Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 263 (2010) 197–210
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Reachability in Tree-Like Component Systems is PSPACE-Complete

Mila Majster-Cederbaum[1] Nils Semmelrock [2]

Department of Computer Science
University Mannheim
Mannheim, Germany

Abstract

The reachability problem in component systems is PSPACE-complete. We show here that even the reachability problem in the subclass of component systems with "tree-like" communication is PSPACE-complete. For this purpose we reduce the question if a Quantified Boolean Formula (QBF) is true to the reachability problem in "tree-like" component systems.

Keywords: Component-Based Modeling, Architectural Constraints, Reachability, PSPACE-Completeness

1 Introduction

In component-based modeling techniques the size of the global state space of a system is in the worst case exponential in the number of its components. This problem is often referred to as the effect of state space explosion. Thus, checking properties of a component-based system by exploring the state space very quickly becomes inefficient. Here we explore the complexity-theoretical classification of reachability.

As a formal model for component-based systems we consider here interaction systems [8], a formalism for component-based modeling which offers in general an arbitrary degree of synchronization. Reachability in general interaction systems was proved to be PSPACE-complete [15] similar to results in 1-safe Petri nets [6].

Tree-like component systems are component systems where the communication structure forms a tree. This is an important class of systems which has been early studied e.g. in [9,5] and more recently e.g. in [4,12].

[1] Email:mcb@informatik.uni-mannheim.de

[2] Email:nsemmelr@informatik.uni-mannheim.de

doi:10.1016/j.entcs.2010.05.012

Here we show that even in the subclass of tree-like interaction systems the reachability problem (and therefore proving deadlock-freedom as well) is PSPACE-complete. We also sketch that deciding progress in tree-like interaction systems is PSPACE-complete.

2 Interaction Systems

Interaction systems are a component-based formalism, i.e. a system is composed of subsystems called components. Components are put together by some kind of glue-code. Interaction systems are defined in two layers. The first layer, the interaction model, describes the interfaces of the components and the glue-code of a system by connecting the interfaces of the components. The second layer describes the behavior of the components, which is here given in form of labeled transition systems.

Definition 2.1 Let $K = \{1, \ldots, n\}$ be a finite set of components. For each $i \in K$ let A_i be a finite set of ports such that $\underset{i,j \in K}{\forall} i \neq j \Rightarrow A_i \cap A_j = \emptyset$. An **interaction model** is a tuple $IM := (K, \{A_i\}_{i \in K}, C)$, where C is a set such that

a) $\forall c \in C : c \subseteq \bigcup_{i \in K} A_i$,

b) $\forall c \in C \forall i \in K : |c \cap A_i| \leq 1$ and

c) $\bigcup_{c \in C} c = \bigcup_{i \in K} A_i$.

The elements of C are called **connectors**. Let for $c \in C$ and $i \in K$ $i(c) := c \cap A_i$ be the set of ports of i which participate in c, i.e. $|i(c)| \leq 1$.

Let $T_i = (Q_i, A_i, \rightarrow_i, q_i^0)$ for $i \in K$ be a labeled transition system with a finite set of states Q_i, a transition relation $\rightarrow_i \subseteq Q_i \times A_i \times Q_i$ and an initial state $q_i^0 \in Q_i$. A transition system T_i for $i \in K$ models the behavior of component i. We will write $q_i \xrightarrow{a_i}_i q_i'$ instead of $(q_i, a_i, q_i') \in \rightarrow_i$.

Definition 2.2 An **interaction system** is a tuple $Sys := (IM, \{T_i\}_{i \in K})$. The behavior of Sys is given by the transition system
$T = (Q_{Sys}, C, \rightarrow, q^0)$ where

a) $Q_{Sys} := \prod_{i \in K} Q_i$ is the state space,

b) $q^0 = (q_1^0, \ldots, q_n^0) \in Q_{Sys}$ is the initial state and

c) $\rightarrow \subseteq Q_{Sys} \times C \times Q_{Sys}$ is the transition relation with $q \xrightarrow{c} q'$ iff for all $i \in K$ $q_i \xrightarrow{i(c)}_i q_i'$ if $i(c) \neq \emptyset$ and $q_i = q_i'$ otherwise.

Definition 2.3 Let Sys be an interaction system and $T = (Q_{Sys}, C, \rightarrow, q^0)$ the associated global transition system. A global state $q \in Q_{Sys}$ is called **reachable** iff there is a path that leads from the initial state q^0 to q in T.

As mentioned we focus on a structural constraint on interaction systems. More precisely we look at the important class of interaction systems such that the glue-code describes a tree-like communication pattern, i.e. components never form a cycle with respect to their connectors.

A tree-like communication structure induces an important class of component-based systems. Many interesting systems belong to this class, e.g. hierarchical systems or networks built by a master-slave operator [9]. For these reasons, this class of component systems has been studied intensely e.g. in [3,4,5,12].

Definition 2.4 Let $IM = (K, \{A_i\}_{i\in K}, C)$ be an interaction model. The **interaction graph** $G^* = (K, E)$ of IM is an undirected graph with $\{i, j\} \in E$ iff there is a connector $c \in C$ with $i(c) \neq \emptyset$ and $j(c) \neq \emptyset$.

An interaction model IM is called **tree-like** iff the associated interaction graph G^* is a tree. An interaction system Sys is called tree-like if its associated interaction model is tree-like.

Remark 2.5 Note, that a tree-like interaction system with a set C of connectors implies that on all $c \in C$ $|c| \leq 2$.

Example 2.6 Consider the dining philosophers problem with n philosophers and n forks. The philosophers, respectively the forks, are labeled with $0, \ldots, n-1$. The philosophers are placed in order around a table such that between philosopher i and $i+1$ (we assume modulo n arithmetics) fork i is placed. We construct an interaction system such that each philosopher and each fork corresponds to a component. Let $i \in \{0, \ldots, n-1\}$ then philosopher i is modeled by component $Phil_i$ with the set of ports $A_{Phil_i} := \{t\_left_i, t\_right_i, p\_left_i, p\_right_i\}$ such that the ports are modeling, from left to right: "take left fork", "take right fork", "put left fork back on the table" and "put right fork back on the table". Fork i is modeled by component $Fork_i$ with the set of ports $A_{Fork_i} := \{take_i, release_i\}$.

The following connectors describe the synchronization between the philosophers and the forks, corresponding to the seating order.

$$\begin{aligned} take\_left_i &:= \{t\_left_i, take_i\} & take\_right_i &:= \{t\_right_i, take_{i-1}\} \\ put\_left_i &:= \{p\_left_i, release_i\} & put\_right_i &= \{p\_right_i, release_{i-1}\} \end{aligned}$$

Consider the problem for $n = 3$ philosophers, then the set K of components is given by $K = \{Phil_0, Phil_1, Phil_2, Fork_0, Fork_1, Fork_2\}$ and the set C of connectors by $C := \{take\_left_i, take\_right_i, put\_left_i, put\_right_i | i = 0, \ldots, 2\}$. The interaction model is given by $IM = (K, \{A_i\}_{i\in K}, C)$. The corresponding interaction graph G^* for IM is given in Figure 1 and forms a cycle.

3 QBF Reduction to Tree-Like Interaction Systems

We will show that reachability in tree-like interaction systems is PSPACE-complete. The PSPACE-hardness will be proved by a reduction from QBF [7]. PSPACE-hardness of reachability in general interaction systems was shown by a reduction from reachability in 1-safe Petri nets [15]. To show the PSPACE-hardness of reachability in 1-safe Petri nets a reduction from QBF was used [6].

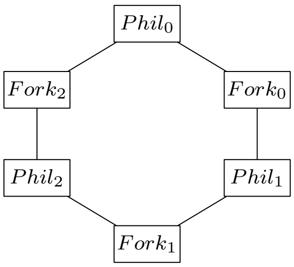

Fig. 1. Interaction graph G^* for the interaction model IM.

3.1 Reduction

Reachability in tree-like interaction systems is in PSPACE. Given a tree-like interaction system and a global state q one can guess a sequence of connectors (because PSPACE=NPSPACE) and check in linear space if it leads from the initial state q^0 to q. At any time we story exactly one global state from which we guess a valid successor state. To prove the PSPACE-hardness we present a reduction from the validity problem for Quantified Boolean Formulas (QBF) to the reachability problem in tree-like interaction systems.

3.1.1 QBF

An instance of QBF [7] is given as a well-formed quantified Boolean formula where its variables $x_1, \ldots, x_n$ are all bound and distinct. Without loss of generality we look at QBF instances over the grammar

$$P ::= x|\neg P|P \wedge P|\exists x.P.$$

In the following we will assume that a QBF formula is built over this grammar. Let H be a QBF then the question is if H is true. The language TQBF is defined as the set of true QBF instances and is well known to be PSPACE-complete.

There is a straightforward, recursive algorithm called *eval* to determine whether a QBF P given over the grammar above is in TQBF.

Algorithm 1

```
1 $eval(P)$
2     $if(P = x)$
3         $return\ value(x)$
4     $if(P = \neg P')$
5         $return\ \neg eval(P')$
6     $if(P = P' \wedge P'')$
7         $return\ eval(P') \wedge eval(P'')$
8     $//\ P = \exists x.P'$ is the only remaining possibility
9     $return\ eval(P'_{x=true}) \vee eval(P'_{x=false})$
```

In line 9 $P'_{x=true}$ denotes the subformula P' with *true* assigned to the variable x. In line 3 $value(x)$ returns the truth value that is assigned to x. This is possible because every variable x in H is bound by an existential quantifier and therefore a truth value is assigned in line 9. Obviously, $H \in TQBF \Leftrightarrow eval(H) = true$.

3.1.2 RIST

Let IST be the class of tree-like interaction systems. For $Sys \in IST$ let $Reach(Sys) \subseteq Q_{Sys}$ be the set of reachable states. Let

$$RIST := \bigcup_{Sys \in IST} (\{Sys\} \times Q_{Sys}) .$$

For $(Sys, q) \in RIST$ we want to decide if q is reachable in Sys. Let $TRIST \subseteq RIST$ be the set of true $RIST$ instances, i.e.

$$TRIST = \bigcup_{Sys \in IST} (\{Sys\} \times Reach(Sys)) .$$

In the following we will introduce for a QBF H a tree-like interaction system Sys_H and a global state q^t such that

i) $H \in TQBF \Leftrightarrow (Sys_H, q^t) \in TRIST$ and

ii) the size of Sys_H is polynomial in the size of H.

The idea for the construction of Sys_H can be sketched as follows: the interaction system basically simulates the evaluation of the formula H, as in algorithm *eval* (see Algorithm 1), based on the syntax tree of H. The subformulas of H are the components of the system, and the interaction model describes the propagation of truth values between the nodes of the syntax tree.

Example 3.1 Consider the formula $H = \neg\exists x_1.(x_1 \wedge \neg x_1)$. The associated interaction graph G^*_H of IM_H is given in Figure 2 where components with highlighted frames denote components that do not model subformulas of H. IM_H is constructed accordingly to the following reduction.

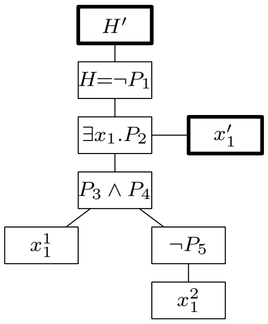

Fig. 2. Interaction graph G^*_H of IM_H.

We now describe in detail how Sys_H is constructed:

3.1.3 Components

Let H be a QBF with variables $x_1, \ldots, x_n$ and $K_2 = \{x'_i | x_i \text{ is a variable in } H\}$. The set of components K_2 is needed to avoid cycles in the interaction graph. Generally, there may be several occurrences of a variable x_i in H. Let x_i occur k_i times for $i = 1, \ldots, n$ as a subformula in H, then we assume that the jth occurrence of variable x_i is renamed in H as x_i^j for $j \in \{1, \ldots, k_i\}$.

Let $K_H = K_1 \cup K_2 \cup \{H'\}$ be a set of components such that $K_1 = \{P | P \text{ is a subformula of } H\}$. The component H' is an auxiliary component which simplifies the definition of the behavior of the components in K_1.

Given a truth assignment to the variables, subformulas are assigned true or false. Therefore, when we mention an assignment to a component in $K_1 \cup K_2$ we refer to the assignment of the subformula that is modeled by this component.

In the following we will give the port sets of the components. Many ports, in different components, serve the same purpose and only differ in their subscripts. Once such a port is introduced it will not be explicitly explained again.

Port sets of components modeling variables

For $i = 1, \ldots, n$ and $j = 1, \ldots, k_i$ the component $P = x_i^j \in K_1$ represents the jth occurrence of variable x_i in H. The set A_P of ports is given by

$$A_P := \{\mathfrak{a}_P, t_P, f_P, r_Pt\} \cup \{r_Px_lt, r_Px_lf | l = 1, \ldots, n\}.$$

- $\mathfrak{a}_P$ abbreviates “activate P” and starts the evaluation of P.
- t_P respectively f_P confirm that currently true respectively false is assigned to P.
- r_Px_lt abbreviates “P receives instruction to set x_l true”. If $l = i$ then true is assigned to P. For $i \neq l$ r_Px_lt has no effect on P. The same applies to r_Px_lf setting x_l to false.
- r_Pt has the function to assign true to P.

Port sets for negated formulas

A component modeling a negation, i.e. a subformula of the form $P = \neg P_1$ has the following set of ports A_P

$$A_P := \{\mathfrak{e}_P^1, \mathfrak{a}_P, sub_P^1t, sub_P^1f, t_P, f_P, r_Pt, s_P^1t\} \cup \{r_Px_lt, r_Px_lf, s_P^1x_lt, s_P^1x_lf | l = 1, \ldots, n\}.$$

- $\mathfrak{e}_P^1$ abbreviates “evaluate the first subformula of P” and evaluates the subformula P_1.
- sub_P^1t (abbreviates “subformula 1 is true”) respectively sub_P^1f affirm that P_1 was evaluated true respectively false.
- According to the structure of a negation f_P (abbreviates “$P = \neg P_1$ is false”) is enabled if P_1 was evaluated true. Conversely t_P is enabled if P_1 was evaluated false.
- As above r_Px_lt models that P receives the instruction to set x_l true. On the other hand $s_P^1x_lt$ (“set x_l true in the first subformula of P”) models that P itself sends the instruction to set x_l to true to P_1. The same applies to $s_P^1x_lf$ if x_l needs to be set to false.

- $s^1_P t$ has the function to set the truth assignment of P's subformula P_1 to true.

Port sets for conjunctions
The component that models a conjunction, i.e. a subformula of the form $P = P_1 \wedge P_2$ has the set of ports

$$\begin{aligned} A_P := &\{\mathfrak{a}_P, \mathfrak{e}^1_P, \mathfrak{e}^2_P, sub^1_P t, sub^1_P f, sub^2_P t, sub^2_P f, t_P, f_P, r_P t, s^1_P t, s^2_P t\} \cup \\ &\{r_P x_l t, r_P x_l f, s^1_P x_l t, s^1_P x_l f, s^2_P x_l t, s^2_P x_l f | l = 1, \ldots, n\}. \end{aligned}$$

This is the only formula that has two direct subformulas. $P = P_1 \wedge P_2$ needs to evaluate P_1 and P_2, therefore there are ports $\mathfrak{e}^1_P$ and $\mathfrak{e}^2_P$. Similarly there are $sub^1_P t$, $sub^1_P f$, $sub^2_P t$, $sub^2_P f$ for actually receiving the truth values of P_1 and P_2. Likewise, $s^1_P x_l t$ and $s^2_P x_l t$ model that P needs to set x_l to true in its first and second subformula and respectively $s^1_P x_l f$ and $s^2_P x_l f$ to false.

Port sets for existentially quantified formulas and associated component x'_i
In the interaction system Sys_H a component for a subformula of the form $P = \exists x_i.P_1$ with $i = 1, \ldots, n$ needs to have access to the current truth assignment of the variable x_i. For this purpose the set of components K_2 was introduced. Let x_i be the variable that is quantified by the subformula $P = \exists x_i.P_1$. The component x'_i models the truth assignment of x_i. The set of ports $A_{x'_i}$ is given by

$$A_{x'_i} := \{rx_i t, rx_i f, t_{x_i}, f_{x_i}\}.$$

t_{x_i} respectively f_{x_i} affirm that the current state of x'_i is true respectively false. $rx_i t$ assigns x'_i true. Analogously $rx_i f$ switches the assignment to false.

The port set A_P for $P = \exists x_i.P_1$ is given by

$$\begin{aligned} A_P := &\{\mathfrak{a}_P, \mathfrak{e}^1_P, sub^1_P t, sub^1_P f, t_P, f_P, x_i t, x_i f, sx_i t, sx_i f, r_P t, s^1_P t\} \cup \\ &\{r_P x_l t, r_P x_l f, s^1_P x_l t, s^1_P x_l f | l = 1, \ldots, n\}. \end{aligned}$$

$\mathfrak{a}_P$, $\mathfrak{e}^1_P$, $sub^1_P t$, $sub^1_P f$, t_P and f_P act similarly to the corresponding ports of the other components specified above. $x_i t$ confirms that true is assigned to x_i, and $sx_i t$ sets x_i to true if the current assignment is false. On the other hand $x_i f$ confirms that false is assigned to x_i, and $sx_i f$ assigns false to x_i if that is not the case.

Port set for the auxiliary component H'
Given the syntax tree for H, whose root is labeled H, H' can be interpreted as a direct dummy predecessor formula of H without any logical operator. The set of ports $A_{H'}$ is given by

$$A_{H'} := \{\mathfrak{e}^1_{H'}, sub^1_{H'} t, sub^1_{H'} f, s^1_{H'} t, end_{H'}\}.$$

All ports but $end_{H'}$ act exactly as the ports described above. It will be shown that the formula H is in TQBF iff the component associated with H is evaluated true, i.e. $sub^1_{H'} t$ can interact eventually. When the evaluation of the QBF H has been simulated, i.e. H' reached a state that represent the fact that H was evaluated true or false, then the port $end_{H'}$ becomes enabled. This only assures that the behavior of H' does not deadlock.

3.1.4 Connectors

We will now define a set C of connectors. Let $P \in K_1 \cup \{H'\}$ be a subformula which is not an occurrence of a variable. P can have one direct subformula which is P_1 or two direct subformulas P_1 and P_2. If P needs the truth value of P_k, $k \in \{1, 2\}$, to be evaluated then the evaluation in P_k needs to be activated. This is realized by the synchronization of $\mathfrak{e}_P^k$ and $\mathfrak{a}_{P_k}$. Furthermore P can ask P_k for its current truth value. These interactions are realized by

$$eval\_P \rightarrow P_k := \{\mathfrak{e}_P^k, \mathfrak{a}_{P_k}\} \qquad P\_ask\_P_k\_true := \{sub_P^k t, t_{P_k}\}$$
$$P\_ask\_P_k\_false := \{sub_P^k f, f_{P_k}\}$$

for $k \in \{1, 2\}$. These connectors already connect all components in $K_1 \cup \{H'\}$ and result in an interaction graph that is related to the syntax tree of the QBF H.

If P needs all occurrences of variable x_i to be set to true or false a direct interaction with the components that model these variables would lead to a cycle in the associated interaction graph. Therefore, P passes this information to its subformulas, i.e. $s_P^k x_i t$ in P has to synchronize with $r_{P_k} x_i t$ in P_k where P_k is a direct subformula of P. Let $i \in \{1, \ldots, n\}$. The following connectors, for $k \in \{1, 2\}$, realize the synchronizations needed to propagate the information to switch a variable.

$$set\_x_i\_true\_P \rightarrow P_k := \{s_P^k x_i t, r_{P_k} x_i t\}$$
$$set\_x_i\_false\_P \rightarrow P_k := \{s_P^k x_i f, r_{P_k} x_i f\}$$

If the QBF H is true, we need all components to be in one fixed state − this will be a state that models the assignment true. In fact, the component H' will observe if H is true and reach a fixed state. To assure that all components can reach a fixed state, a similar technique as above is used. A component can set the truth assignment of the components that represent its subformulas to true by the following connector for $k \in \{1, 2\}$.

$$set\_P_k\_true\_P \rightarrow P_k := \{s_P^k t, r_{P_k} t\}$$

Consider a subformula of the form $P = \exists x_i.P_1 \in K_1$ and the associated component $x_i' \in K_2$. The component representing P can assign x_i' the truth value true or false and can ask x_i' whether the current truth assignment is true or false. This is realized by

$$set\_x_i'\_true := \{sx_i t, rx_i t\} \qquad ask\_true_{x_i'} := \{x_i t, t_{x_i}\}$$
$$set\_x_i'\_false := \{sx_i f, rx_i f\} \qquad ask\_false_{x_i'} := \{x_i f, f_{x_i}\}$$

IF H' reaches a state that indicates that H was evaluated true or false, i.e. the simulation of the evaluation of H is finished, then the unary connector *evaluated* := $\{end_{H'}\}$ becomes enabled.

Let C be the set of connectors given by

$$\{eval\_P \to P_k, P\_ask\_P_k\_true, P\_ask\_P_k\_false | P \in K_1 \cup \{H'\} \text{ with succ. } P_k\} \cup$$
$$\{set\_x'_i\_true, set\_x'_i\_false, ask\_true_{x'_i}, ask\_false_{x'_i} | x'_i \in K_2\} \cup$$
$$\{set\_P_k\_true\_P \to P_k | P \in K_1 \cup \{H'\} \text{ with succ. } P_k\} \cup$$
$$\{set\_x_i\_true\_P \to P_k, set\_x_i\_false\_P \to P_k | P \in K_1 \text{with succ. } P_k, i \in \{1, \dots, n\}\} \cup$$
$$\{evaluated\}.$$

So far we have the interaction model $IM_H := (K_H, \{A_P\}_{P \in K_H}, C)$. This way any QBF formula H over the grammar, given above, can be mapped to an interaction model IM_H.

Remark 3.2 The interaction graph G^*_H, associated to IM_H, is a tree, as it is constructed along the syntax tree and augmented with the components H' and x'_i for $i = 1, \dots, n$ without forming cycles.

3.1.5 Local Behavior

The local behavior of the components is given by labeled transition systems. Every system has one state labeled t and one labeled f. These states model the fact that either true respectively false was assigned to this component or it was evaluated true respectively false. The initial state will be denoted by an ingoing arrow.

Figure 3(a) depicts the transition system of the component modeling the jth occurrence of variable x_i. Figure 3(b) gives the local behavior of a component $x'_i \in K_2$. The behavior of H' is given in 3(c). The transition systems for a variable x^j_i and a $x'_i \in K_2$ are self-explanatory. If in $T_{H'}$ the port $\mathfrak{e}^1_{H'}$ is performed, i.e. component H needs to be evaluated, then $T_{H'}$ waits to perform either $sub^1_{H'}t$ or $sub^1_{H'}f$. This ports can only be performed if T_H reaches its state labeled t respectively f. It will be shown that this indicates whether the associated QBF is true or false.

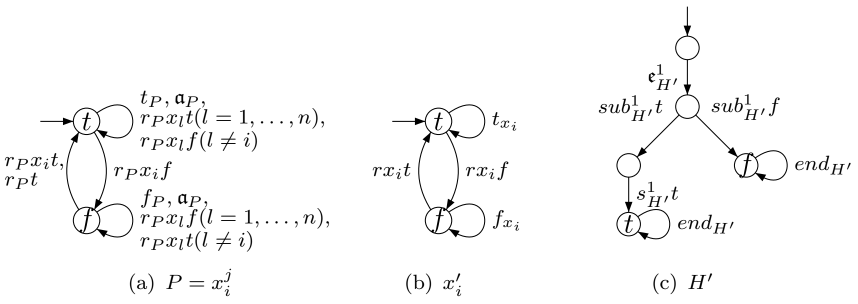

Fig. 3. Transition systems $T_{x^j_i}$ for a component x^j_i (a), $T_{x'_i}$ for x'_i (b) and $T_{H'}$ for the component H' (c).

In Figure 4 the transition system for a component of the form $P = \neg P_1$ is pictured. Note, that for better readability, the transition system in Figure 4(a) is not completely displayed. In system 4(a) the transitions and states pictured in

Figure 4(b) and 4(c) have to be included between the states labeled t and f for $l = 1, \ldots, n$.

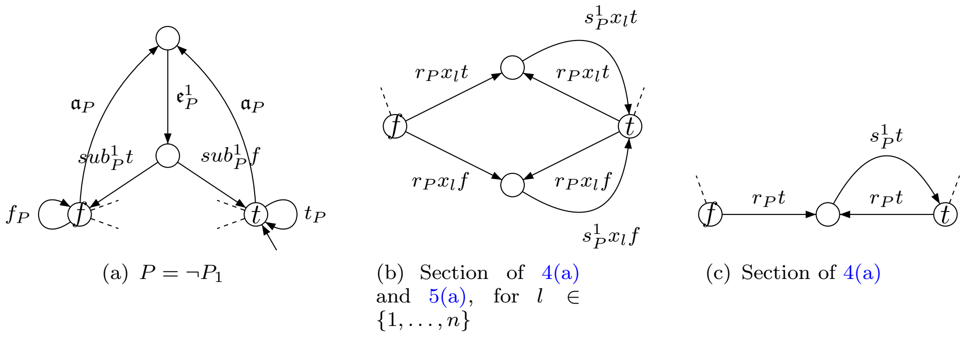

Fig. 4. Main section of the transition systems $T_{\neg P_1}$ (a), part of $T_{\neg P_1}$ for $l \in \{1, \ldots, n\}$ (b), part of $T_{\neg P_1}$ (c).

In Figure 5 the transition system for a component of the form $P = \exists x_i.P_1$ is pictured. For better readability, the transition system in Figure 5(a) is not completely displayed. In system 5(a) the transitions and states pictured in Figure 4(b) and 5(b) have to be included between the states labeled t and f for $l = 1, \ldots, n$.

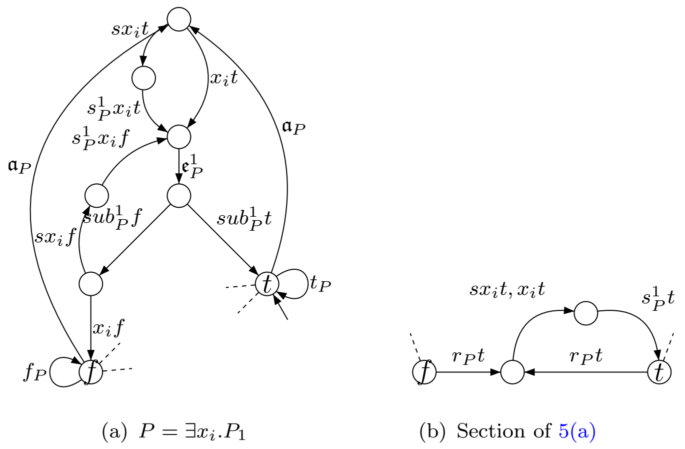

Fig. 5. Main sections of the transition system $T_{\exists x_i.P_1}$ (a), part of $T_{\exists x_i.P_1}$ (b).

In Figure 6 the transition system for a component of the form $P = P_1 \wedge P_2$ is pictured. Note, that the transition system in Figure 6(a) is not completely displayed. The transitions and states pictured in Figure 6(b) and 6(c) have to be included between the states labeled t and f for $l = 1, \ldots, n$.

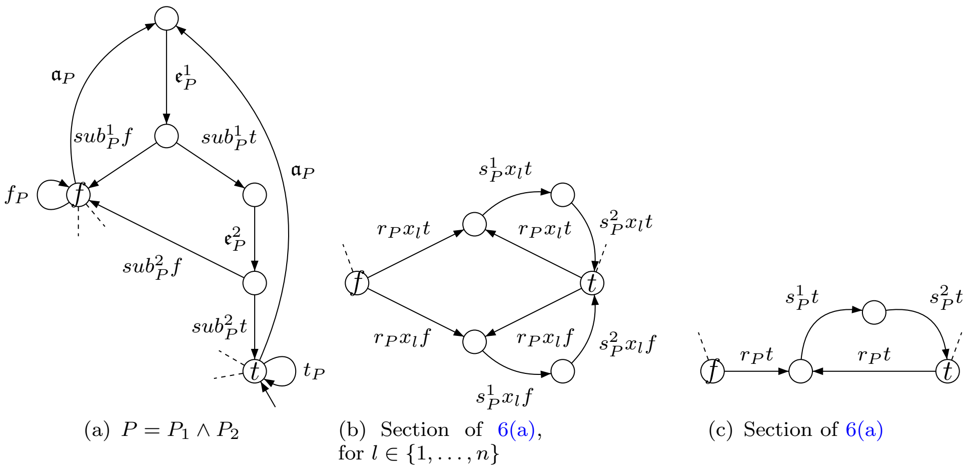

Fig. 6. Transition system $T_{P_1 \wedge P_2}$.

The resulting interaction system is denoted by $Sys_H := (IM_H, \{T_P\}_{P \in K_H})$.

Theorem 3.3 *Let H be a QBF over the grammar $P ::= x|\neg P|P \wedge P|\exists x.P$ and Sys_H the associated interaction system obtained from the reduction. Let q^t be the global state in which all components are in their state labeled t, then*

$$H \in TQBF \Leftrightarrow (Sys_H, q^t) \in TRIST.$$

The proof of Theorem 3.3 can be found in the Appendix of [14].

4 QBF Reduction to Progress in Tree-Like Interaction Systems

By minor modification of the reduction given above it is possible to show the PSPACE-completeness of the progress property in tree-like interaction systems. At first we give some definitions to introduce progress in interaction systems and then give an overview why it is PSPACE-complete to decide this property in tree-like interaction systems. In general interaction systems progress is PSPACE-complete [15], so progress in tree-like interaction systems is in PSPACE.

Definition 4.1 Let Sys be an interaction system and $T = (Q_{Sys}, C, \rightarrow, q^0)$ the associated global transition system. A global state $q \in Q_{Sys}$ is called a **deadlock** if no connector is enabled in q, i.e. there is no $c \in C$ and $q' \in Q_{Sys}$ such that $q \xrightarrow{c} q'$. A system Sys is **free of deadlocks** if there is no reachable state $q \in Q_{Sys}$ such that q is a deadlock.

Definition 4.2 Let Sys be a deadlock-free interaction system. A **run of** Sys is an infinite sequence σ

$$q^0 \xrightarrow{c_1} q^1 \xrightarrow{c_2} q^2 \dots,$$

with $q^l \in Q_{Sys}$ and $c_l \in C$ for $l \geq 1$.

Definition 4.3 Let Sys be a deadlock-free interaction system with components K. $k \in K$ **may progress** in Sys if for every run σ k participates infinitely often in σ.

An instance of the progress problem in interaction systems is given by a tuple (Sys, k) where Sys is a deadlock-free interaction system with components K and $k \in K$. The question is if k may progress in Sys.

We modify Sys_H as follows:

We introduce an additional component called *pro* with the set of ports $A_{pro} := \{t_{pro}\}$ and the behavior given by the transition system T_{pro} in Figure 7. The idea is to embed *pro* in Sys_H such that t_{pro} will participate infinitely often in every run σ iff H is true.

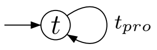

Fig. 7. Transition system T_{pro} for the component *pro*.

In addition we modify the component H' as follows. The set of ports $A_{H'}$ of the component H' is now given by

$$A_{H'} := \{\mathfrak{e}^1_{H'}, sub^1_{H'}t, sub^1_{H'}f, s^1_{H'}t, end\_true_{H'}, end\_false_{H'}\},$$

i.e. $end_{H'}$ is removed and the ports $end\_true_{H'}$ and $end\_false_{H'}$ are added. The modified behavior of H' is given by the transition system $T_{H'}$ in Figure 8.

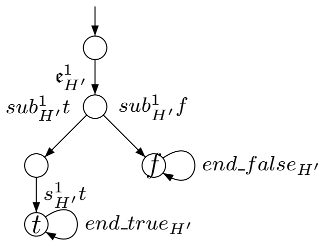

Fig. 8. Modified transition system $T_{H'}$ for the component H'.

In addition, the connector *evaluated* is removed von the set C of connectors, and the two following connectors are added.

$$evaluated\_true := \{end\_true_{H'}, t_{pro}\},$$
$$evaluated\_false := \{end\_false_{H'}\}.$$

It is easy to see that the connector *evaluated_true* is the only connector that is enabled if the state q^t is reached. In this case, *evaluated_true* will perform infinitely often, i.e. the component *pro* will participate infinitely often. Therefore the component *pro* may progress iff H is true.

5 Conclusion and Related Work

We investigated a complexity issue for component-based systems. In [6] the reachability in 1-safe Petri nets was proven to be PSPACE-complete and [15] used this result to show the PSPACE-completeness of the reachability problem in component-based systems. Here we restricted ourselves to tree-like systems and showed that even in this class deciding reachability is PSPACE-complete.

We conjecture that this result can be still strengthened such that it should be possible to show that even for the class of linear interaction systems, where the interaction graph forms a sequence of components, the reachability problem is PSPACE-complete. Given these complexity issues it makes sense to look for conditions that can be tested in polynomial time and guarantee a desired property.

For general component systems this is pursued e.g. in [1,2,10,11,13,16]. For tree-like component systems [3,4,5,12] have followed this approach and in particular established conditions that ensure deadlock-freedom.

References

[1] Robert Allen and David Garlan. A Formal Basis for Architectural Connection. *ACM Transactions on Software Engineering and Methodology*, 6:213–249, 1997.

[2] Paul Attie and Hana Chockler. Efficiently Verifiable Conditions for Deadlock-Freedom of Large Concurrent Programs. In *Proceedings of VMCAI'05*, LNCS 3385, pages 465–481, 2005.

[3] H. Baumeister, F. Hacklinger, R. Hennicker, A. Knapp, and M. Wirsing. A Component Model for Architectural Programming. In *Proceedings of FACS'05*, volume 160 of *ENTCS*, pages 75–96. Elsevier, 2006.

[4] Marco Bernardo, Paolo Ciancarini, and Lorenzo Donatiello. Architecting Families of Software Systems with Process Algebras. *ACM Trans. on Software Engineering and Methodology*, 11:386 – 426, October 2002.

[5] Stephen D. Brookes and A. W. Roscoe. Deadlock Analysis in Networks of Communicating Processes. *Distributed Computing*, 4:209–230, 1991.

[6] Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity Results for 1-safe Nets. In *Theoretical Computer Science*, pages 326–337. Springer-Verlag, 1995.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, January 1979.

[8] Gregor Gössler and Joseph Sifakis. Composition for Component-Based Modeling. *Sci. Comput. Program.*, 55(1-3):161–183, 2005.

[9] Charles A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International series in computer science. Prentice-Hall International, Englewood Cliffs, NJ [u.a.], 1985.

[10] Paola Inverardi and Sebastián Uchitel. Proving Deadlock Freedom in Component-Based Programming. In *FASE '01: Proceedings of the 4th International Conference on Fundamental Approaches to Software Engineering*, pages 60–75, London, UK, 2001. Springer-Verlag.

[11] M. Majster-Cederbaum, M. Martens, and C. Minnameier. A Polynomial-Time Checkable Sufficient Condition for Deadlock-Freedom of Component-Based Systems. *Lecture Notes in Computer Science*, 4362/2007:888–899, 2007.

[12] Mila Majster-Cederbaum and Moritz Martens. Compositional Analysis of Deadlock-Freedom for Tree-Like Component Architectures. In *EMSOFT '08: Proceedings of the 7th ACM international conference on Embedded software*, pages 199–206, New York, NY, USA, 2008. ACM.

[13] Mila Majster-Cederbaum, Moritz Martens, and Christoph Minnameier. Liveness in Interaction Systems. *Electron. Notes Theor. Comput. Sci.*, 215:57–74, 2008.

[14] Mila Majster-Cederbaum and Nils Semmelrock. Reachability in Tree-Like Component Systems is PSPACE-Complete. Technical Report TR-2009-004, University of Mannheim, Germany, 2009.

[15] Mila E. Majster-Cederbaum and Christoph Minnameier. Everything Is PSPACE-Complete in Interaction Systems. In *ICTAC*, pages 216–227, 2008.

[16] Christoph Minnameier and Mila Majster-Cederbaum. Cross-Checking – Enhanced Over-Approximation of the Reachable Global State Space of Component-Based Systems, 2009. submitted for puplication.