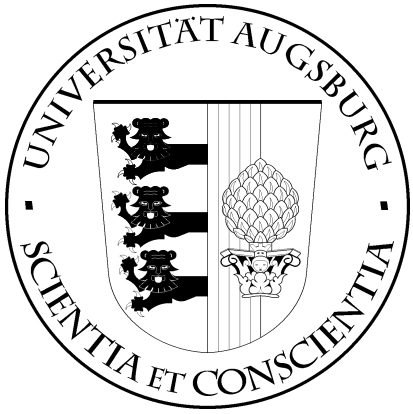
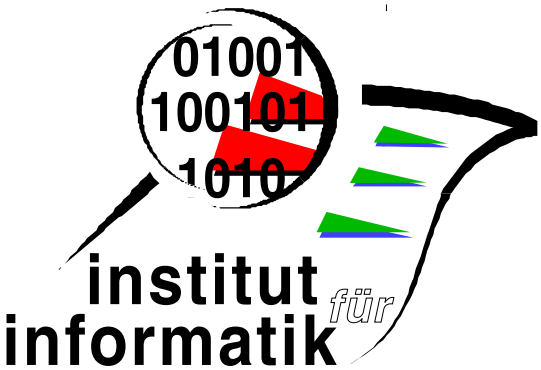
Universität Augsburg



Component Refinement and CSC Solving for STG Decomposition

Mark Schäfer and Walter Vogler

Report 2004-13 Oktober 2004



Institut für Informatik
D-86135 Augsburg

Component Refinement and CSC Solving for STG Decomposition*

Mark Schäfer and Walter Vogler

University of Augsburg, Germany
{schaefer,vogler}@informatik.uni-augsburg.de

Abstract. STGs give a formalism for the description of asynchronous circuits based on Petri nets. To overcome the state explosion problem one may encounter during circuit synthesis, a nondeterministic algorithm for decomposing STGs was suggested by Chu and improved by one of the present authors.
In this paper it is studied how CSC solving (which is essential for circuit synthesis) can be combined with decomposition. For this purpose the correctness definition for decomposition is enhanced with internal signals and it is shown that speed-independent CSC solving preserves correctness. The latter uses a more general result about correctness of top-down decomposition. Furthermore, we apply our definition to give the first correctness proof for the decomposition method of Carmona and Cortadella [CC03].
Keywords: STG, system decomposition, CSC, implementation relation, speed-independent

1 Introduction

Signal Transition Graphs (STG) are a formalism for the description of asynchronous circuit behaviour. An STG is a labelled Petri net where the labels denote signal changes between logical high and logical low. The synthesis of circuits from STGs is supported by several tools, e.g. PETRIFY [CKK+97], and it often involves the generation of the reachability graph, which may have a size exponential in the size of the STG (state explosion). To cope with this problem, Chu suggested a nondeterministic method for decomposing an STG (without internal signals) into several smaller ones [Chu87], see also [KKT93]. While there are strong restrictions on the structure and labelling of STGs in [Chu87], the improved decomposition algorithm of Vogler, Wollowski and Kangsah [VW02,VK04] works under – comparatively moderate – restrictions on the labelling only.

Roughly, this decomposition algorithm works as follows. Initially, a partition of the output signals has to be chosen, and for each set in this partition a component producing the respective output signals will be constructed as follows.

For each component, our algorithm finds a set of signals that (at least initially) can be regarded as irrelevant for the output signals under consideration; then, it takes a copy of the original STG and turns each transition corresponding to an irrelevant signal into a dummy (λ-labelled) transition; finally, it tries to remove all dummy transitions by so-called secure transition contractions and deletions of (structurally) redundant places or redundant transitions.

In general, our algorithm might find during this process that additional signals are relevant; then, it has to start anew from a suitably modified copy of the original STG – which eventually gives a correct component as proven in [VW02,VK04].

* This work was partially supported by the DFG-project 'STG-Dekomposition' Vo615/7-1.

Complete state coding (CSC) is an important property for STGs and must be achieved before an asynchronous circuit can be synthesised; e.g. PETRIFY can *solve CSC*, i.e. modify an STG on the basis of its reachability graph such that CSC holds. While some decomposition methods [CC03,YOM04] have to assume that the original STG satisfies CSC, our decomposition algorithm is more general since it does not presuppose this; on the other hand, the methods in [CC03,YOM04] construct components with CSC, while our components might not have CSC. For each such component one can solve CSC and synthesise a separate circuit e.g. by using PETRIFY; compared to solving CSC for the original STG (with its potentially huge reachability graph) and synthesising one circuit, this can be much faster [VK04].

One would expect that the components generated by our decomposition algorithm are still correct when they have been modified to achieve CSC, and in fact it would also be very interesting in what sense CSC-solving with PETRIFY is correct – independently of the issue of decomposition; it seems that no correctness for this has been proven so far. For such correctness results, one needs a correctness definition that takes *internal signals* into account.

The purpose of this paper is to enhance the correctness definition of [VW02,VK04] appropriately, to study its properties and give applications in the area of decomposition and CSC-solving.

As the main property of the new correctness notion, we show that it is preserved when decomposition is performed stepwise. While this correctness of top-down decomposition is of interest in itself, it can in particular be used to improve the efficiency of our decomposition algorithm. Then we prove that CSC-solving for speed-independent circuits as performed by PETRIFY is correct in our sense. With our result on the correctness of top-down decomposition, we then conclude that speed-independent CSC-solving can indeed be combined with the decomposition algorithm of [VW02,VK04]. As another contribution, we prove that the decomposition method in [CC03] is correct in the sense of our enhanced correctness definition; in [CC03] itself, no correctness proof is given.

The paper is organized as follows. In the next section, Petri Nets, STGs and their basic notions are introduced. Furthermore the correctness definition is enhanced with internal signals. In Section 2, we prove top-down decomposition correct in terms of our enhanced correctness definition; the succeeding section studies correctness of speed-independent CSC solving on its own and in combination with decomposition. Section 5 shows the correctness for the approach of [CC03]. We conclude with Section 6.

2 Basic Definitions

This section provides the basic notions for Petri nets and STGs, for a more detailed explanation confer e.g. [Pet81,CKK$^+$02].

A *Petri net* is a 4-tuple $N = (P, T, W, M_N)$ where P is a finite set of *places* and T a finite set of *transitions* with $P \cap T = \emptyset$. $W : P \times T \cup T \times P \to \mathbb{N}_0$ is the *weight function* and M_N the *initial marking*, where a *marking* is a function $P \to \mathbb{N}_0$. A *node* is a place or a transition and a Petri net can be considered as a bipartite graph with weighted and directed edges between its nodes. A marking is a function which assigns a number of *tokens* to each place. Whenever a Petri net N, N', N_1, etc. is introduced,

the corresponding tuples $(P,T,W,M_N),(P',T',W',M_{N'})$, (P_1,T_1,W_1,M_{N_1}) etc. are introduced implicitly and the same applies to STGs later on.

The *preset* of a node x is denoted as ${}^\bullet x$ and defined by ${}^\bullet x = \{y \in P \cup T \mid W(y,x) > 0\}$, the *postset* of a node x is denoted as $x^\bullet$ and defined by $x^\bullet = \{y \in P \cup T \mid W(x,y) > 0\}$. We write ${}^\bullet x^\bullet$ as shorthand for ${}^\bullet x \cup x^\bullet$. All these notions are extended to sets as usual. We say that there is an *arc* from each $y \in {}^\bullet x$ to x.

In a graphical representation of a Petri net places are drawn as circles, transitions as rectangles, the weight function as directed arcs xy (labelled with $W(x,y)$ if $W(x,y) > 1$) and a marking of a place as a number or as a set of small dots drawn in the interior of the corresponding circle.

Given a *sequence* $x \in X^*$, and a set $X' \subseteq X$, $x{\downarrow_{X'}}$ denotes the *projection of x onto X'* and is obtained from x by omitting all elements not in X'. This is extended to sets of sequences as usual, i.e. element-wise.

A transition t is *enabled under a marking M* if $\forall p \in {}^\bullet t : M(p) \geq W(p,t)$, which is denoted by $M[t\rangle$. An enabled transition can *fire* or *occur* yielding a new marking M', written as $M[t\rangle M'$, if $M[t\rangle$ and $M'(p) = M(p) - W(p,t) + W(t,p)$ for all $p \in P$.

A transition sequence $v = t_0 t_1 \dots t_n$ is *enabled under a marking M* (yielding M') if $M[t_0\rangle M_0[t_1\rangle M_1 \dots M_{n-1}[t_n\rangle M_n = M'$, and we write $M[v\rangle$, $M[v\rangle M'$ resp.; v is called *firing sequence* if $M_N[v\rangle$. The empty transition sequence λ is enabled under every marking.

M' is called *reachable from M* if a transition sequence v with $M[v\rangle M'$ exists. The set of all markings reachable from M is denoted by $[M\rangle$. $[M_N\rangle$ is the set of *reachable markings* (of N), and we only deal with N where this set is finite (i.e. N is bounded).

An *STG* is a tuple $N = (P,T,W,M_N,In,Out,Int,l)$ where (P,T,W,M_N) is a Petri net and In, Out and Int are disjoint sets of *input*, *output* and *internal signals*. We define the set of all signals $Sig := In \cup Out \cup Int$, the set of *locally controlled* or just *local signals* $Loc := Out \cup Int$ and the set of all *external signals* $Ext := In \cup Out$. $l : T \to Sig \times \{+,-\}$ is the labelling function. In this paper we do not have to consider λ-labelled *dummy transitions*, which play an important role in the decomposition algorithm of [VW02,VK04].

An STG can be taken as a formalism for *asynchronous circuits*. Such a circuit has input signals, which are under the control of its environment, and local signals, whose values are changed by the circuit. The STG describes which output and internal signals should be performed; at the same time, it describes assumptions about the environment, which should perform input signals only if this is specified by the STG.

$Sig \times \{+,-\}$ or short $Sig\pm$ is the set of *signal changes* or *signal transitions*; its elements are denoted shortly as $a+$, $a-$ resp. instead of $(a,+)$, $(a,-)$ resp. A plus sign denotes that a signal value changes from *logical low* (written as 0) to *logical high* (written as 1), and a minus sign denotes the other direction. We write $a\pm$ if it is not important or unknown which direction takes place. If such a term appears more than once in the same context, it always denotes the same direction.

Some of the results of this paper do not depend on the fact that transition labels are of the form $a+$ or $a-$, i.e. they can be applied in any setting where actions can be regarded as inputs, outputs or internal.

We lift the notion of enabledness to transition labels: We write $M[a\pm\rangle\rangle M'$ if $M[t\rangle M'$ and $l(t) = a\pm$. This is extended to sequences as usual. A sequence $v \in (Sig\pm)^*$

is called a *trace of a marking* M if $M[v\rangle\rangle$, and a *trace* of N if $M = M_N$. The *language of* N is the set of all traces of N and denoted by $L(N)$.

The *reachability graph* RG_N of an STG N is an edge-labelled directed graph on the reachable markings with M_N as root; there is an edge from M to M' labelled $s\pm$ whenever $M[s\pm\rangle\rangle M'$. RG_N can be seen as a finite automaton (where all states are final), and $L(N)$ is the language of this automaton. N is *deterministic* if its reachability graph is a deterministic automaton, i.e. if for each reachable marking M and each signal transition $s\pm$ there is at most one M' with $M[s\pm\rangle\rangle M'$.

The identity of the transitions or places of an STG, as well as the names of the internal signals are not relevant for us; hence, we regard STGs N and N' as equal if they are *externally isomorphic*, i.e. if they have the same input and output signals, and we can rename the internal signals of N and then map the transitions (places resp.) of the resulting STG bijectively onto the transitions (places resp.) of N' such that the weight function, the marking and the labelling are preserved. (Altogether, the external signals are preserved while the internal signals might be renamed.)

For the modular construction of STGs, the operations *hiding*, *relabelling* and *parallel composition* are of interest.

Given an STG N and a set H of signals with $H \cap In = \emptyset$, the *hiding of* H results in the STG $N/H = (P, T, W, M_N, In, Out \setminus H, Int \cup H, l)$.

Given a bijection ϕ defined for all external signals of N, the *relabelling* of N is $\phi(N) = (P, T, W, M_0, \phi(In), \phi(Out), Int, \phi \circ l)$; this assumes that, if necessary, the internal signals of N are renamed such that $Int \cap (\phi(In) \cup \phi(Out)) = \emptyset$ and ϕ is extended to be the identity on the internal signals.

Observe that hiding and relabelling preserve determinism as defined above and the same will apply for parallel composition. In particular hiding does not change the identity of signals or removes them completely from the STG as it is done in other settings.

In the following definition of *parallel composition* $\|$, we will have to consider the distinction between input, output and internal signals. The idea of parallel composition is that the composed systems run in parallel synchronising on common signals. Since a system controls its outputs, we cannot allow a signal to be an output of more than one component; input signals, on the other hand, can be shared. An output signal of one component can be an input of one or several others, and in any case it is an output of the composition. Internal signals of one component are not shared with other components (this can be achieved with a suitable renaming) and they become internal signals of the composition.

A composition can also be ill-defined due to what e.g. Ebergen [Ebe92] calls computation interference; this is a semantic problem, and we will not consider it here, but later in the definition of correctness.

The parallel composition of STGs N_1 and N_2 is defined if $Loc_1 \cap Loc_2 = \emptyset$ and $Int_1 \cap In_2 = Int_2 \cap In_1 = \emptyset$. Then, let $A = Sig_1 \cap Sig_2$ be the set of common signals; observe that A contains no internal signals. If e.g. s is an output of N_1 and an input of N_2, then an occurrence of s in N_1 is 'seen' by N_2, i.e. it must be accompanied by an occurrence of s in N_2. Since we do not know a priori which $s\pm$-labelled transition of N_2 will occur together with some $s\pm$-labelled transition of N_1, we have to allow for each possible pairing. Thus, the *parallel composition* $N = N_1 \parallel N_2$ is obtained

from the disjoint union of N_1 and N_2 by combining each $s\pm$-labelled transition t_1 of N_1 with each $s\pm$-labelled transition t_2 from N_2 if $s \in A$. In the formal definition of parallel composition, $*$ is used as a dummy element, which is formally combined e.g. with those transitions that do not have their label in the synchronisation set A. (We assume that $*$ is not a transition or a place of any net.) Thus, N is defined by

$$\begin{aligned}
P &= P_1 \times \{*\} \cup \{*\} \times P_2 \\
T &= \{(t_1, t_2) \mid t_1 \in T_1, t_2 \in T_2, l_1(t_1) = l_2(t_2) \in A\pm\} \\
&\quad \cup \{(t_1, *) \mid t_1 \in T_1, l_1(t_1) \notin A\pm\} \\
&\quad \cup \{(*, t_2) \mid t_2 \in T_2, l_2(t_2) \notin A\pm\} \\
W((p_1, p_2), (t_1, t_2)) &= \begin{cases} W_1(p_1, t_1) & \text{if } p_1 \in P_1,\ t_1 \in T_1 \\ W_2(p_2, t_2) & \text{if } p_2 \in P_2,\ t_2 \in T_2 \end{cases} \\
W((t_1, t_2), (p_1, p_2)) &= \begin{cases} W_1(t_1, p_1) & \text{if } p_1 \in P_1,\ t_1 \in T_1 \\ W_2(t_2, p_2) & \text{if } p_2 \in P_2,\ t_2 \in T_2 \end{cases} \\
l((t_1, t_2)) &= \begin{cases} l_1(t_1) & \text{if } t_1 \in T_1 \\ l_2(t_2) & \text{if } t_2 \in T_2 \end{cases} \\
M_N &= M_{N_1} \dot\cup M_{N_2}, \text{ i.e.} \\
M_N((p_1, p_2)) &= \begin{cases} M_{N_1}(p_1) & \text{if } p_1 \in P_1 \\ M_{N_2}(p_2) & \text{if } p_2 \in P_2 \end{cases} \\
Int = Int_1 \cup Int_2 \qquad & Out = Out_1 \cup Out_2 \qquad In = (In_1 \cup In_2) - (Out_1 \cup Out_2)
\end{aligned}$$

It is not hard to see that parallel composition is associative and commutative up to external isomorphism and $||_{i \in I} N_i$ is defined if each $N_i || N_j$ is defined. Furthermore, one can consider the place set of the composition as the disjoint union of the place sets of the components; therefore, we can consider markings of the composition (regarded as multisets) as the disjoint union of markings of the components; the latter makes clear what we mean by the restriction $M\big|_{P_i}$ for a marking M of the composition.

STGs together with the three operations defined above form a *circuit algebra* as defined in Dill's PhD thesis [Dil88], when regarding externally isomorphic STGs as equal. For our further considerations we will use the properties

$$(C6) : (N/H_1)/H_2 = N/(H_1 \cup H_2) \text{ and}$$

$$(C8) : N_1/H_1 || N_2/H_2 = (N_1||N_2)/(H_1 \cup H_2) \text{ if } H_i \cap Sig_{3-i} = \emptyset, i = 1, 2$$

satisfied by a circuit algebra.[1]

Let RG be the reachability graph of an STG N. A *state vector* is a function $sv : Sig \to \{0, 1\}$ where '0' means logical low and '1' logical high. A *state assignment* assigns a state vector to each marking M of RG denoted by sv_M.

A state assignment must satisfy for every signal $x \in Sig$ and every pair of markings $M, M' \in [M_N\rangle$:

$$M[x+\rangle\rangle M' \text{ implies } sv_M(x) = 0, sv_{M'} = 1$$

[1] There are 7 additional laws a circuit algebra must fulfil (in our setting): (C1) $(N_1||N_2)||N_3 = N_1||(N_2||N_3) = N_1||N_2||N_3$, (C2): $N_1||N_2 = N_2||N_1$; (C3): $\phi_2(\phi_1(N)) = (\phi_2 \circ \phi_1)(N)$, (C4): $\phi(N_1||N_2) = \phi(N_1)||\phi(N_2)$, (C5): $id(N) = N$, (C7): $N/\emptyset = N$, (C9): $\phi(N/H) = \phi'(N)/\phi'(H)$.
It is clear that these properties are satisfied for our definitions, where (C4) and (C9) only have to hold if both sides are defined.

$$M[x-\rangle\rangle M' \text{ implies } sv_M(x) = 0, sv_{M'} = 1$$

$$M[y\pm\rangle\rangle M' \text{ for } y \neq x \text{ implies } sv_M(x) = sv_{M'}(x)$$

If such an assignment exists, it is uniquely defined by these properties, and the reachability graph (and also the underlying STG) is called *consistent*. From an *inconsistent* STG, one cannot synthesise a circuit.

Another necessary condition for synthesis is *complete state coding (CSC)*. We say that a consistent *RG* (and N) *has CSC* if:

$$\forall x \in Loc,\ M, M' \in [M_N\rangle : sv_M = sv_{M'} \Rightarrow (M[x\pm\rangle\rangle \ \Leftrightarrow\ M'[x\pm\rangle\rangle)$$

If *RG violates CSC*, no circuit can be synthesised because a circuit determines the next local signal changes only from the current state of its signals (the state vector); hence, the circuit cannot distinguish the two markings with the same state vector and the same local signals must be enabled. It is possible that different input signals are enabled in M and M' because these are not controlled by the circuit.

As mentioned in the introduction, PETRIFY can modify an STG such that CSC is satisfied. If one is interested in speed-independent circuits, as we are in this paper, one can require that PETRIFY preserves the following important property.

Definition 1 (Input Properness). An *STG* is *input proper* if no input signal becomes enabled by the occurrence of an internal signal, i.e. $M_1[t\rangle M_2$ with M_1 a reachable marking, $\neg M_1[a\rangle\rangle$ and $M_2[a\rangle\rangle$, $a \in In$, implies $l(t) \notin Int$. □

Recall that an STG also specifies which inputs the environment may perform; if the environment performs an input that is not enabled in the current marking of the STG, then such an unexpected input may lead to a malfunction of the circuit. To meet this assumption, the environment must "know" whether an input is expected or not. But if input properness is violated, the environment cannot see whether the respective input is already allowed, since internal signal transitions cannot be observed from the outside.

Actually, the implementation of non-input-proper STGs is still possible, but one has to make *timing assumptions* about the relative order of signal transitions, e.g. one might assume that an input signal transition is slower than an internal signal transition if both are triggered by the same output. Such assumptions are not necessary for input proper STGs, and *speed-independent* implementations are possible.

Now we are ready to give our improved correctness definition; afterwards, we will explain its specific properties and why they are sound.

Definition 2 (Correct Decomposition). A collection of deterministic components $(C_i)_{i\in I}$ is a *correct decomposition* of (or simply *correct w.r.t.* a deterministic STG N – also called *specification – when hiding* H, if the parallel composition $C' = ||_{i\in I} C_i$ is defined, $C = C'/H$, $In_C \subseteq In_N$, $Out_C \subseteq Out_N$ and there is an *STG-bisimulation* $\mathcal{B}$ between the markings of N and those of C with the following properties:

1. $(M_N, M_C) \in \mathcal{B}$
2. For all $(M, M') \in \mathcal{B}$, we have:

(N1) If $a \in In_N$ and $M[a\pm\rangle\rangle M_1$, then either $a \in In_C$, $M'[a\pm\rangle\rangle M_1'$ and $(M_1, M_1') \in \mathcal{B}$ for some M_1' or $a \notin In_C$ and $(M_1, M') \in \mathcal{B}$.

(N2) If $x \in Out_N$ and $M[x\pm\rangle\rangle M_1$, then $M'[vx\pm\rangle\rangle M_1'$ and $(M_1, M_1') \in \mathcal{B}$ for some M_1' with $v \in (Int_C\pm)^*$.

(N3) If $u \in Int_N$ and $M[u\pm\rangle\rangle M_1$, then $M'[v\rangle\rangle M_1'$ and $(M_1, M_1') \in \mathcal{B}$ for some M_1' and $v \in (Int_{C'}\pm)^*$.

(C1) If $x \in Out_C$ and $M'[x\pm\rangle\rangle M_1'$, then $M[vx\pm\rangle\rangle M_1$ and $(M_1, M_1') \in \mathcal{B}$ for some M_1 with $v \in (Int_N\pm)^*$.

(C2) If $x \in Out_i$ for some $i \in I$ and $M'|_{P_i}[x\pm\rangle\rangle$, then $M'[x\pm\rangle\rangle$. (no *computation interference*)

(C3) If $u \in Int_C$ and $M'[u\pm\rangle\rangle M_1'$, then $M[v\rangle\rangle M_1$ and $(M_1, M_1') \in \mathcal{B}$ for some M_1 and $v \in (Int_N\pm)^*$.

Here, and whenever we have a collection $(C_i)_{i\in I}$ in the following, P_i stands for P_{C_i}, Out_i for Out_{C_i} etc.

In the most simple case, $(C_i)_{i\in I}$ consists of just one component C_1 and H is empty; in this case we say that C_1 is a *(correct) implementation* of N, and (C2) is always trivially true. □

$\mathcal{B}$ describes how behaviour of N and C closely match each other, similar to ordinary bisimulation. As in [VW02,VK04], we allow Out_C to be a proper subset of Out_N for the case that there are output signals, which are in fact never produced by the specification. Our decomposition algorithm actually only produces components C_i where $Out_C = Out_N$; in any case, if equality is desired, it can be achieved by formally adding the missing output signals $Out_N \setminus Out_C$ to some set Out_i.

For a different reason we allow In_C to be a proper subset of In_N; there are cases where some inputs are just *irrelevant* for the behaviour of a circuit, but they were possibly included by some design error. The decomposition algorithm might detect such signals, since they are not needed for any component. Because of this possibility, in (N1) an input signal transition of the specification does not have to be matched by the implementation.

(C2) ensures that no computation interference (mentioned before the definition of parallel composition) occurs; i.e. if a component produces an output (which is under the control of this component), then the other components expect this signal if it belongs to their inputs, and no malfunction of these other components must be feared. (C2) is actually also satisfied for $x \in Int_i$, since internal signals of one component are by definition unknown to the other components.

Remarkably, there is no condition that requires a matching for an input occurring in the implementation. On the one hand, if also the specification allows such an input in a matching marking, then the markings after the input must match again by (N1) due to determinism. On the other hand, there are very natural decompositions which allow additional inputs compared to the specification, and it does no harm to include these decompositions in our definition: since the specification also describes which inputs are or are not allowed for the environment, the additional inputs will actually never occur if the decomposition runs in an environment it is meant for. (The additional input leads to a marking which in a way corresponds to a don't-care entry in a Karnaugh-diagram.)

As a consequence, the components might have behaviour and markings that never turn up if the components run in an appropriate environment; also, these markings

do not appear in $\mathcal{B}$. A subtle property of our correctness definition is that it allows e.g. computation interference for such markings, which is perfectly reasonable since such an interference will not occur in practical use.

The features discussed so far are taken from [VW02], where some more explanations can be found. The new features deal with internal signals; they extend the definition of [VW02] conservatively: for STGs without internal signals, the two correctness notions coincide. The consequence will be that the result about top-down decomposition in the next section also applies in the setting of our decomposition algorithm, where we have not considered internal signals so far.

First of all, we allow the hiding of some output signals in the parallel composition of the components; this concerns additional signals to enable communication between the components. It is no problem that we allow hiding at the "top-level" only: by way of an example, assume that the components C_1 and C_2 communicate via a signal x which should not be visible to the other components; this would be modelled by $\big(((C_1||C_2)/\{x\}) \;||\; (||_{i\in I\setminus\{1,2\}}C_i)\big)/H$. Now this equals $||_{i\in I}C_i/(H\cup\{x\})$ by the properties (C8) and (C6) of a circuit algebra, where (C8) is applicable since x is internal to $(C_1||C_2)/\{x\}$ and hence not a signal of $||_{i\in I\setminus\{1,2\}}C_i$. We will use similar reasoning in Section 3 where a component will be replaced by a decomposition of its own.

In (N2) and (C1) output signal transitions do not have to be matched directly; (N2) allows the components to prepare the production of this output by some internal signals, e.g. to achieve CSC or to inform other components about this event; (C1) allows the specification to perform superfluous internal signals. In any case, from an external point of view each output is matched by the same output.

In contrast, input signals must be matched directly; if the implementation could precede the input by some internal signals, the environment could produce the input as specified in N at a stage where the implementation is not ready yet to receive it, which could lead to malfunction as discussed above in connection with input properness. As for computation interference, the absence of this malfunction is only checked for markings appearing in $\mathcal{B}$, since only for these the problem is practically relevant.

In fact, the direct matching of inputs implies that the implementation is in a sense input proper, at least in its "reachable behaviour": assume that $M_1[t\rangle M_2$ with M_1 a reachable marking of C, and $M_2[a\rangle\rangle$ for some $a \in In_C$; then either there is no pair (M, M_1) in the STG-bisimulation (hence, M_1 will not be reached if C works in a proper environment) or $\neg M[a\rangle\rangle$ (a proper environment will not produce a) or $M_1[a\rangle\rangle$ by (N1).

Finally, (N3) and (C3) prescribe the matching of an internal signal by a sequence of internal signals – just as in ordinary weak bisimulation. Note that we have several internal signals, since these have to be implemented physically; but regarding external behaviour, the identity of an internal signal does not matter. In principle, performing an internal signal could make a choice, e.g. by disabling an output; according to these clauses, this choice has to be matched.

Translating the treatment of internal signals in the definition of the somewhat related notion of I/O-compatibility [CC02] to our setting, one would require that e.g. in (N3) $(M_1, M') \in \mathcal{B}$ without involving any u – and analogously in (C3); the idea is that internal signals cannot make decisions in digital circuits. There are several reasons not to follow this idea. First of all, this concerns a property one might like all STGs

to have and it is not related to comparing STGs or to the communication between circuits – in contrast to e.g. computation interference; if one wants this property to ensure physical implementability, it has to hold also for markings not appearing in $\mathcal{B}$. Therefore, this property has no adequate place in a correctness definition and should be required separately. Secondly, one might want to use ME-elements [YKK$^+$96], which can make decisions; the respective signals could be internal to the parallel composition. We see it as an advantage that we can cover such cases. Finally, the alternative definition turned out to be technically inconvenient.

Observe that the alternative definition coincides with ours if the specification does not have internal signals; then, (N3) is never applicable, and in (C3) we have $v = \lambda$ and $M = M_1$.

There is another important comment. Our correctness definition concerns the correctness of a decomposition, but it also covers the question whether one STG is an implementation of another. With this notion, we will prove in Section 4 that speed-independent CSC-solving with Petrify produces a correct implementation.

One would like this implementation relation to be a preorder. Reflexivity is obvious (choose $\mathcal{B}$ as the identity), and transitivity will follow from our first main result in the next section. One would also like it to be a precongruence for the operations of interest. This is obvious for relabelling and easy for hiding (use the same STG-bisimulation). The much more important case of parallel composition will be discussed in the next section.

Actually, one can see a more general result for hiding just as easily: $(*)$ if $(C_i)_{i\in I}$ is correct w.r.t. N when hiding H, then $(C_i)_{i\in I}$ is also correct w.r.t. N/H' when hiding $H \cup H'$. As a consequence, we can apply our decomposition algorithm [VW02,VK04] also to an STG N_1 with internal signals as follows. Since the algorithm can only decompose STGs without internal signals, we change the internal signals of N_1 to outputs obtaining an STG N_2 with $N_1 = N_2/H$ for a suitable set H. Then we decompose N_2, obtaining a correct decomposition $(C_i)_{i\in I}$ of N_2. After that, the formerly internal signals are hidden in N_2 and in $||_{i\in I}C_i$ and from $(*)$ we get that $(C_i)_{i\in I}$ is a correct decomposition of $N_1 = N_2/H$ when hiding H.

3 Decomposition of Subcomponents

In this section we will show that correctness is preserved when we decompose a component of an STG decomposition into *subcomponents*. This result makes it possible to design and implement STGs in a top-down fashion.

In particular, such top-down decomposition can be useful for efficiency of our decomposition algorithm. For example, consider a case where only one component C_i of a decomposition needs a specific input signal a, which therefore will be removed from every other one by the decomposition algorithm (cf. Section 1). Alternatively, the algorithm could first construct a component C_j which generates every output signal that is not produced by C_i, and afterwards decompose it into smaller components. This way, the signal a will only be removed from one component (C_j), which can improve performance.

Stepwise decomposition is possible under two minor conditions stated in the following theorem: the composition of the subcomponents must have all output signals

of the decomposed component and its internal signals must be unknown to the other components. The first condition is often automatically true or can be achieved easily as mentioned after the definition of correctness, the latter one is an obvious restriction required by our definition of parallel composition and can trivially be fulfilled renaming internal signals. The proof of this theorem requires a careful and detailed case analysis.

Theorem 3 (Correctness of top-down decomposition).

1. *Let N be an STG and $(C_i)_{i\in I}$ a correct decomposition of N when hiding H_C. Furthermore let $(C_k)_{k\in K}$ be a correct decomposition of some C_j when hiding H_K ($j \in I$, $I \cap K = \emptyset$). Then $(C_i)_{i\in I'}$ with $I' := I \cup K - \{j\}$ is a correct decomposition of N when hiding $H_C \cup H_K$ if $\bigcup_{k\in K} Out_{C_k} \setminus H_K = Out_{C_j}$ and $(\bigcup_{k\in K} Int_{C_k} \cup H_K) \cap \bigcup_{i\in I\setminus\{j\}} Sig_{C_i} = \emptyset$.*
2. *The implementation relation is a preorder.*

Remark: One might expect that refining a component C_j of $(||_{i\in I}C_i)/H_C$ with $(||_{k\in K}C_k)/H_K$ would give the STG $\left(||_{i\in I\setminus\{j\}}C_i \;||\; (||_{k\in K}C_k/H_K)\right)/H_C$, where there is not just one hiding on the top-level as in the theorem. But with the same reasoning already used in the discussion of Definition 2, we can derive from the properties (C8) (use the second assumption on H_K) and (C6) of a circuit algebra that for $H = H_C \cup H_K$:

$$\left(||_{i\in I\setminus\{j\}}C_i \;||\; (||_{k\in K}C_k/H_K)\right)/H_C = \left((||_{i\in I'}C_i)/H_K\right)/H_C = ||_{i\in I'}C_i/H$$

Proof. We define $C = (||_{i\in I}C_i)/H_C$, $C_K = (||_{k\in K}C_k)/H_K$ and $C' = (||_{i\in I'}C_i)/H$, where $H := H_C \cup H_K$. Without loss of generality let $I = \{1, 2, \ldots, |I|\}$, $j = |I|$ and $K = \{|I|+1, |I|+2, \ldots, |I|+|K|\}$. We will write Out_i for Out_{C_i} etc.

Regarding (2), we already know that the implementation relation is reflexive; transitivity is just a special case of (1), where both hiding sets are empty and the decompositions have just one component each. So (1) tells us that, if C is an implementation of N and C' an implementation of C, then C' is an implementation of N – except that we do not have the two extra conditions required in (1). But observe that the second condition is trivially true since $\bigcup_{i\in I\setminus\{j\}} Sig_{C_i}$ is empty. The first condition is only needed to prove claims that are obvious for this restricted case, namely that the parallel composition C' (which has only one component here) is defined and that $In_{C'} \subseteq In_N$. Thus, it suffices to prove (1).

First, we show that the parallel composition of $(C_i)_{i\in I'}$ is defined.
Obviously, $Loc_i \cap Loc_{i'}$ for different i, i' with either $i, i' \in I \setminus \{j\}$ or $i, i' \in K$, because $||(C_i)_{i\in I}$ and $||(C_k)_{k\in K}$ are defined. Therefore let $k \in K, i \in I\setminus\{j\}$; then $Loc_k \cap Loc_i = (Int_k \cup Out_k) \cap Loc_i = Int_k \cap Loc_i \cup Out_k \cap Loc_i = \emptyset \cup Out_k \cap Loc_i$, by assumption about Int_k. $Out_k \cap Loc_i \subseteq (Out_j \cup H_K) \cap Loc_i = (Out_j \cap Loc_i) \cup (H_K \cap Loc_i) \subseteq (Loc_j \cap Loc_i) \cup (H_K \cap Loc_i) = \emptyset$ by the assumption about H_K and because $||_{i\in I}C_i$ is defined.

For i, i' as above, $Int_i \cap In_{i'} = \emptyset$. Let therefore i, k be as above, then $In_k \cap Int_i \subseteq In_j \cap Int_i = \emptyset$ and $Int_k \cap In_i = \emptyset$ by the assumptions.

Next, we show the requirements for the sets of output and input signals.

$$Out_{C'} = \bigcup_{i\in I'} Out_i \setminus H = \Big(\bigcup_{i\in I\setminus\{j\}} Out_i \cup \bigcup_{k\in K} Out_k\Big) \setminus H$$

$$= (\bigcup_{i\in I\setminus\{j\}} Out_i \cup (\bigcup_{k\in K} Out_k \setminus H_K)) \setminus H_C = (\bigcup_{i\in I\setminus\{j\}} Out_i \cup Out_j) \setminus H_C = \bigcup_{i\in I} Out_i \setminus H_C \subseteq Out_N$$

where the third equality holds by the second assumption on H_K.

For the input signals, we have

$$In_C = \bigcup_{i\in I} In_i \setminus \bigcup_{i\in I} Out_i \subseteq In_N \text{ and } \bigcup_{k\in K} In_k \setminus \bigcup_{k\in K} Out_k \subseteq In_j$$

It follows that

$$In_{C'} = \bigcup_{i\in I'} In_i \setminus \bigcup_{i\in I'} Out_i = (\bigcup_{i\in I\setminus\{j\}} In_i \cup \bigcup_{k\in K} In_k) \setminus (\bigcup_{i\in I\setminus\{j\}} Out_i \cup \bigcup_{k\in K} Out_k)$$

$$\subseteq (\bigcup_{i\in I\setminus\{j\}} In_i \cup In_j) \setminus (\bigcup_{i\in I\setminus\{j\}} Out_i \cup \bigcup_{k\in K} Out_k) \overset{(*)}{\subseteq} \bigcup_{i\in I} In_i \setminus (\bigcup_{i\in I\setminus\{j\}} Out_i \cup Out_j) = In_C \subseteq In_N$$

The inclusion $(*)$ might fail if we had only $(\bigcup_{k\in K} Out_k) \setminus H_K \subseteq Out_j$. Observe that we do not need to consider hiding here.

We proceed with the main part of this proof: the requirements for an STG-bisimulation between the markings of N and C'.
Let $\mathcal{B}_1$ be the STG-bisimulation between the markings of N and C and $\mathcal{B}_2$ the one between the markings of C_j and C_K. We define a relation $\mathcal{B}$ between the markings of N and C' as follows:

$$(M_1, M_2, M_3) \in \mathcal{B} \Leftrightarrow (M_1, M_2, M_j) \in \mathcal{B}_1 \text{ and } (M_j, M_3) \in \mathcal{B}_2 \text{ for some } M_j$$

where M_1 denotes a marking of N, M_2 a marking of $C_1||\ldots||C_{|I|-1}$, M_j one of C_j and M_3 a marking of $C_{|I|+1}||\ldots||C_{|I|+|K|}$; thus (M_2, M_j) can be regarded as marking of C and (M_2, M_3) as one of C'. We will show that $(C_i)_{i\in I'}$ is a correct decomposition of N when hiding H due to STG-bisimulation $\mathcal{B}$.

(1): Obviously fulfilled by definition of $\mathcal{B}$.

(2): Let $(M_1, M_2, M_3) \in \mathcal{B}$ due to $(M_1, M_2, M_j) \in \mathcal{B}_1$ and $(M_j, M_3) \in \mathcal{B}_2$ for some marking M_j of C_j.

(N1): $a \in In_N$ and $M_1[a\pm\rangle\rangle\hat{M}_1$.

1. Let $a \in In_{C'} \subseteq In_C$ (see above) and therefore $(\mathcal{B}_1)$ $(M_2, M_j)[a\pm\rangle\rangle(\hat{M}_2, \hat{M}_j)$ and $(\hat{M}_1, \hat{M}_2, \hat{M}_j) \in \mathcal{B}_1$ for some $(\hat{M}_2, \hat{M}_j)$.
   (a) If $a \notin In_j$ we get $M_j = \hat{M}_j$ and immediately $(M_2, M_3)[a\pm\rangle\rangle(\hat{M}_2, M_3)$ with $(\hat{M}_1, \hat{M}_2, M_3) \in \mathcal{B}$.
   (b) $a \in In_j$ with $M_j[a\pm\rangle\rangle\hat{M}_j$ implies $(\mathcal{B}_2)$ either $a \in In_K$ and $M_3[a\pm\rangle\rangle\hat{M}_3$, $(\hat{M}_j, \hat{M}_3) \in \mathcal{B}_2$ for some $\hat{M}_3$ or $a \notin In_K$ and $(\hat{M}_j, M_3) \in \mathcal{B}_2$.
   In the first case, we get $(M_2, M_3)[a\pm\rangle\rangle(\hat{M}_2, \hat{M}_3)$ with $(\hat{M}_1, \hat{M}_2, \hat{M}_3) \in \mathcal{B}$.
   In the latter one we get $(M_2, M_3)[a\pm\rangle\rangle(\hat{M}_2, M_3)$ with $(\hat{M}_1, \hat{M}_2, M_3) \in \mathcal{B}$.
2. Let $a \notin In_{C'}$. There are two reasons for this:
   (a) $a \notin In_C$. Then, $(\hat{M}_1, M_2, M_j) \in \mathcal{B}_1$ and by definition $(\hat{M}_1, M_2, M_3) \in \mathcal{B}$.
   (b) $a \in In_j$, but $a \notin In_i$ for $i \neq j$ and $a \notin In_K$ (a only element of C_j). Therefore, $a \in In_C$ and $(M_2, M_j)[a\pm\rangle\rangle(M_2, \hat{M}_j)$ with $(\hat{M}_1, M_2, \hat{M}_j) \in \mathcal{B}_1$, since a cannot be a signal of any other component. $a \notin In_K$ implies $(\hat{M}_j, M_3) \in \mathcal{B}_2$ and by definition $(\hat{M}_1, M_2, M_3) \in \mathcal{B}$.

(N2): Let $x \in Out_N$ and $M_1[x\pm\rangle\rangle\hat{M}_1$.

Then $(\mathcal{B}_1)$ $(M_2, M_j)[vx\pm\rangle\rangle(\hat{M}_2, \hat{M}_j)$ and $(\hat{M}, \hat{M}_2, \hat{M}_j) \in \mathcal{B}_1$ for some $(\hat{M}_2, \hat{M}_j)$ and $v \in (Int_C\pm)^*$. Let $v' = vx\pm = v_1v_2 \dots v_n$ with $v_i \in Sig_C\pm$ for $i = 1, \dots, n$ and

$$(M_2, M_j) = (M_2^0, M_j^0)[v_1\rangle\rangle(M_2^1, M_j^1)[v_2\rangle\rangle(M_2^2, M_j^2) \dots$$

$$(M_2^{n-2}, M_j^{n-2})[v_{n-1}\rangle\rangle(M_2^{n-1}, M_j^{n-1})[v_n\rangle\rangle(M_2^n, M_j^n) = (\hat{M}_2, \hat{M}_j)$$

We will show by induction over $m \in \{0, \dots, n\}$ that

$$\exists w_m \in (Int_{C'}\pm)^*\{x\pm, \lambda\}, M_3^m : (M_2, M_3)[w_m\rangle\rangle(M_2^m, M_3^m)$$

$$\wedge w_m{\downarrow_{Ext_{C'}}} = (v_1 \dots v_m){\downarrow_{Ext_C}} \wedge (M_j^m, M_3^m) \in \mathcal{B}_2$$

Observe that the case $m = n$ proves our claim.

For $m = 0$ let $M_3^0 = M_3$ and $w_0 = \lambda$. By assumption $(M_j^0, M_3^0) \in \mathcal{B}_2$.
Let now $m < n$ and $(M_2, M_3)[w_m\rangle\rangle(M_2^m, M_3^m)$, $w_m{\downarrow_{Ext_{C'}}} = (v_1 \dots v_m){\downarrow_{Ext_C}}$, $(M_j^m, M_3^m) \in \mathcal{B}_2$ and $(M_2^m, M_j^m)[v_{m+1}\rangle\rangle(M_2^{m+1}, M_j^{m+1})$. We distinguish several cases, where in items (3) - (5) we have either $v_{m+1} \in H_C$ or $v_{m+1} = v_n = x\pm$:

1. $v_{m+1} \in Int_i$ for $i \in I \setminus \{j\} \Rightarrow M_j^{m+1} = M_j^m, M_3^{m+1} = M_3^m$, $w_{m+1} = w_m v_{m+1}$ and $(M_j^{m+1}, M_3^{m+1}) \in \mathcal{B}_2$.
2. $v_{m+1} \in Int_j \Rightarrow M_2^{m+1} = M_2^m$,$M_3^m[w'_m\rangle\rangle M_3^{m+1}$, $w'_m \in (Int_K\pm)^*$, $w_{m+1} = w_m w'_m$ and $(M_j^{m+1}, M_3^{m+1}) \in \mathcal{B}_2$.
3. $v_{m+1} \in Out_i$ for $i \in I \setminus \{j\}$ and $v_{m+1} \notin In_j$: ref. item (1).
4. $v_{m+1} \in Out_i$ for $i \in I \setminus \{j\}$ and $v_{m+1} \in In_j$: (N1) implies
   (a) $v_{m+1} \in In_K \Rightarrow M_3^m[v_{m+1}\rangle\rangle M_3^{m+1}$ with $(M_j^{m+1}, M_3^{m+1}) \in \mathcal{B}_2$ and $w_{m+1} = w_m v_{m+1}$.
   (b) $v_{m+1} \notin In_K \Rightarrow M_3^{m+1} = M_3^m$ with $(M_j^{m+1}, M_3^{m+1}) \in \mathcal{B}_2$ and $w_{m+1} = w_m v_{m+1}$.
5. $v_{m+1} \in Out_j \Rightarrow M_3^m[w'_{m+1}v_{m+1}\rangle\rangle M_3^{m+1}$ with $w'_{m+1} \in (Int_K\pm)^*$, $(M_j^{m+1}, M_3^{m+1}) \in \mathcal{B}_2$ and $w_{m+1} = w_m w'_{m+1} v_{m+1}$.

(N3): Let $u \in Int_N$ and $M_1[u\pm\rangle\rangle\hat{M}_1$. Therefore $(M_2, M_j)[v\rangle\rangle(\hat{M}_2, \hat{M}_j)$ with $(\hat{M}_1, \hat{M}_2, \hat{M}_j) \in \mathcal{B}_1$ and $v \in (Int_C\pm)^*$ for some $(\hat{M}_2, \hat{M}_j)$. At this point the proof can be continued analogously to the previous item.

(C1): Let $x \in Out_{C'}$ and $(M_2, M_3)[x\pm\rangle\rangle(\hat{M}_2, \hat{M}_3)$.

1. If $x \in Out_K \setminus H_K = Out_j$ it follows that $M_j[vx\pm\rangle\rangle\hat{M}_j$, $(\hat{M}_j, \hat{M}_3) \in \mathcal{B}_2$ for some $\hat{M}_j$ and $v \in (Int_j\pm)^*$. Let $M_j[v\rangle\rangle M'_j[x\pm\rangle\rangle\hat{M}_j$; then $(M_2, M_j)[v\rangle\rangle(M_2, M'_j)$ and by (C3) $M_1[w_1\rangle\rangle M'_1$ with $w_1 \in (Int_N\pm)^*$ and $(M'_1, M_2, M'_j) \in \mathcal{B}_1$ for some $\hat{M}'_1$.
   Since $(M_2, M'_j)[x\pm\rangle\rangle(\hat{M}_2, \hat{M}_j)$ (where $M_2[x\pm\rangle\rangle\hat{M}_2$ or $\hat{M}_2 = M_2$), we get by (C1) for $\mathcal{B}_1$ that $M'_1[w_2x\pm\rangle\rangle\hat{M}_1$ and $(\hat{M}_1, \hat{M}_2, \hat{M}_j) \in \mathcal{B}_1$ for some $\hat{M}_1$ and $w_2 \in (Int_N\pm)^*$. Altogether, we get that $M_1[w_1w_2x\rangle\rangle\hat{M}_1$ with $w_1w_2 \in (Int_N\pm)^*$ and $(\hat{M}_1, \hat{M}_2, \hat{M}_3) \in \mathcal{B}$.
2. If $x \notin Out_K \setminus H_K = Out_j$, there exists an $m \in I \setminus \{j\}$ such that $x \in Out_m \subseteq Out_C$.
   (a) $x \notin In_j$ implies that neither M_3 nor M_j are changed when firing $x\pm$: $\hat{M}_3 = M_3$ and $(M_2, M_j)[x\pm\rangle\rangle(\hat{M}_2, M_j)$; we get directly $(\mathcal{B}_1)$ $M_1[wx\pm\rangle\rangle\hat{M}_1$, $(\hat{M}_1, \hat{M}_2, M_j) \in \mathcal{B}_1$ for some $\hat{M}_1$ and $w \in (Int_N\pm)^*$, and by definition of $\mathcal{B}$: $(\hat{M}_1, \hat{M}_2, \hat{M}_3) \in \mathcal{B}$.

(b) If $x \in In_j$ then $M_j[x\pm\rangle\rangle\hat{M}_j$ by (C2) for $\mathcal{B}_1$ and by (N1)
   i. $x \notin In_K$, $\hat{M}_3 = M_3$ and $(\hat{M}_j, M_3) \in \mathcal{B}_2$.
   ii. $x \in In_K$ and $(\hat{M}_j, \hat{M}_3) \in \mathcal{B}_2$.
   In both cases $(M_2, M_j)[x\pm\rangle\rangle(\hat{M}_2, \hat{M}_j)$, $M_1[wx\pm\rangle\rangle\hat{M}_1$ with $w \in (Int_N\pm)^*$ and $(\hat{M}_1, \hat{M}_2, \hat{M}_j) \in \mathcal{B}_1$, hence $(\hat{M}_1, \hat{M}_2, \hat{M}_3) \in \mathcal{B}$.

(C2): Let $x \in Out_m, m \in I'$ and $(M_2, M_3)|_{P_m}[x\pm\rangle\rangle$.

1. $x \in Out_m$ for some $m \in I \setminus \{j\}$ and $M_2|_{P_m}[x\pm\rangle\rangle$. Then $(\mathcal{B}_1)$ $(M_2, M_j)[x\pm\rangle\rangle$ and therefore $(M_2, M_3)[x\pm\rangle\rangle$, because either $x \notin In_j$ and $x \notin Sig_k$ for $k \in K$ or $x \in In_j$, and by (N1) for $\mathcal{B}_2$ either $M_3[x\pm\rangle\rangle$ or $x \notin Sig_k$, $k \in K$.
2. $x \in Out_m$ for some $m \in K$ and $M_3|_{P_m}[x\pm\rangle\rangle$. Hence, $M_3[x\pm\rangle\rangle$ by (C2) for $\mathcal{B}_2$.
   (a) If $x \in Out_K$, then $M_j[v\rangle\rangle M'_j[x\pm\rangle\rangle$ by (C1) for $\mathcal{B}_2$ for some $v \in (Int_j\pm)^*$ and M'_j. Since C_j can fire its internal signals without changing the state of the other components, we get $(M_2, M_j)[v\rangle\rangle(M_2, M'_j)$ and $(M'_1, M_2, M'_j) \in \mathcal{B}_1$ for some M'_1. Applying (C2) for $\mathcal{B}_1$ we get $(M_2, M'_j)[x\pm\rangle\rangle$ and therefore $(M_2, M_3)[x\pm\rangle\rangle$, too.
   (b) If $x \in Int_K$, i.e. $x \in H_K$, then $M_3[x\pm\rangle\rangle$ gives immediately that $(M_2, M_3)[x\pm\rangle\rangle$.

(C3): Let $u \in Int_{C'}$ and $(M_2, M_3)[u\pm\rangle\rangle(\hat{M}_2, \hat{M}_3)$.

1. If $u \in Int_i$ for $i \in I \setminus \{j\}$, then $\hat{M}_3 = M_3$, $(M_2, M_j)[u\pm\rangle\rangle(\hat{M}_2, M_j)$ and by (C3) for $\mathcal{B}_1$: $M_1[v\rangle\rangle\hat{M}_1$, $w \in (Int_N\pm)^*$ and $(\hat{M}_1, \hat{M}_2, M_j) \in \mathcal{B}_1$ for some $\hat{M}_1$ and therefore $(\hat{M}_1, \hat{M}_2, \hat{M}_3) \in \mathcal{B}$.
2. If $u \in Int_k$ for $k \in K$, then $\hat{M}_2 = M_2$ and by (C3) for $\mathcal{B}_2$: $M_j[v\rangle\rangle\hat{M}_j$, $v \in (Int_j\pm)^*$ and $(\hat{M}_j, \hat{M}_3) \in \mathcal{B}_2$ for some $\hat{M}_j$. Let $v = v_1v_2 \dots v_n$, $v_i \in Int_j\pm$ and $M_j = M_j^0[v_1\rangle\rangle M_j^1 \dots M_j^{n-1}[v_n\rangle\rangle M_j^n = \hat{M}_j$. By (C3) for $\mathcal{B}_1$ we get that $M_1 = M_1^0[w_1\rangle\rangle M_1^1[w_2\rangle\rangle \dots M_1^{n-1}[w_n\rangle\rangle M_1^n = \hat{M}_1$ with $w_i \in (Int_N\pm)^*$ and $(M_1^m, \hat{M}_2, M_j^n) \in \mathcal{B}_2$ for every $m = 0, \dots, n$. In particular, $M_1[w_1 \dots w_m\rangle\rangle\hat{M}_1$, $(\hat{M}_1, \hat{M}_2, \hat{M}_j) \in \mathcal{B}_1$ and therefore $(\hat{M}_1, \hat{M}_2, \hat{M}_3) \in \mathcal{B}$.
3. If $u \in Out_i \cap H_C$ for $i \in I \setminus \{j\}$.
   (a) $u \in In_j$. Then $(M_2, M_j)[u\pm\rangle\rangle(\hat{M}_2, \hat{M}_j)$ and by (C3) for $\mathcal{B}_1$: $M_1[v\rangle\rangle\hat{M}_1$, $v \in (Int_N\pm)^*$ and $(\hat{M}_1, \hat{M}_2, \hat{M}_j) \in \mathcal{B}_1$; by (N1) for $\mathcal{B}_2$ either $M_3[u\pm\rangle\rangle\hat{M}_3$ or $u \notin In_K$ and $\hat{M}_3 = M_3$; in both cases $(\hat{M}_j, \hat{M}_3) \in \mathcal{B}_2$ and it follows that $(\hat{M}_1, \hat{M}_2, \hat{M}_3) \in \mathcal{B}$.
   (b) $u \notin In_j$. Analogous to item (1)
4. If $u \in Out_k \cap H_C$ for $k \in K$. Then, $M_3[u\pm\rangle\rangle\hat{M}_3$ and by (C1) for $\mathcal{B}_2$ $M_j[vu\pm\rangle\rangle\hat{M}_j$ and $(\hat{M}_j, \hat{M}_3) \in \mathcal{B}_2$ for $v \in (Int_j\pm)^*$. Furthermore, $(M_2, M_j)[vu\pm\rangle\rangle(\hat{M}_2, \hat{M}_j)$ and by (C3) for $\mathcal{B}_1$ we get $M_1[w\rangle\rangle\hat{M}_1$, $w \in (Int_N\pm)^*$ and $(\hat{M}_1, \hat{M}_2, \hat{M}_j) \in \mathcal{B}_1$ (with $\hat{M}_2 = M_2$ if $u \notin In_m$ for $m \in I \setminus \{j\}$). It follows that $(\hat{M}_1, \hat{M}_2, \hat{M}_3) \in \mathcal{B}$.
5. If $u \in Out_k \cap H_K$ for $k \in K$. Then, by (C3) for $\mathcal{B}_2$: $M_j[v\rangle\rangle\hat{M}_j$, $v \in (Int_j\pm)^*$ and $(\hat{M}_j, \hat{M}_3) \in \mathcal{B}_2$ and $\hat{M}_2 = M_2$. Thus, $(M_2, M_j)[v\rangle\rangle(\hat{M}_2, \hat{M}_j)$ and by (C3) for $\mathcal{B}_1$: $M_1[w\rangle\rangle\hat{M}_1$, $w \in (Int_N\pm)^*$ and $(\hat{M}_1, \hat{M}_2, \hat{M}_j) \in \mathcal{B}_1$. Therefore by definition of $\mathcal{B}$, $(\hat{M}_1, \hat{M}_2, \hat{M}_3) \in \mathcal{B}$. □

As explained after Definition 2, our correctness definition coincides with the one of [VW02,VK04] if we restrict ourselves to STGs without internal signals; hence, the above theorem also holds in this setting (where of course no hiding is applied, i.e.

the hiding sets are taken to be empty). Therefore, the theorem can indeed be used to improve the decomposition of [VW02,VK04] as explained at the beginning of this section. It is an open problem how to group the output signals for optimal efficiency.

Surprisingly, the theorem has also an impact on the question whether the implementation relation between STGs is a precongruence for parallel composition, which we will now show under some mild restrictions. Recall that, for some $N_1||N_2$ to be defined, we only had some syntactic requirements regarding the signal sets; but the composition only makes sense in the area of circuits, if we also ensure absence of computation interference; for the following definition cf. the discussion on condition (C2) of Definition 2.

Definition 4. A parallel composition $N_1||N_2$ is *interference-free* if, for all its reachable markings (M_1, M_2), $i \in \{1, 2\}$ and $x \in Out_i$, $M_i[x\pm\rangle$ implies $(M_1, M_2)[x\pm\rangle\rangle$. □

Corollary 5. *If N_2 is a correct implementation of N_1, N_1 and N_2 have the same output signals, and $N_1||N$ is a well-defined and interference-free parallel composition, then $N_2||N$ is a correct implementation of $N_1||N$.*

Proof. Since the composition is interference-free, the identity is an STG-bisimulation showing that the family (N_1, N) is a correct decomposition of $N_1||N$; note that in this setting all conditions for an STG-bisimulation are trivially fulfilled except for (C2). With this observation, the claim follows from our theorem. □

Note that each of our operations hiding, renaming and parallel composition with another STG changes the set of output signals in the same way, such that equality of these sets is preserved.

Corollary 6 (Implementation relation as precongruence). *The implementation relation is a precongruence for hiding, relabelling and parallel composition when restricted to STGs with the same output signals.*

We will see another application of the theorem in the next section.

4 CSC-Solving for Components of a Decomposition

In this section we will prove that CSC-solving fits into our correctness definition, i.e. that it leads to a correct implementation. Theorem 3 then implies that CSC-solving can be combined with our decomposition algorithm. The latter could be shown directly without this theorem, but its use makes the following proof much easier, because we have to consider only one component. First, we will introduce an operation that the tool Petrify uses to achieve CSC.

Given an STG without CSC, Petrify can (in many cases) insert internal signals into the STG such that their values distinguish between the markings with equal state vectors and different outputs. This insertion takes place on the level of reachability graphs (as most of our considerations in this paper do). Petrify can also derive an STG for the modified reachability graph, and although this is not important for the synthesis of a circuit, it fits our manner-of-speaking well. We take the following definition of event insertion from [CKK+02]. Run with an appropriate option, Petrify performs a number of input proper event insertions arriving at an STG with CSC, and this we call *speed-independent CSC-solving*.

Definition 7 (Event insertion). Let N be a deterministic STG, $u\pm$ a signal transition not appearing in N for a (possibly new) internal signal u and $R \subseteq [M_N\rangle$. The *event insertion of $u\pm$ at region R into N* modifies the reachability graph RG as follows (and results in a corresponding STG N'):

1. For every marking $M \in R$ add a duplicate M' and add the transition $M[u\pm\rangle\rangle M'$.
2. If $M_1, M_2 \in R$ and $M_1[s\pm\rangle\rangle M_2$, add the transition $M_1'[s\pm\rangle\rangle M_2'$.
3. If $M_1 \in R$, $M_2 \notin R$ and $M_1[s\pm\rangle\rangle M_2$, remove this transition and add $M_1'[s\pm\rangle\rangle M_2$.
4. The initial marking of N' is the same as that of N. Add u to Int.

The insertion is called *input proper*, if there is no $M_1[a\pm\rangle\rangle M_2$ in RG with $a \in In$, $M_1 \in R$ and $M_2 \notin R$.

We define the *marking relation* $\mathcal{M}$ between the markings of N and of N' such that $(M_1, M_2) \in \mathcal{M}$ if $M_2 = M_1$ or $M_2 = M_1'$. □

An example for this definition can be found in Figure 1.

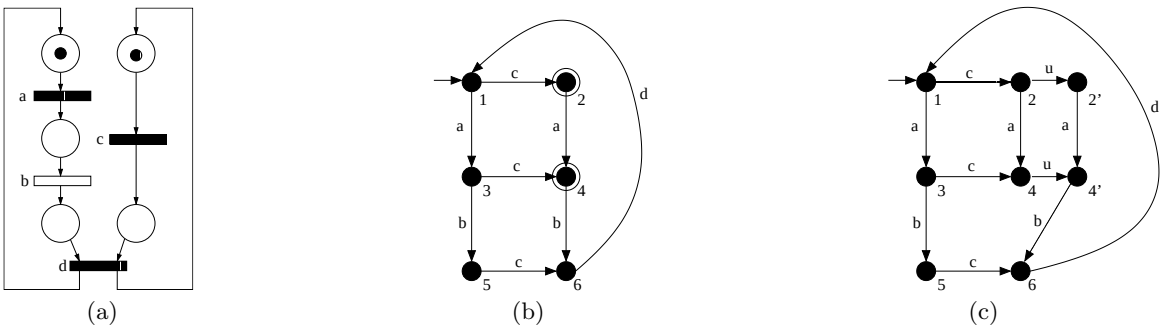


Fig. 1. Example for an event insertion. (a) A Petri net and its reachability graph in (b). The two marked states are the region R where the new event written u will be inserted. (c) The reachability graph with the inserted event u. Transitions entering R (e.g. $1 \xrightarrow{c} 2$), are not duplicated, transitions within R ($2 \xrightarrow{a} 4$) are duplicated and transitions leaving R ($4' \xrightarrow{b} 6$) only leave duplicated states. The marking relation is $\mathcal{M} = \{(1,1), (2,2), (2,2'), (3,3), (4,4), (4,4'), \ldots\}$.

It is not hard to see that N' as above is deterministic again. We first note some properties.

Lemma 8. *Let N be an STG and N' be obtained from N by event insertion of $u\pm$ at region R. Let $(M_1, M_2) \in \mathcal{M}$ and $a \in Sig_N$.*

1. *If $M_2 = M_1'$, then $M_1[a\pm\rangle\rangle\hat{M}_1$ in N implies $M_2[a\pm\rangle\rangle\hat{M}_2$ in N' with $(\hat{M}_1, \hat{M}_2) \in \mathcal{M}$.*
2. *$M_2[a\pm\rangle\rangle\hat{M}_2$ in N' implies $M_1[a\pm\rangle\rangle\hat{M}_1$ in N with $(\hat{M}_1, \hat{M}_2) \in \mathcal{M}$.*

Proof. 1. $M_2 = M_1'$ implies $M_1 \in R$ and by Definition 7.2,3 $M_2[a\pm\rangle\rangle\hat{M}_2$ in N' with $(\hat{M}_1, \hat{M}_2)$, where we have $\hat{M}_2 = \hat{M}_1'$ if case 2 is applicable and $\hat{M}_2 = \hat{M}_1$ if case 3 is applicable.

2. The reasoning is similar. □

The following result explains the definition of an input-proper event insertion and why we speak of speed-independent CSC-solving.

Proposition 9. *Let N be an input proper STG and let N' be obtained by the insertion of $u\pm$ at R. Then N' is input proper if and only if the insertion is.*

Proof. Assume the insertion is not input proper because of $M_1[a\pm\rangle\rangle M_2$; then we have in N': $M_1[u\pm\rangle\rangle M_1'$ due to Definition 7.1 and $\neg M_1[a\pm\rangle\rangle$ and $M_1'[a\pm\rangle\rangle M_2$ due to 3.

Vice versa, assume that N' is not input proper because of $M_1[v\pm\rangle\rangle M_2$, $\neg M_1[a\pm\rangle\rangle$ and $M_2[a\pm\rangle\rangle M_3$ for some $a \in In$. If $v\pm$ is the newly inserted $u\pm$, then $M_2 = M_1'$; we cannot have $M_2[a\pm\rangle\rangle M_3$ due to Definition 7.2 because then $M_1[a\pm\rangle\rangle$, and thus we must have $M_2[a\pm\rangle\rangle M_3$ due to Definition 7.3, i.e. the insertion is not input proper because of $M_1[a\pm\rangle\rangle M_3$ in N.

Otherwise, there are $\hat{M}_1$ and $\hat{M}_2$ with $(\hat{M}_1, M_1) \in \mathcal{M}$, $(\hat{M}_2, M_2) \in \mathcal{M}$, $\hat{M}_1[v\pm\rangle\rangle\hat{M}_2$ in N and $\hat{M}_2[a\pm\rangle\rangle$ in N by Lemma 8.2. Since N is input proper, this implies $\hat{M}_1[a\pm\rangle\rangle$; since $\neg M_1[a\pm\rangle\rangle$ in N', this in turn implies $\hat{M}_1 = M_1$ by Lemma 8.1. Thus, $\hat{M}_1$ has "lost" an $a\pm$-transition during the event insertion; this can only be due to Definition 7.3, and also in this case, the insertion is not input proper. □

We come to the main result of this section.

Theorem 10 (Correctness of CSC solving). *Let N be an STG and N' be obtained from N by speed-independent CSC-solving; then N' is a correct implementation of N.*

Proof. N' is obtained from N by a sequence of input proper event insertions. It suffices to show the claim for one such insertion, and then the theorem follows from Theorem 3.2. Thus, assume that N' is obtained from N by the input-proper insertion of $u\pm$ at R, with $\mathcal{M}$ being the corresponding marking relation.

We obviously have $In_N = In_{N'}$, $Out_N = Out_{N'}$. We will show that $\mathcal{M}$ is an STG-bisimulation for N and N'.

(1): Fulfilled by definition of event insertion.

(2): For this part observe that (C2) is trivially fulfilled, because we consider only one component. Let now $(M_1, M_2) \in \mathcal{M}$.

For (N1)–(N3), we only have to consider $M_2 = M_1$ due to Lemma 8.1. If $a \in In_N$ and $M_1[a\pm\rangle\rangle\hat{M}_1$ in N, then Definition 7.3 cannot be applicable since the insertion is input proper, hence $M_1[a\pm\rangle\rangle\hat{M}_1$ in N' as well with $(\hat{M}_1, \hat{M}_1) \in \mathcal{M}$ and (N1) follows.

Now let $x \in Loc_N$ and $M_1[x\pm\rangle\rangle\hat{M}_1$ in N. Then we have in N' that $M_1[x\pm\rangle\rangle\hat{M}_1$ if $M_1 \notin R$ or $M_2 \in R$ and $M_1[u \pm x\pm\rangle\rangle\hat{M}_1$ otherwise; obviously $(\hat{M}_1, \hat{M}_1) \in \mathcal{M}$ and (N2) and (N3) follow.

(C1/C3): Let $x \in Loc_{N'}$ and $M_2[x\pm\rangle\rangle\hat{M}_2$ in N'. If $x\pm$ is not the inserted $u\pm$, Lemma 8.2 implies $M_1[x\pm\rangle\rangle\hat{M}_1$ and $(\hat{M}_1, \hat{M}_2) \in \mathcal{M}$. Otherwise, we have $M_2 = M_1$, $\hat{M}_2 = M_1'$ and $(M_1, \hat{M}_2) \in \mathcal{M}$. □

Now we can conclude that speed-independent CSC-solving can be combined with decomposition. For this, we have to apply Theorems 3.1 and 10 to each component; observe that the crucial first condition on H_K in 3.1 is satisfied since $H_K = \emptyset$ and event insertion does not change the sets of output and of input signals.

Corollary 11. *Let $(C_i)_{i\in I}$ be a correct decomposition of N when hiding H, and let C_i' be obtained from C_i by speed-independent CSC-solving for all $i \in I$. Then $(C_i')_{i\in I}$ is a correct decomposition of N when hiding H.*

5 Correctness of an ILP Approach to Decomposition

In this section we will show that the decomposition method of Carmona and Cortadella [Car03,CC03], which has not been proven correct so far, yields components which are a correct decomposition according to our definition. For this method, it is assumed that an STG with CSC is given, where CSC can also be achieved by modifications on the STG-level, i.e. without considering the reachability graph. (It can also be given due to a suitable translation from a description in a high-level language to STGs as in [YOM04]). As explained at the end of Section 2, we can assume that there are no internal signals.

The method of [Car03,CC03] works roughly as follows. Starting with a deterministic STG N that already has CSC, for every output signal x a CSC support is determined; this is a set of signals, which guarantees CSC for x. Here is the formal definition:

Definition 12 (CSC Support). Let N be an STG and $S \subseteq Sig_N$.

1. Let $v \in (Sig_N\pm)^*$. $code\_change(S, v)$ is defined as the vector over S, which maps every $s \in S$ to the difference between the numbers of $s+$ and of $s-$ in v.
2. S is called *CSC support for the output signal* x if, for all reachable markings M_1, M_2 with $M_1[v\rangle\rangle M_2$ and $code\_change(S, v) = \mathbf{0}$ for some $v \in (Sig_N\pm)^*$, M_1 enables x iff M_2 does. □

A sufficient condition for being a CSC support used in the algorithm is that some integer linear programming (ILP) problem is infeasible. The algorithm starts for every output x with the set including the so-called *syntactical triggers* of x and x itself, and iteratively improves it – mostly by adding additional signals – until it is a CSC support for x; since the original STG has CSC, this algorithm is always successful.

After that, for every output signal the original STG is *projected* onto the corresponding CSC support: the other signals are considered as dummies, and these dummies and redundant places are removed as far as possible much as in our decomposition algorithm. If the resulting component still contains dummies, then [priv. comm.]: the reachability graph is generated and viewed as a finite automaton with dummies regarded as the empty word. Now the automaton is made deterministic with well-known methods, which in particular remove all λ-labelled edges. Finally, we can regard this automaton as an STG again, which e.g. has the edges of the automaton as transitions.

The projection part is similar to our algorithm, the difference is where backtracking is performed: the method of [Car03,CC03] uses some form of backtracking when determining the CSC support as described above — our algorithm uses backtracking when the contraction of a dummy signal is not possible.

An advantage of the method of [Car03,CC03] is that the components have CSC. Actually, the defining condition for a CSC support is slightly too weak to guarantee CSC in all cases,[2] but in most practical cases CSC holds, the condition and the corresponding ILP problem could easily be corrected, and most of all the given condition is sufficient for the proof of Theorem 13.

[2] The condition should consider all markings with the same state vector for signals in S, and not only those where one is reachable from the other.

For this proof, we need the following properties of the components $(C_i)_{i\in I}$ produced by the CSC-support algorithm:

1. Every component is deterministic.
2. The signals of every C_i are a CSC support of the only output signal.
3. $\forall i \in I : L(C_i) = L(N)\downarrow_i$

In the last item, $L(N)\downarrow_i$ denotes the projection of $L(N)$ onto the signals of C_i. We can now prove that $(C_i)_{i\in I}$ is a correct decomposition by our definition.

In the last item, $L(N)\downarrow_i$ denotes the projection of $L(N)$ onto the signals of C_i; note that this item is not equivalent to $L(||_{i\in I}C_I) = L(N)$; , see e.g. Figure 2 taken from [VW02] We can now prove that $(C_i)_{i\in I}$ is a correct decomposition by our definition.

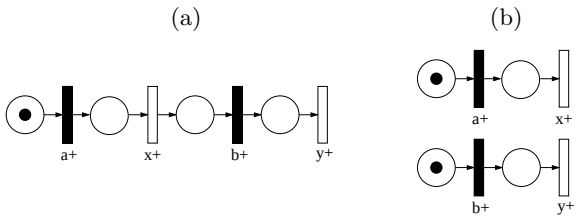


Fig. 2. (a) Original STG N (b) Decomposition into two components. In the second component and also in the parallel composition – but not in N – the input signal $b+$ can occur at the initial marking yielding a marking M. But this is not a problem since no environment suitable for N will fire $b+$. Correspondingly, the marking M will not appear in the STG-bisimulation corresponding to this decomposition.

Theorem 13 (Correctness of the CSC-support algorithm). *Let N be an STG and $(C_i)_{i\in I}$ be given as above. Then, $(C_i)_{i\in I}$ is correct w.r.t. N.*

Proof. Let $C = ||_{i\in I}C_i$. We define a relation $\mathcal{B}$ between the markings of N and C by

$$(M, (M_i)_{i\in I}) \in \mathcal{B} \Leftrightarrow \exists w : M_N[w\rangle\rangle M \wedge \forall i \in I : M_{C_i}[w\downarrow_i\rangle\rangle M_i$$

we will show that $\mathcal{B}$ is an STG-bisimulation.

(1): Obviously fulfilled for $w = \lambda$.

(2): Let $(M, (M_i)_{i\in I}) \in \mathcal{B}$. Therefore $\exists w : M_N[w\rangle\rangle M \wedge \forall i \in I : M_{C_i}[w\downarrow_i\rangle\rangle M_i$.

(N1): Let $a \in In_N$ and $M[a\pm\rangle\rangle\hat{M}$. This implies $wa\pm \in L(N)$ and therefore $\forall i \in I : (wa\pm)\downarrow_i \in L(C_i)$. If $a \notin In_C$ we are done, otherwise it follows from the determinism of the components that every C_i with $a \in In_i$ can fire $a\pm$: there is only one transition sequence v with $l(v) = w\downarrow_i$ and one sequence v' with $l(v') = w\downarrow_i a\pm$, obviously v is a prefix of v' and reaches M_i, and therefore $M_i[a\pm\rangle\rangle\hat{M}_i$.

This holds for every component with $a \in In_i$ and therefore $(M_i)_{i\in I}[a\pm\rangle\rangle(\hat{M}_i)_{i\in I}$ where $\hat{M}_i = M_i$ if $a \notin In_i$, and by definition of $\mathcal{B}$ we get $(\hat{M}, (\hat{M}_i)_{i\in I}) \in \mathcal{B}$.

(N2): Analogous to (N1).

(C2): Let $x \in Out_j$ and $M_j[x\pm\rangle\rangle$. Therefore, $w\downarrow_j x\pm \in L(C_j) = L(N)\downarrow_j$ which implies that $M[vx\pm\rangle\rangle$ for some $v \in ((Sig_N \setminus Sig_j)\pm)^*$ by determinism of N. Obviously, $code\_change(Sig_j, v) = 0$ and since Sig_j is a CSC support for x, we get $M[x\pm\rangle\rangle$. Applying (N2) yields the desired result: $(M_i)_{i\in I}[x\pm\rangle\rangle$.

(C1): Let $x \in Out_C$ and $(M_i)_{i\in I}[x\pm\rangle\rangle(\hat{M}_i)_{i\in I}$. If $x\pm$ is produced by component j, we get $M_j[x\pm\rangle\rangle$; then our considerations for (C2) imply $M[x\pm\rangle\rangle\hat{M}$. By definition of $\mathcal{B}$: $(\hat{M}, (\hat{M}_i)_{i\in I}) \in \mathcal{B}$.

(N3,C3): Not relevant. □

6 Conclusion

We have generalised the correctness definition for decompositions from [VW02,VK04] to STGs with internal signals and proven that speed-independent CSC-solving as performed by Petrify is correct. We have shown that the new correctness is preserved in a top-down decomposition, and this result has a number of consequences: now we can use step-wise decomposition in the algorithm of [VW02,VK04] to improve efficiency, and we know that this algorithm in combination with speed-independent CSC-solving gives correct results. Applying the correctness definition to compare two STGs, we get an implementation relation, and consequences of our result are that this is a preorder and, with a small restriction, a precongruence for parallel composition, relabelling and hiding.

As another application of the correctness definition, we have shown that a decomposition method based on integer linear programming [CC03] is correct. It remains an open problem whether a related method in [YOM04] is correct: while the first method checks on the original STG to be decomposed whether a set of signals is a CSC-support and in the positive case removes the other signals, the related method removes some signals and checks CSC on the remaining STG; this is in general not sufficient, but it might be sufficient under the specific circumstances of the algorithm in [YOM04].

For a further validation of our correctness definition, it would be interesting to compare the resp. implementation relation with another one derived from the notion of I/O-compatibility in [CC02]. We think that the derived implementation relation holds whenever our implementation relation holds, but the reverse direction can only be true under suitable restrictions; the latter still have to be identified, but we expect that they will shed some light on the conceptual ideas behind I/O-compatibility and our correctness.

References

[Car03] Josep Carmona. *Structural Methods for the Synthesis of Well-Formed Concurrent Specifications*. PhD thesis, Universitat Politècnica de Catalunya, 2003.

[CC02] J. Carmona and J. Cortadella. Input/output compatibility of reactive systems. In *Formal Methods in Computer-Aided Design, FMCAD 2002, Portland, USA*, Lect. Notes Comp. Sci. 2517, pages 360–377. Springer, 2002.

[CC03] J. Carmona and J. Cortadella. ILP models for the synthesis of asynchronous control circuits. In *Proc. of the IEEE/ACM International Conference on Computer Aided Design*, pages 818–825, 2003.

[Chu87] T.-A. Chu. *Synthesis of Self-Timed VLSI Circuits from Graph-Theoretic Specifications*. PhD thesis, MIT, 1987.

[CKK+97] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Trans. Information and Systems*, E80-D, 3:315–325, 1997.

[CKK+02] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. *Logic Synthesis of Asynchronous Controllers and Interfaces*. Springer, 2002.

[Dil88] D. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent circuits*. MIT Press, Cambridge, 1988.

[Ebe92] J. Ebergen. Arbiters: an exercise in specifying and decomposing asynchronously communicating components. *Sci. of Computer Programming*, 18:223–245, 1992.

[KKT93] A. Kondratyev, M. Kishinevsky, and A. Taubin. Synthesis method in self-timed design. Decompositional approach. In *IEEE Int. Conf. VLSI and CAD*, pages 324–327, 1993.

[Pet81] J.L. Peterson. *Petri Net Theory*. Prentice-Hall, 1981.

[VK04] W. Vogler and B. Kangsah. Improved decomposition of signal transition graphs. Technical Report 2004-8, University of Augsburg, http://www.Informatik.Uni-Augsburg.DE/skripts/techreports/, 2004.

[VW02] W. Vogler and R. Wollowski. Decomposition in asynchronous circuit design. In J. Cortadella et al., editors, *Concurrency and Hardware Design*, Lect. Notes Comp. Sci. 2549, 152 – 190. Springer, 2002.

[YKK+96] A. Yakovlev, M. Kishinevsky, A. Kondratyev, L. Lavagno, and M. Pietkiewicz-Koutny. On the models for asynchronous circuit behaviour with or causality. *Formal Methods in System Design*, 9:189–233, 1996.

[YOM04] T. Yoneda, H. Onda, and C. Myers. Synthesis of speed independent circuits based on decomposition. In *ASYNC 2004*, pages 135–145. IEEE, 2004.