits. In *Proc. Int. Conf. on Application of Concurrency Theory to System Design*, pages 157–166. IEEE Computer Society, 2001.

[3] T.-A. Chu. *Synthesis of Self-Timed VLSI Circuits from Graph-theoretic Specifi cations*. PhD thesis, MIT, 1987.

[4] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.

[5] J. Esparza. Model checking using net unfoldings. *Science of Computer Programming*, 23:151–195, 1994.

[6] J. Esparza. A polynomial-time algorithm for checking consistency of free-choice signal transition graphs. *Fundamenta Informaticae*, 62(2):197–220, 2004.

[7] V. Khomenko, M. Koutny, and A. Yakovlev. Detecting State Coding Conflicts in STGs Using Integer Programming. In *Proc. of the Design, Automation and Test in Europe Conference and Exhibition*, pages 338–345. IEEE Computer Society, 2002.

[8] V. Khomenko, M. Koutny, and A. Yakovlev. Detecting State Coding Conflicts in STG Unfoldings using SAT. In *Proc. of the 4th Int. Conf. on Application of Concurrency to System Design*, pages 16–25. IEEE Computer Society, 2004.

[9] E. Pastor and J. Cortadella. Polynomial algorithms for the synthesis for hazard-free circuits from signal transition graphs. In *1993 International Conference on Computer-Aided Design (ICCAD'93), Santa Clara, CA, USA*, pages 250–254. IEEE Computer Society / ACM, 1993.

[10] L. Rosenblum and A. Yakovlev. Signal graphs: from self-timed to timed ones. In *Proc. Int. Workshop on Timed Petri nets*, pages 199–207. IEEE Computer Society, 1985.

[11] P. Vanbekbergen, F. Catthoor, G. Goossens, and H. D. Man. Optimized synthesis of asynchronous control circuits from graph-theoretic specifi cations. In *1990 International Conference on Computer-Aided Design (ICCAD'90)*, pages 184–197. IEEE Computer Society, 1990.

[12] H. Yamasaki, J. Huang, and T. Murata. Reachability analysis of petri nets via structural and behavioral classifi cations of transitions. *Petri Net Newsletter*, (60):5–21, 2001.

[13] C. Ykman-Couvreur, B. Lin, G. Goossens, and H. D. Man. Synthesis and optimization of asynchronous controllers based on extended lock graph theory. In *4th European Conference on Design Automation, Paris, France*, pages 512–517. IEEE Computer Society, 1993.

[14] M. Yu and P. Subrahmanyam. A new approach for checking the unique state coding property of signal transition graphs. In *Proc. 3rd Int. European Conference on Design Automation*, pages 312–321. IEEE Computer Society, 1992.