Transition Removal for Compositional Supervisor Synthesis

This document has been downloaded from Chalmers Publication Library (CPL). It is the author’s version of a work that was accepted for publication in:


Citation for the published paper:

http://dx.doi.org/10.1109/CoASE.2012.6386447

Downloaded from: http://publications.lib.chalmers.se/publication/165508

Notice: Changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published source. Please note that access to the published version might require a subscription.

Chalmers Publication Library (CPL) offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all types of publications: articles, dissertations, licentiate theses, masters theses, conference papers, reports etc. Since 2006 it is the official tool for Chalmers official publication statistics. To ensure that Chalmers research results are disseminated as widely as possible, an Open Access Policy has been adopted. The CPL service is administrated and maintained by Chalmers Library.
Transition Removal for Compositional Supervisor Synthesis

Sahar Mohajerani  Robi Malik  Martin Fabian

Abstract—This paper investigates under which conditions transitions can be removed from an automaton while preserving important synthesis properties. The work is part of a framework for compositional synthesis of least restrictive controllable and nonblocking supervisors for modular discrete event systems. The method for transition removal complements previous results, which are largely focused on state merging. Issues concerning transition removal in synthesis are discussed, and redirection maps are introduced to enable a supervisor to process an event, even though the corresponding transition is no longer present in the model. Based on the results, different techniques are proposed to remove controllable and uncontrollable transitions, and an example shows the potential of the method for practical problems.

I. INTRODUCTION

Supervisory control theory [1] provides a general framework to compute least restrictive strategies to control a given plant such that its behaviour satisfies a given specification. Synthesis for systems with a large number of components is impeded by an inherent complexity problem known as state-space explosion. A lot of research has been devoted to overcome the state-space explosion problem, and also to find more comprehensible supervisors [1]–[3]. Compositional methods seek to avoid large state spaces using abstraction, and have been used in verification [4], [5] and synthesis [3], [6], [7]. In a system with a large number of components, it is often possible to simplify individual components before composing them with the rest of the system, achieving significant performance improvements. Several ways to simplify components have been investigated in recent years.

Natural projection is a standard and effective way to compute abstractions, although strong restrictions need to be imposed to ensure the preservation of synthesis results [8], [9]. Observation equivalence [10] and conflict equivalence [11] are well-known abstraction methods for nonblocking verification [5], but for synthesis these abstractions can only be applied in combination with unobservable events [12], [13], which limits their applicability.

Recently, frameworks for compositional synthesis based on abstractions of nondeterministic automata have been proposed [3], [6], [7], in some cases showing substantial reduction of the number of states encountered during synthesis. This paper seeks to enhance these methods by providing means to remove transitions. This is important, because for large systems, the number of transitions may exceed the number of states by several orders of magnitude.

Compositional verification often uses observation equivalence for abstraction, which allows for transition removal using the transitive reduction [14], but observation equivalence does not necessarily preserve synthesis results [6]. Supervision equivalence [3] allows for transition removal, but relies on additional state labels that make some desirable abstractions impossible. The methods [6], [7] avoid event hiding that may cause problems in synthesis abstraction, but these approaches make it difficult to remove transitions.

This paper proposes some concrete means to identify transitions that are redundant for the purpose of synthesis. These methods are based on observation equivalence [10], but are more restrictive because of the need to preserve synthesis results. It is also shown how to restore the removed transitions to enable a synthesised supervisor to make control decisions based on a model with removed transitions.

This paper is organised as follows. After the preliminaries in Sect. II, a framework to support transition removal in compositional synthesis is presented in Sect. III. In Sect. IV, a sufficient condition for transition-removing abstraction is described, and in Sect. V, concrete methods to remove transitions are given. Finally, Sect. VI demonstrates transition removal using a practical example, and Sect. VII adds some concluding remarks. Formal correctness proofs are omitted for lack of space in this paper and can be found in [15].

II. PRELIMINARIES

A. Events and Languages

The behaviour of discrete event systems is described using events and languages. Events represent incidents that cause transitions from one state to another and are taken from a finite alphabet Σ. For the purpose of supervisory control, this alphabet is partitioned into the set Σc of controllable events and the set Σu of uncontrollable events. Controllable events can be disabled by a supervisor, while uncontrollable events occur spontaneously, and are prefixed by an exclamation mark (!) in this paper. The special termination event ω ∈ Σc denotes completion of a task, and does not appear anywhere else but to mark such completions.

Σ∗ is the set of all finite traces of events from Σ, including the empty trace ϵ. A subset L ⊆ Σ∗ is called a language. The concatenation of two traces s, t ∈ Σ∗ is written as st. A trace s ∈ Σ∗ is a prefix of t ∈ Σ∗, written s ⊆ t, if t = su for some u ∈ Σ∗. For Ω ⊆ Σ, the natural projection PΩ: Σ∗ → Ω∗ is the operation that removes from traces s ∈ Σ∗ all events not in Ω.
B. Finite-State Automata

Discrete event systems are typically modelled as deterministic automata, but nondeterministic automata may be obtained as intermediate results from abstraction.

Definition 1: A (nondeterministic) finite-state automaton is a tuple $G = \langle \Sigma, Q, \rightarrow, Q_0 \rangle$, where $\Sigma$ is a finite set of events, $Q$ is a finite set of states, $\rightarrow \subseteq Q \times \Sigma \times Q$ is the state transition relation, and $Q_0 \in Q$ is the set of initial states.

The transition relation is written in infix notation $x \xrightarrow{\sigma} y$, and is extended to traces in $\Sigma^*$ by letting $x \xrightarrow{\sigma_1 \ldots \sigma_n} x$ for all $x \in Q$, and $x \xrightarrow{\sigma \sigma_1 \ldots \sigma_n} y$ if $x \xrightarrow{\sigma} y$ and $y \xrightarrow{\sigma_1 \ldots \sigma_n} z$ for some $y \in Q$. Furthermore, $x \xrightarrow{\sigma} y$ means $x \xrightarrow{\sigma} y$ for some $y \in Q$, and $x \xrightarrow{\sigma} y$ means $x \xrightarrow{\sigma} y$ for some $s \in \Sigma^*$. For an alphabet $\Omega \subseteq \Sigma$, the notation $x \xrightarrow{\sigma} y$ means $x \xrightarrow{\sigma} y$ for some $\sigma \in \Omega$, and $G \xrightarrow{\sigma} x$ means $q \xrightarrow{\sigma} x$ for some $q \in Q$. The language of automaton $G$ is $L(G) = \{ s \in \Sigma^* \mid G \xrightarrow{s} \}$.

A special requirement is that states reached by the termination event $\omega$ do not have any outgoing transitions, i.e., if $x \xrightarrow{\omega} y$ then there does not exist $\sigma \in \Sigma$ such that $y \xrightarrow{\sigma} x$. This ensures that the termination event, if it occurs, is always the final event of any trace. The traditional set of marked states is $Q^\omega = \{ x \in Q \mid x \xrightarrow{\omega} \}$ in this notation. For graphical simplicity, states in $Q^\omega$ are shown shaded in the figures of this paper instead of explicitly showing $\omega$-transitions.

When multiple automata are brought together to interact, lock-step synchronisation in the style of [16] is used.

Definition 2: Let $G_1 = \langle \Sigma_1, Q_1, \rightarrow_1, Q_1^0 \rangle$ and $G_2 = \langle \Sigma_2, Q_2, \rightarrow_2, Q_2^0 \rangle$ be two automata. The synchronous composition of $G_1$ and $G_2$ is

\[
G_1 \parallel G_2 = \langle \Sigma_1 \cup \Sigma_2, Q_1 \times Q_2, \rightarrow, Q_1^0 \times Q_2^0 \rangle
\]

(1)

where

\[
\begin{align*}
(x, y) & \xrightarrow{\sigma} (x', y') \text{ if } \sigma \in \Sigma_1 \cap \Sigma_2, \\
(x, y) & \xrightarrow{\sigma_1} (x', y) \text{ if } \sigma \in \Sigma_1 \setminus \Sigma_2, \\
(x, y) & \xrightarrow{\sigma_2} (x, y') \text{ if } \sigma \in \Sigma_2 \setminus \Sigma_1,
\end{align*}
\]

C. Supervisory Control Theory

Given plant and specification automata, supervisory control theory [1] provides a method to synthesise a supervisor that restricts the behaviour of the plant such that the specification is always fulfilled. Two common requirements for this supervisor are controllability and nonblocking.

Definition 3: Specification $K = \langle \Sigma, Q_K, \neg K, Q_K^0 \rangle$ is controllable with respect to plant $G = \langle \Sigma, Q_G, \neg G, Q_G^0 \rangle$ if, for every trace $s \in \Sigma^*$, every state $x \in Q_K$, and every uncontrollable event $\nu \in \Sigma_u$ such that $K \xrightarrow{\nu} x$ and $G \xrightarrow{\nu} x$, it holds that $x \not\xrightarrow{\neg K}$.

Definition 4: An automaton $G = \langle \Sigma, Q, \rightarrow, Q_0 \rangle$ is nonblocking if, for every state $x \in Q$ and every trace $s \in (\Sigma \setminus \{ \omega \})^*$ such that $Q_0 \xrightarrow{\sigma} x$, there exists a trace $t \in \Sigma^*$ such that $x \xrightarrow{\sigma t}$. Two automata $G_1$ and $G_2$ are nonconflicting if $G_1 \parallel G_2$ is nonblocking.

For a plant $G$ and specification $K$, it is shown in [1] that there exists a least restrictive controllable sublanguage

\[
\sup_{\mathcal{C}_G} \mathcal{L}(\mathcal{K}) \subseteq \mathcal{L}(K)
\]

(2)

such that $\sup_{\mathcal{C}_G} \mathcal{L}(\mathcal{K})$ is controllable with respect to $G$ and nonblocking, and this language can be computed using a fixpoint iteration. This result can be reformulated in automata form, using an iteration on the state set. The synthesis result for an automaton $G$ is obtained by restricting $G$ to a maximal set of controllable and nonblocking states.

Definition 5: The restriction of $G = \langle \Sigma, Q, \rightarrow, Q_0 \rangle$ to $X \subseteq Q$ is $G_{\mid X} = \langle \Sigma, Q, \rightarrow_{\mid X}, Q_{0} \cap X \rangle$ where $\rightarrow_{\mid X} = \{(x, y) \in \rightarrow \mid x, y \in X\}$.

Definition 6: [17] Let $G = \langle \Sigma, Q, \rightarrow, Q_0 \rangle$ be an automaton. The synthesis step operator $\Theta_G: 2^Q \rightarrow 2^Q$ for $G$ is defined by $\Theta_G(X) = \Theta_G^{\text{cont}}(X) \cap \Theta_G^{\text{nonb}}(X)$, where

\[
\Theta_G^{\text{cont}}(X) = \{ x \in X \mid x \xrightarrow{\nu} \text{ implies } y \in X \} \quad \text{(3)}
\]

\[
\Theta_G^{\text{nonb}}(X) = \{ x \in X \mid x \xrightarrow{\sigma t} \text{ for some } t \in \Sigma^* \} \quad \text{(4)}
\]

Theorem 1: [17] Let $G = \langle \Sigma, Q, \rightarrow, Q_0 \rangle$. The synthesis step operator $\Theta_G$ has a greatest fixpoint $\text{gfp}(\Theta_G) = \Theta_G \subseteq Q$. If the state set $Q$ is finite, then the sequence $X^0 = Q$, $X^{i+1} = \Theta_G(X^i)$ reaches this fixpoint in a finite number of steps, i.e., $\Theta_G = X^n$ for some $n \geq 0$.

Definition 7: The synthesis result for $G = \langle \Sigma, Q, \rightarrow, Q_0 \rangle$ is $\text{supCN}(G) = G_{\mid \Theta_G}$.

Theorem 2: Let $G = \langle \Sigma, Q, \rightarrow, Q_0 \rangle$ be a deterministic automaton. $\text{supCN}(G)$ is the least restrictive subautomaton of $G$ that is controllable with respect to $G$ and nonblocking.

The synthesis operator $\text{supCN}$ performs synthesis for a plant automaton $G$. A simple transformation [3] exists to transform problems that also involve specifications into the plant-only control problems considered in this paper.

The result of synthesis is an automaton $\text{supCN}(G)$ or a language $\mathcal{L}(\text{supCN}(G))$, which describes the behaviour of a controlled system. In practice this is implemented as a supervisor that decides which controlling events are to be enabled or disabled in a given state. In this paper, a supervisor is a map

\[
S: \Sigma^* \rightarrow \{0, 1\}
\]

(5)

If $S(s\sigma) = 0$ for some $s \in \Sigma^*$ and $\sigma \in \Sigma_e$ then the supervisor disables the controllable event $\sigma$ after observing trace $s$, otherwise it enables $\sigma$. This results in the following closed-loop behaviour $\mathcal{L}(S/G)$ of the plant $G$ under the control of supervisor $S$:

\[
\mathcal{L}(S/G) = \{ s \in \mathcal{L}(G) \mid S(s) = 1 \}
\]

(6)

A supervisor can be constructed naturally from a language $L \subseteq \Sigma^*$, by letting $S_L(s) = 1$ if and only if $s \in L$. For such a supervisor to be feasible, $L$ must be controllable [1].
as explained below in Example 2, the supervisor removed transitions. A supervisor can enact control over the events labelling the plant model from an automaton. This leads to a problem, because it is no longer obvious how to construct a supervisor from such an abstraction. After removal of transitions it is not clear how a supervisor can enact control over the events labelling the removed transitions.

Example 1: Consider automata $G$ and $T$ in Fig. 1 with $\Sigma_\alpha = \Upsilon = \{\gamma\}$. Automaton $H$ is obtained by removing $q_0 \xrightarrow{\alpha} q_2$. Although $H$ is an appropriate abstraction of $G$, as explained below in Example 2, the supervisor $S_H = \sup \mathcal{CN}(H \parallel T)$ disables event $\alpha$ in the initial state, and therefore is not a least restrictive supervisor for $G \parallel T$.

To solve this problem, the models (7) are augmented by a redirection map that contains the information needed to finally implement a supervisor.

Definition 8: A synthesis pair is a pair $(G; D)$, where

- $G = \{G_1, G_2, \ldots, G_n\}$ is a set of uncontrolled plant automata;
- $D: \Sigma^* \rightarrow \Sigma^*$ is a prefix-preserving redirection map, i.e., a map such that $s \subseteq t$ implies $D(s) \subseteq D(t)$.

The compositional synthesis algorithm manipulates synthesis pairs. Each pair represents a partially solved synthesis problem, consisting of the plant model $G$ to be controlled and the redirection map $D$, which maps each input trace $s$ accepted by the original plant before all abstractions, to a trace accepted by the current abstracted plant $G$. A solution to the abstracted synthesis problem $G$ can be interpreted as a supervisor for the original plant by taking the redirection map into account.

Definition 9: For every synthesis pair $(G; D)$, define the represented supervisor map $S_{(G; D)} : \Sigma^* \rightarrow \{0, 1\}$ as follows:

$$S_{(G; D)}(s) = \begin{cases} 1, & \text{if } D(s) \in \mathcal{L}(\sup \mathcal{CN}(G)); \\ 0, & \text{otherwise}. \end{cases}$$

Compositional synthesis starts by converting a control problem such as (7) into a synthesis pair $(G_0; \text{id})$ where $G_0 = \{G_1, G_2, \ldots, G_n\}$ and $\text{id}: \Sigma^* \rightarrow \Sigma^*$ is the identity map, i.e., $\text{id}(s) = s$ for all $s \in \Sigma^*$. This initial synthesis pair is repeatedly abstracted such that the supervisor obtained from the abstraction remains a solution for the original problem. To ensure this property, each new synthesis pair needs to be synthesis equivalent to the previous pair.

Definition 10: Two synthesis pairs $(G_1; D_1)$ and $(G_2; D_2)$ are called synthesis equivalent with respect to plant $G$, written $(G_2; D_2) \simeq_{\text{synth, } G} (G_1; D_1)$, if $\mathcal{L}(S_{(G_1; D_1)}/G) = \mathcal{L}(S_{(G_2; D_2)}/G)$. Furthermore, $(G_1; D_1)$ and $(G_2; D_2)$ are synthesis equivalent, written $(G_2; D_2) \simeq_{\text{synth}} (G_1; D_1)$, if $(G_2; D_2) \simeq_{\text{synth, } G} (G_1; D_1)$ for every automaton $G$.

Compositional synthesis terminates once $G = \{H\}$ consists of a single automaton representing the abstracted system description. The following result, proved in [15], confirms that the closed-loop behaviour obtained in the end is equal to a solution for the original synthesis problem.

Proposition 3: Let $G_0 = \{G_1, \ldots, G_n\}$ be a set of automata, and let $(G_k; D_k)$ be a synthesis pair such that $(G_0; \text{id}) \simeq_{\text{synth, } G_0} (G_k; D_k)$. Then

$$\mathcal{L}(S_{(G_k; D_k)}/G_0) = \mathcal{L}(\sup \mathcal{CN}(G_0)).$$

IV. TRANSITION-WISE SYNTHESIS EQUIVALENCE

Several methods are known to abstract synthesis pairs such that the number of states is reduced [3], [6]. The abstractions are performed by manipulating the states and transitions of individual automata, such that synthesis equivalence is preserved. To allow for transition removal, state-wise synthesis abstraction, which is a special case of a definition from [6], is augmented by a transition-based concept in Def. 12.

Definition 11: Let $G = (\Sigma, Q, \rightarrow_G, Q^o)$ and $H = (\Sigma, Q, \rightarrow_H, Q^o)$ be two automata. $H$ is a state-wise synthesis abstraction of $G$ with respect to $\Upsilon \subseteq \Sigma$, if it holds for all automata $T$ with $\Sigma_T \cap \Upsilon = \emptyset$ that $\Theta_G[T] \subseteq \Theta_H[T]$.

Definition 12: Let $G = (\Sigma, Q, \rightarrow_G, Q^o)$ and $H = (\Sigma, Q, \rightarrow_H, Q^o)$ be two automata. $H$ is a transition-wise synthesis abstraction of $G$ with respect to $\Upsilon \subseteq \Sigma$ if for every transition $x \xrightarrow{\tau} y$ there exist $t, u \in \Upsilon^*$ such that:

(i) $x \xrightarrow{\tau P_1(\sigma) u} y$;
(ii) for all automata $T$ such that $\Sigma_T \cap \Upsilon = \emptyset$ and all transitions $(x, x' y, \tau P_2(\sigma) y') \in \sup \mathcal{CN}(G \parallel T)$, it holds that $(x, x') \xrightarrow{\tau P_1(\sigma) u} (y, y')$.

Definition 13: Two automata $G$ and $H$ are state-wise (or transition-wise) synthesis equivalent with respect to $\Upsilon$, if $G$ is a state-wise (or transition-wise) synthesis abstraction of $H$. 

with respect to $\Upsilon$ and $H$ is a state-wise (or transition-wise) synthesis abstraction of $G$ with respect to $\Upsilon$.

To preserve transition-wise synthesis equivalence after removal of a transition, Def. 12 requires the existence of a so-called redirection path that links the source and target states of the removed transition. A redirection path for transition $x \rightarrow y$ with respect to $\Upsilon$ is a path $x \underbrace{\tau_{\gamma} \sigma}_t u y$ such that $t, u \in \Upsilon^*$. Using these paths, the redirection map is constructed to replace the removed transitions by the matching redirection paths. This enables the supervisor to make control decisions about the removed transitions.

Example 2: Consider again the automata in Fig. 1. Transition $q_0 \xrightarrow{\alpha} q_2$ can be removed from $G$, producing the state-wise and transition-wise synthesis equivalent automaton $H$. From this abstraction, a redirection map $D : \Sigma^* \rightarrow \Sigma^*$ is constructed where $D(\alpha s) = \gamma \sigma \alpha s$ for all $s \in \Sigma^*$ and $D(s) = s$ for all $s$ such that $\alpha$ is not a prefix of $s$.

If $G$ in Fig. 1 is placed in a larger system, say $G = (G, T)$, then the synthesis pair $(G; id)$ is synthesis equivalent to $(H; D)$ where $H = \{H, T\}$. Although the supervisor $S_H = \text{supCN}(H \parallel T)$ obtained for $H$ cannot directly be used to control the original plant $G$, this becomes possible in combination with the redirection map $D$. As $D(\alpha s) = \gamma \sigma \alpha s$ in $\mathcal{L}(\text{supCN}(H \parallel T))$, the supervisor computed for $(H, D)$ will enable the controllable event $\alpha$ in the initial state, in the same way as a supervisor computed for the original system $G$.

The following result confirms that a redirection map as shown in Example 2 can be constructed in all cases where transition removal applied to a component results in a state-wise and transition-wise synthesis equivalent abstraction.

**Theorem 4:** [15] Let $G = \{G_1, \ldots, G_n\}$ and $\mathcal{H} = \{H_1, G_2, \ldots, G_n\}$ such that $G_1$ and $H_1$ are state-wise and transition-wise synthesis equivalent with respect to $\Upsilon \subseteq \Sigma_1$ such that $\Upsilon \cap \Sigma_2 = \cdots = \Upsilon \cap \Sigma_n = \emptyset$ and $\neg H_1 \subseteq \neg G_1$. Then there exists a redirection map $D_1 : \Sigma^* \rightarrow \Sigma^*$ such that $(G; D) \equiv_{\text{synth}} (H; D_1 \circ D)$.

V. TRANSITION REMOVAL ABSTRACTION

According to Theorem 4, synthesis results are preserved if transition removal in a component results in a state-wise and transition-wise synthesis equivalent abstraction. This section proposes some concrete methods to construct such abstractions, based on the idea of observation equivalence.

A. Observation Equivalence

**Observation equivalence or weak bisimilarity** is a well-known general abstraction method for nondeterministic automata [10]. It can be implemented by simple algorithms, and its application in compositional verification can substantially reduce the state space [5]. Observation equivalence is tested based on the transitional closure of the local event transitions [18]. The number of transitions can be substantially reduced by considering only the transitive reduction. More precisely, a transition $x \rightarrow y$ is observation equivalence redundant and can be removed [14] if the automaton contains a matching redirection path.

**Definition 14:** Let $G = \langle \Sigma, Q, \rightarrow_G, Q^o \rangle$ and $H = \langle \Sigma, Q, \rightarrow_H, Q^o \rangle$ be two automata with $\Sigma = \Omega \cup \Upsilon$ and $\neg H \subseteq \neg G$. Automaton $H$ is a result of **observation equivalence redundant transition removal** from $G$ with respect to $\Upsilon$, if for all transitions $x \xrightarrow{t} y$ there exist $t, u \in \Upsilon^*$ such that $x \xrightarrow{t \rho_3(\gamma)} y$.

Observation equivalence redundant transitions can be removed while preserving observation equivalence, which in turn ensures preservation of most temporal logic properties [10], [14]. Unfortunately, this does not include synthesis equivalence [6].

Example 3: Consider automata $G, H$, and $T$ in Fig. 2. The uncontrollable transition $q_1 \xrightarrow{\nu} q_3$ is observation equivalence redundant with respect to $\Upsilon = \{\beta\}$. Removing it has no effect on $H$. In $G$ and $H$, the uncontrollable event $\nu$ leads to the blocking state $q_3$. With $H$, blocking can be prevented by disabling $\beta$, leaving only the initial state. But with $G$, the uncontrollable transition $q_1 \xrightarrow{\nu} q_3$ produces an empty synthesis result. The test $T$ demonstrates that $G$ and $H$ are not state-wise synthesis equivalent. $G$ is not a state-wise synthesis abstraction of $H$.

This counterexample shows that in general synthesis equivalence is not preserved by removing observation equivalence redundant transitions, so extra restrictions need to be imposed.

B. Uncontrollable Redundant Transitions

In Example 3, if the local event $\beta$ was uncontrollable, then the resultant abstraction $H$ would be a transition-wise synthesis abstraction of $G$. This suggests to interpret an uncontrollable transition as redundant if the local transitions used in the redirection path are also uncontrollable.

**Definition 15:** Let $G = \langle \Sigma, Q, \rightarrow_G, Q^o \rangle$ and $H = \langle \Sigma, Q, \rightarrow_H, Q^o \rangle$ be two automata with $\Sigma = \Omega \cup \Upsilon$ and $\neg H \subseteq \neg G$. Automaton $H$ is a result of **uncontrollable redundant transition removal** from $G$ with respect to $\Upsilon$, if the following conditions hold for all transitions $x \xrightarrow{t} y$.

(i) If $\sigma \in \Sigma_e$ then $x \xrightarrow{t \rho_3(\gamma)} y$.
(ii) If $\sigma \in \Sigma_u$ then there exist $t, u \in (\Upsilon \cap \Sigma_u)^*$ such that $x \xrightarrow{t \rho_3(\gamma) u} y$.

The transitions present in $\neg G$ but not in $\neg H$ in Def. 15 are called uncontrollable redundant transitions. These transitions can be removed while producing a synthesis equivalent abstraction.

**Theorem 5:** [15] Let $H = \langle \Sigma, Q, \rightarrow_H, Q^o \rangle$ be a result of uncontrollable redundant transition removal from $G = \langle \Sigma, Q, \rightarrow_G, Q^o \rangle$ with respect to $\Upsilon \subseteq \Sigma$. Then $G$ and $H$ are state-wise and transition-wise synthesis equivalent with respect to $\Upsilon$. 

![Fig. 2. $H$ is observation equivalent to $G$, but not a synthesis abstraction.](image-url)
C. Controllable Redundant Transitions

For uncontrollable events, an uncontrollable redirection path guarantees transition-wise synthesis equivalence. For controllable events, all events on a redirection path except for the event of the removed transition should be uncontrollable. However, the following counterexample reveals that one more condition is needed to guarantee a correct abstraction.

**Example 4:** Consider automaton $G_1$ in Fig. 3 where $\Sigma_u = \{\mu, \nu\}$. Transition $q_0 \xrightarrow{\alpha} q_3$ is observation equivalence redundant because $q_0 \xrightarrow{\mu\rho_{\alpha}} q_3$. Let $H_1$ be the result of removing the transition $q_0 \xrightarrow{\alpha} q_3$. In both $G_1$ and $H_1$, the controllable transition $q_1 \xrightarrow{\nu\rho_{\alpha}} q_2$ must be disabled to avert blocking via the uncontrollable event $\nu$. Removing this transition makes $q_3$ unreachable in $\supCN(H_1 \parallel T)$, but it remains reachable in $\supCN(G_1 \parallel T)$. The test $T$ demonstrates that $G_1$ and $H_1$ are not transition-wise synthesis equivalent as $G_1$ is not a transition-wise synthesis abstraction of $H_1$.

Example 4 shows that there is a problem with uncontrollable local events after the event of a removed transition on a redirection path. The problem disappears if there are no further events after the removed event, as in automaton $G_2$ in Fig. 3. This leads to the idea of controllable prefix-redundant transition removal, which can be shown to imply both state-wise and transition-wise synthesis abstraction.

**Definition 16:** Let $G = (\Sigma, Q, \rightarrow, Q^0)$ and $H = (\Sigma, Q, \rightarrow, Q^0)$ be two automata with $\Sigma = \Omega \cup \Upsilon$ and $\rightarrow \subseteq \rightarrow_G$. Automaton $H$ is a result of controllable prefix-redundant transition removal from $G$ with respect to $\Upsilon$, if the following conditions hold for all transitions $x \xrightarrow{\sigma} y$.

(i) If $\sigma \in \Sigma_u$ then $x \xrightarrow{\sigma} H y$.
(ii) If $\sigma \in \Sigma_c$ then there exists $t \in (\Upsilon \cap \Sigma_u)^*$ such that $x \xrightarrow{t \rho_{\sigma}(\sigma)} H y$.

Controllable prefix-redundant transition removal only allows for local events before the event of a removed transition. Local events after this event can also be considered by adding additional requirements.

**Example 5:** As shown in Example 4, removal of the transition $q_0 \xrightarrow{\alpha} q_3$ in $G_1$ in Fig. 3 does not ensure synthesis abstraction because of the uncontrollable $!\nu$-transition in state $q_2$. Automaton $G_2$ also has the observation equivalence redundant transition $q_0 \xrightarrow{\alpha} q_3$ and an $!\nu$-transition enabled after $\alpha$ on the redirection path $q_0 \xrightarrow{\mu\rho_{\alpha}} q_3$. Yet, in this case, the $!\nu$-transition does not lead to a blocking state, and the removal of $q_0 \xrightarrow{\alpha} q_3$ results in a state-wise and transition-wise synthesis equivalent automaton.

Automata $G_1$ and $G_2$ in Fig. 3 differ in the target state of $q_2 \xrightarrow{\nu}$. This suggests to allow uncontrollable events in the second part of a redirection provided that they are local and lead to a target state on the redirection path.

**Definition 17:** Let $G = (\Sigma, Q, \rightarrow, Q^0)$ be an automaton and $\Upsilon \subseteq \Sigma$. A path

$$x_0 \xrightarrow{\sigma_1} x_1 \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_k} x_k$$

is a weakly controllable $\Upsilon$-path if $\sigma_1, \ldots, \sigma_k \in \Sigma$ and for all uncontrollable transitions $x_l \xrightarrow{\nu} y$ with $0 \leq l < k$ and $\nu \in \Sigma_u$ it holds that $\nu \in \Upsilon$ and $y = x_j$ for some $0 \leq j \leq k$.

A weakly controllable path consists of only local transitions, and furthermore all uncontrollable transitions enabled along this path must use local events and lead to states along the path. Imposing this condition on the redirection path gives the condition for a controllable suffix-redundant transition, which is sufficient for synthesis equivalence.

**Definition 18:** Let $G = (\Sigma, Q, \rightarrow, Q^0)$ and $H = (\Sigma, Q, \rightarrow, Q^0)$ be two automata with $\Sigma = \Omega \cup \Upsilon$ and $\rightarrow \subseteq \rightarrow_G$. Automaton $H$ is a result of controllable suffix-redundant transition removal from $G$ with respect to $\Upsilon$, if the following conditions hold for all transitions $x \xrightarrow{\sigma} y$.

(i) If $\sigma \in \Sigma_u$ then $x \xrightarrow{\sigma} H y$.
(ii) If $\sigma \in \Sigma_c$ then there exists $u \in \Upsilon^*$ such that $x \xrightarrow{\rho_{\Upsilon}(\sigma)} H z \xrightarrow{\nu} H y$, and $z \xrightarrow{\nu} H y$ is a weakly controllable $\Upsilon$-path.

Both controllable prefix-redundant and suffix-redundant transition removal preserve synthesis equivalence. These conditions can be combined to allow sequences of local events before and after a removed transition.

**Definition 19:** Let $G = (\Sigma, Q, \rightarrow, Q^0)$ and $H = (\Sigma, Q, \rightarrow, Q^0)$ be two automata with $\Sigma = \Omega \cup \Upsilon$ and $\rightarrow \subseteq \rightarrow_G$. Automaton $H$ is a result of controllable redundant transition removal from $G$ with respect to $\Upsilon$, if the following conditions hold for all transitions $x \xrightarrow{\sigma} y$.

(i) If $\sigma \in \Sigma_u$ then $x \xrightarrow{\sigma} H y$.
(ii) If $\sigma \in \Sigma_c$ then there exists $t \in (\Upsilon \cap \Sigma_u)^*$ and $u \in \Upsilon^*$ such that $x \xrightarrow{t \rho_{\Upsilon}(\sigma)} H z \xrightarrow{\nu} H y$, and $z \xrightarrow{\nu} H y$ is a weakly controllable $\Upsilon$-path.

**Theorem 6:** [15] Let $H = (\Sigma, Q, \rightarrow, Q^0)$ be a result of controllable redundant transition removal from $G = (\Sigma, Q, \rightarrow, Q^0)$ with respect to $\Upsilon \subseteq \Sigma$. Then $G$ and $H$ are state-wise and transition-wise synthesis equivalent with respect to $\Upsilon$.

VI. Example

In this section, the proposed synthesis procedure is applied to a manufacturing system. The model consists of four machines $M_1$, $M_2$, $M_3$, and $M_4$, linked by two buffers $B_1$ and $B_2$. Workpieces are first processed by $M_1 (s_1)$ and then placed into $B_1 (f_1)$, then they go to $M_2$ ($s_2$) and are placed into $B_2 (f_2)$. From $B_2$, the workpieces either go to $M_3$ for final processing ($s_3$) or to $M_4$ ($s_4$) for additional processing. However, $M_4$ has a fault that occasionally sends a workpiece back to $B_1$ ($\text{re}$). At any time, $M_1$ and $B_1$ can be reset by the controllable event $rs$. Fig. 4 shows the system layout and the automata model. Events $!f_1$, $!f_2$, $!f_3$, $!f_4$ and $!\text{re}$ are uncontrollable, all other events are controllable.
Compositional synthesis starts with the pair $\{G_0; \text{id}\}$ where $G_0 = \{M_1, M_2, M_3, M_4, B_1, B_2\}$. The first step is to calculate the composition $B_1 \parallel M_1$ shown in Fig. 5. Now $f_1$, $r_s$, and $s_1$ are local events, which makes $q_0 \xrightarrow{r_s} q_0$ and $q_2 \xrightarrow{r_s} q_0$ controllable prefix-redundant transitions with redirection paths $q_0 \xrightarrow{f_1} q_3$ and $q_2 \xrightarrow{f_1} q_3$. Removal of these transitions results in $H_1$. The modified synthesis pair is $(G_1; D_1)$ where $G_1 = \{H_1, M_2, M_3, M_4, B_2\}$ and $D_1$ redirects $q_2 \xrightarrow{r_s} q_0$ and $q_0 \xrightarrow{r_s} q_0$ via $q_2 \xrightarrow{f_1} q_3 \xrightarrow{r_s} q_0$ and $q_0 \xrightarrow{f_1} q_0$ respectively.

Next, $B_2 \parallel M_3$ is computed, shown in Fig. 5. This makes $f_3$ and $s_3$ local events, and $q_3 \xrightarrow{f_3} \perp$ becomes an uncontrollable redundant transition with redirection path $q_3 \xrightarrow{f_3} q_1 \xrightarrow{f_3} \perp$. The new synthesis pair is $(G_2; D_2 \circ D_1)$ where $G_2 = \{H_1, M_2, M_4, H_2\}$ and $D_2$ redirects $q_3 \xrightarrow{f_3} \perp$ via $q_3 \xrightarrow{f_3} q_1 \xrightarrow{f_3} \perp$.

The final synthesis step to compute $\text{supCN}(G_2)$ explores the state space of $G_2$ which has 100 states and 290 transitions. This is in contrast to standard monolithic synthesis, which explores the same state space using 340 transitions. Both the final monolithic and compositional supervisor have 26 states. However, the compositional supervisor has 63 transitions, while the monolithic supervisor has 81 transitions.

These improvements have been achieved by removing just three transitions from the model. More savings are likely in larger contexts, particularly in combination with state-removing abstraction rules.

**VII. CONCLUSIONS**

It has been shown under which conditions transitions can be removed from an automaton while preserving compositional synthesis results. Different techniques to remove controllable and uncontrollable transitions have been presented, and a practical example has demonstrated how the number of transitions is reduced. The methods proposed in this paper are not intended to be used in isolation, but they will be combined with other synthesis-preserving abstraction methods. In the future, the authors plan to develop a framework for compositional synthesis that combines abstractions that remove states [3], [6] and transitions, as well as renaming [7] to remove nondeterminism.

**REFERENCES**


