Compositional Synthesis of Discrete Event Systems Using Synthesis Abstraction

Sahar Mohajerani, Robi Malik, Simon Ware and Martin Fabian

***Abstract*— This paper proposes a general method to synthesize a least restrictive supervisor for a large discrete event system model, consisting of a large number of arbitrary automata representing the plants and specifications. A new type of abstraction, called *synthesis abstraction* is introduced and three rules are proposed to calculate an abstraction of a given automaton. Furthermore, a compositional algorithm for synthesizing a supervisor for large-scale systems of composed finite-state automata is proposed. In the proposed algorithm, the synchronous composition is computed step by step and intermediate results are simplified according to synthesis abstraction. Then a supervisor for the abstracted system is calculated, which in combination with the original system gives the least restrictive, nonblocking, and controllable behaviour.**

I. Introduction

Supervisory control theory is a general framework for designing a supervisor for discrete event systems [2], [9]. Synthesising a supervisor for systems with a large number of components suffers from an inherent complexity problem known as state-space explosion. In order to overcome the problem, modular approaches to construct supervisors for large-scale systems have been studied. Modular and hierarchical approaches [11], [13] can produce well-designed supervisors, yet they are based on structural information provided by users and therefore are difficult to automate. Other early methods such as [1] only consider the synthesis of a least restrictive controllable supervisor, ignoring nonblocking.

More recently, abstraction based on *natural projection* has been studied for compositional supervisor synthesis. Natural projection with the *observer property* is shown in [3] to produce a nonblocking but not necessarily least restrictive supervisor; if *output control consistency* is added as an additional requirement, least restrictiveness can be ensured [3]. In [10], it is furthermore shown that output control consistency can be replaced by a weaker condition called *local control consistency*. A drawback of the projection-based methods is their strong connection to events, which makes it difficult to treat different transitions labelled with the same event in different ways.

Supervisor synthesis and abstractions have also been studied in a nondeterministic setting. In [6], [12], *conflict-preserving* abstractions and *weak observation equivalence* are shown to be adequate for the synthesis of nonblocking supervisors, but least restrictiveness is only guaranteed if all observable events are retained in the abstraction. The methods in [5], [7] also allow for the abstraction of observable events through *hiding*. In [5], a least monolithic restrictive supervisor is constructed in symbolic form, after abstracting automata according to *supervision equivalence*. Yet, the equivalence requires additional *state labels*, making some desirable abstractions impossible. State labels are removed in [7], where supervision equivalence is replaced by *synthesis equivalence*, and hiding is used to abstract all local events. The authors propose a two-pass algorithm for compositional synthesis, which produces an over-approximation of the least restrictive solution; an additional nonblocking check is necessary to guarantee correctness.

This paper combines the strengths of the previous work in [5], [7], using three abstraction rules preserving an alternative abstraction relation called *synthesis abstraction*. Local and global events are distinguished in each abstraction step, avoiding both state labels and hiding, while still providing more general simplification than natural projection. Due to the avoidance of hiding, the two-pass procedure of [7] can be replaced by a single pass, and the method is guaranteed to produce a least restrictive modular supervisor in all cases.

This paper is organised as follows. Section II introduces the required notation from supervisory control theory. The proposed algorithm for finding the least restrictive supervisor of a system and the rules to abstract a given automaton, according to synthesis abstraction, are explained in Section III. In Section IV, the proposed algorithm is applied on an example. Finally, conclusions are drawn in Section V.

II. Preliminaries and Notation

A. Events and Languages

The main elements of discrete event system modeling are *states* and *events*. States represent situations under which certain rules and conditions hold. Events represent incidents that cause transitions from one state to another. A set of events will be referred to as an *alphabet*, denoted Σ. For the purpose of supervisory control, Σ is partitioned into two disjoint subsets, the set Σ_c of *controllable* events and the set Σ_u of *uncontrollable* events. The set of *all* finite strings of elements of Σ, including the empty string ε is denoted by Σ^*. A subset $L \subseteq \Sigma^*$, is called a *language*. The concatenation of two strings $s, t \in \Sigma^*$ is written as st. $\overline{L}$ is the prefix-closure of a language $L \subseteq \Sigma^*$ and it is defined as $\overline{L} = \{s \in \Sigma^* | \exists t \in \Sigma^* \text{ and } st \in L\}$.

B. Nondeterministic Automata

Finite-state automata are used to describe discrete event systems behavior. We will assume that all given models are

This work was supported by the Swedish Research Council.

{S. Mohajerani and M. Fabian} Department of Signal and System, Chalmers University of Technology, Gothenburg, Sweden {mohajera, fabian}@chalmers.se

{R. Malik and S.Ware} Department of Computer Science, University of Waikato, Hamilton, New Zealand {robi, siw4}@cs.waikato.ac.nz

deterministic and that non-determinism arises as a consequence of manipulation of these automata.

Definition 1: A nondeterministic finite-state automaton is a 5-tuple $G = \langle \Sigma, Q, \rightarrow, Q^i, Q^m \rangle$, where Σ 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*, $Q^i \subseteq Q$ is the set of *initial states*, and $Q^m \subseteq Q$ is the set of *marked states*. G is deterministic if $|Q^i| \leq 1$ and $x \xrightarrow{\sigma} y_1$ and $x \xrightarrow{\sigma} y_2$ always implies $y_1 = y_2$.

The transition relation is written in infix notation $x \xrightarrow{\sigma} y$, and is extended to strings in Σ^* by letting $x \xrightarrow{\varepsilon} x$ for all $x \in Q$, and $x \xrightarrow{s\sigma} z$ if $x \xrightarrow{s} y$ and $y \xrightarrow{\sigma} z$ for some y. For an automaton $G \xrightarrow{s}$ means the existence of $x \in Q^i$ and $y \in Q$, such that $x \xrightarrow{s} y$ and $G \rightarrow x$ means $\exists s \in \Sigma^*$, such that $G \xrightarrow{s} x$, see [4].

Definition 2: Let $G = \langle \Sigma, Q, \rightarrow, Q^i, Q^m \rangle$ be an automaton. The subset of all the strings s such that $G \xrightarrow{s}$ is the language of the automaton denoted $\mathcal{L}(G)$. The subset of $\mathcal{L}(G)$ that contains only the strings s such that $G \xrightarrow{s} Q^m$, is the marked language and is written as $\mathcal{L}_m(G)$.

When automata are brought together to interact, the interaction occurs on shared events occurring synchronously or not at all. This is modeled by *synchronous composition*.

Definition 3: Let $G_1 = \langle \Sigma_1, Q_1, \rightarrow_1, Q_1^i, Q_1^m \rangle$ and $G_2 = \langle \Sigma_2, Q_2, \rightarrow_2, Q_2^i, Q_2^m \rangle$ be two automata. The *synchronous composition* of G_1 and G_2 is defined as

$$G_1 \parallel G_2 = \langle \Sigma_1 \cup \Sigma_2, Q_1 \times Q_2, \rightarrow, Q_1^i \times Q_2^i, Q_1^m \times Q_2^m \rangle \,, \tag{1}$$

where

$$\begin{aligned} (x,y) \xrightarrow{\sigma} (x',y') &\text{ if } \sigma \in (\Sigma_1 \cap \Sigma_2),\ x \xrightarrow{\sigma}_1 x',\ y \xrightarrow{\sigma}_2 y' \,; \\ (x,y) \xrightarrow{\sigma} (x',y) &\text{ if } \sigma \in (\Sigma_1 \setminus \Sigma_2),\ x \xrightarrow{\sigma}_1 x' \,; \\ (x,y) \xrightarrow{\sigma} (x,y') &\text{ if } \sigma \in (\Sigma_2 \setminus \Sigma_1),\ y \xrightarrow{\sigma}_2 y' \,. \end{aligned}$$

We define a relation between automata, we will say that an automaton is a *subautomaton* of another if the structure of the first is contained within the second, and they both have the same alphabet.

Definition 4: Let $G_1 = \langle \Sigma, Q_1, \rightarrow_1, Q_1^i, Q_1^m \rangle$ and $G_2 = \langle \Sigma, Q_2, \rightarrow_2, Q_2^i, Q_2^m \rangle$ be two automata. G_1 is a *subautomaton* of G_2, written $G_1 \subseteq G_2$, if $Q_1 \subseteq Q_2$, $\rightarrow_1 \subseteq \rightarrow_2$, $Q_1^i \subseteq Q_2^i$, and $Q_1^m \subseteq Q_2^m$.

Another common automaton operation is the *quotient* modulo an equivalence relation on the state set.

Definition 5: Let $G = \langle \Sigma, Q, \rightarrow, Q^i, Q^m \rangle$ be an automaton and let $\sim \subseteq Q \times Q$ be an equivalence relation. The *quotient automaton* of G modulo $\sim$ is

$$G/{\sim} = \langle \Sigma, Q/{\sim}, \rightarrow/{\sim}, Q^i/{\sim}, Q^m/{\sim} \rangle \tag{2}$$

where $\rightarrow/{\sim} = \{\, [x] \xrightarrow{\sigma} [y] \mid x \xrightarrow{\sigma} y \,\}$. Here, $[x] = \{\, x' \in Q \mid x \sim x' \,\}$ denotes the *equivalence class* of $x \in Q$, and $Q/{\sim} = \{\, [x] \mid x \in Q \,\}$ is the set of all equivalence classes modulo $\sim$.

C. Supervisory Control Theory

Considering plant and specification, *supervisory control theory* provides a method to synthesise a supervisor that restricts the behaviour of the plant such that the given specification is always fulfilled. Two requirements for the supervisor are *controllability* and *nonblocking*. Nonblocking expresses the liveness requirements of the system, while controllability captures safety.

Definition 6: [7] Let $G = \langle \Sigma, Q_G, \rightarrow_G, Q_G^i, Q_G^m \rangle$ and $K = \langle \Sigma, Q_K, \rightarrow_K, Q_K^i, Q_K^m \rangle$ be two automata such that $K \subseteq G$. K is *controllable* in G if, for all states $x \in Q_K$ and $y \in Q_G$ and for every uncontrollable event $\upsilon \in \Sigma_u$ such that $x \xrightarrow{\upsilon}_G y$, it also holds that $x \xrightarrow{\upsilon}_K y$.

Definition 7: [7] Let G be an automaton. A state x is called *reachable* in G if $G \rightarrow x$, and *coreachable* in G if $x \rightarrow Q^m$. The automaton G is called reachable or coreachable if every state in G has this property. G is called *nonblocking* if every reachable state is coreachable.

The upper bound of controllable and nonblocking subautomata is again controllable and nonblocking, and this implies the existence of a least restrictive synthesis result.

Definition 8: Let G be an automaton. The supremal controllable and nonblocking subautomaton of G is

$$\sup\mathcal{CN}(G) = \{\, G' \subseteq G \mid G' \text{ is controllable and nonblocking for } G \text{ and } \Sigma_u \,\} \tag{3}$$

Therefore, $\sup\mathcal{CN}(G)$ is the unique synthesis result for a *plant* G. Synthesis is done by iteratively removing blocking and uncontrollable states of a plant, until a fixed point is reached, and restricting the automaton to these states.

Definition 9: [7] Let $G = \langle \Sigma, Q, \rightarrow_G, Q^i, Q^m \rangle$ be an automaton. The *restriction* of G to $X \subseteq Q$ is

$$G_{|X} = \langle \Sigma, X, \rightarrow_{|X}, Q^i \cap X, Q^m \cap X \rangle \,, \tag{4}$$

where $\rightarrow_{|X} = \{\, (x,\sigma,y) \mid x, y \in X \,\}$.

Definition 10: The function that does synthesis is denoted as $\Theta_G(X) = \Theta_G^{nonb}(X) \cap \Theta_G^{cont}(X)$ where

$$\begin{aligned} \Theta_G^{nonb}(X) &= \{\, x \in X \mid \exists y \in Q^m \text{ and } t \in \Sigma^*, x \xrightarrow{t}_{|X} y \,\}, \\ \Theta_G^{cont}(X) &= \{\, x \in X \mid \forall \sigma \in \Sigma_u, x \xrightarrow{\sigma} y \text{ implies } y \in X \,\}. \end{aligned} \tag{5}$$

The first function captures nonblocking and the second one controllability.

The synthesis result is a part of G when the synthesis function is restricted to the greatest fixed point, see [5].

Theorem 1: [7] Let $G = \langle \Sigma, Q, \rightarrow_G, Q^i, Q^m \rangle$. The synthesis step operator Θ_G has a greatest fixed point $\text{gfp}\Theta_G = \hat{\Theta}_G \subseteq Q$, such that $G_{|\hat{\Theta}_G}$ is the unique greatest subautomaton of G that is both controllable in G and coreachable. If the state set Q is finite, the sequence $X^0 = Q$, $X^{i+1} = \Theta_G(X^i)$ reaches this fixed point in a finite number of steps, i.e., $\hat{\Theta}_G = X^n$ for some $n \geq 0$.

D. Translation of Specifications into Plants

A traditional supervisory control problem, see [9], consists of a *plant* G and a *specification* K, given as deterministic automata.

Using the nonblocking condition, control problems can be represented *equivalently* only using plants. A specification automaton is transformed into a plant by adding, for every uncontrollable event that is not enabled in a state, a transition to a new blocking state $\perp$. The following construction from [5] essentially transforms all potential controllability problems into potential blocking problems.

Definition 11: Let $K = \langle \Sigma, Q, \rightarrow, Q^i, Q^m \rangle$ be a specification. The *complete plant automaton* $K^\perp$ for K is

$$K^\perp = \langle \Sigma, Q \cup \{\perp\}, \rightarrow^\perp, Q^i, Q^m \rangle \tag{6}$$

where $\perp \notin Q$ is a new state and

$$\rightarrow^\perp = \rightarrow \cup \{ (x, \upsilon, \perp) \mid x \in Q, \upsilon \in \Sigma_u, x \overset{\upsilon}{\not\rightarrow} \}. \tag{7}$$

Proposition 1: Let G, K, and K' be deterministic automata over the same alphabet Σ, and let K' be reachable. Then the following two statements are equivalent.

(i) $K' \subseteq G \parallel K^\perp$ is nonblocking and controllable in $G \parallel K^\perp$,

(ii) $K' \subseteq G \parallel K$ is nonblocking and controllable with respect to G.

For the proof see [5]. According to this result, the least restrictive, controllable and nonblocking supervisor for plant G and specification K, can be obtained by calculating $\sup\mathcal{CN}(G \parallel K^\perp)$.

III. Compositional Synthesis

In this section, the proposed compositional synthesis algorithm is explained. The rules to calculate the abstracted automaton according to synthesis abstraction are given an proven.

A. Synthesis Abstraction

A modular supervisory control problem consists of a modular specification $K = K_1 \parallel \cdots \parallel K_m$ and a modular plant $G = G_1 \parallel \cdots \parallel G_n$. As discussed in Section II-D all the specifications can be translated to plants and therefore The task is to find the least restrictive supervisor S for a set of plants,

$$G \parallel K^\perp = G_1 \parallel \cdots \parallel G_n \parallel K_1^\perp \parallel \cdots \parallel K_m^\perp \ . \tag{8}$$

In the proposed algorithm, the modular system (8) is abstracted step by step, using the same strategies and heuristics that are proposed in [4]. Therefore instead of finding the supervisor for $G \parallel K$, each automaton G_i or $K_j^\perp$ in (8) may be abstracted and replaced by $G_i/\!\sim$ or $K_j^\perp/\!\sim$. When no more abstraction is possible, the synchronous composition is computed step by step, and in each step the abstraction rules are applied. Eventually this procedure leads to a single automaton H which is the abstracted description of the monolithic system (8). Once H is found, the next step is synthesise a supervisor for H, which is called S'. Finally the *modular supervisor* for the system is $S' \parallel K_1 \parallel K_2 \parallel \cdots \parallel K_m$, which is the least restrictive controllable and nonblocking supervisor for G. Note that since the supervisor is modular, the potential state-space explosion problem can be avoided. In practice the final supervisor $S' \parallel K$ never needs to be calculated. The final supervisor $S' \parallel K$ can instead be implemented keeping its modular structure and performing the synchronization on-line. As the plant generates events under control of the supervisor, the supervisor components accept and transit on those events individually. This effectively performs the synchronous composition on-line.

When abstracting an automaton G_i, in an attempt to replace it by $G_i/\!\sim$, there will typically be some events used in G_i which do not be appear in any other component $K_j^\perp$ or

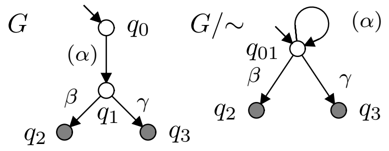

Fig. 1. No active event rule. Automaton G and its abstracted automaton $G/\!\sim$. Local events are shown by parentheses around.

G_j, $i \neq j$. These are called *local events* and will be denoted by Υ in the following. Local events are helpful to find an abstraction.

Definition 12: Let G and and $\tilde{G}$ be two deterministic automata with alphabet Σ_G. Then automaton $\tilde{G}$, is a *synthesis abstraction* of G with respect to $\Upsilon \subseteq \Sigma_G$, the set of local event, written $G \lesssim_{synth,\Upsilon} \tilde{G}$ if for every deterministic automaton $T = \langle \Sigma_T, Q_T, \rightarrow, Q_T^i, Q_T^m \rangle$ such that $\Sigma_T \cap \Upsilon = \emptyset$ the following holds

$$\mathcal{L}(G \parallel T \parallel \sup\mathcal{CN}(\tilde{G} \parallel T)) = \mathcal{L}(G \parallel T \parallel \sup\mathcal{CN}(G \parallel T)) \tag{9}$$

B. Abstraction Rules

In order to abstract the modular system, methods to find an abstracted automaton, of a given automaton are needed. Here some possible methods are discussed.

1) Bisimulation:

Definition 13: Assume $G_1 = \langle \Sigma, Q_1, \rightarrow, q_1^i, Q_1^m \rangle$ and $G_2 = \langle \Sigma, Q_2, \rightarrow, q_2^i, Q_2^m \rangle$ are two automata. A relation $\approx \subseteq Q_1 \times Q_2$ is called a bisimulation between G_1 and G_2 if, for all $x_1 \in Q_1$ and $x_2 \in Q_2$ and for all $\sigma \in \Sigma$ such that $x_1 \approx x_2$,

if $x_1 \xrightarrow{\sigma} y_1$ then $\exists y_2 \in Q_2$ such that $x_2 \xrightarrow{\sigma} y_2$ and $y_1 \approx y_2$,
if $x_2 \xrightarrow{\sigma} y_2$ then $\exists y_1 \in Q_1$ such that $x_1 \xrightarrow{\sigma} y_1$ and $y_1 \approx y_2$,
$x_1 \in Q_1^m$ if and only if $x_2 \in Q_2^m$.

G_1 and G_2 are bisimular if there exists a bisimulation $\approx$ between G_1 and G_2 such that $q_1^i \approx q_2^i$.

Theorem 2: Let G_1 and G_2 be two automata such that $G_1 \approx G_2$. Then $G_1 \lesssim_{synth,\emptyset} G_2$.

Proof: It must be shown that for every test automaton $T = \langle \Sigma_T, Q_T, \rightarrow_T, q_T^i, Q_T^m \rangle$, (9) holds. Since $G_1 \approx G_2$, it follows from the congruence result of [8] that $G_1 \parallel T \approx G_2 \parallel T$. By Lemma 2 in the appendix,

$$\sup\mathcal{CN}(G_1 \parallel T) \approx \sup\mathcal{CN}(G_2 \parallel T).$$

By congruence

$$G_1 \parallel T \parallel \sup\mathcal{CN}(G_1 \parallel T) \approx G_1 \parallel T \parallel \sup\mathcal{CN}(G_2 \parallel T),$$

which implies

$$\mathcal{L}(G_1 \parallel T \parallel \sup\mathcal{CN}(G_1 \parallel T)) = \mathcal{L}(G_1 \parallel T \parallel \sup\mathcal{CN}(G_2 \parallel T)).$$

■

2) No Active Event Rule: Two states that are connected by a local event such that the target state has no uncontrollable active events and also has the same or more outgoing transitions as the source state, can be merged. The abstracted automaton is a synthesis abstraction of the original automaton. Fig. 1 shows an example application of this rule.

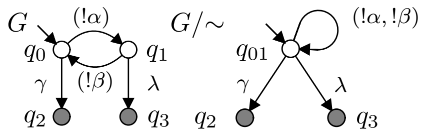

Fig. 2. Silent uncontrollable loop rule. Automaton G and its abstracted automaton $G/\sim$. ! denotes uncontrollable events and local events are shown by parentheses around.

Theorem 3: Let $G = \langle \Sigma, Q, \rightarrow, Q^i, Q^m \rangle$ be an automaton and $\Upsilon \subseteq \Sigma$ be the set of local events. Let $\sim \subseteq Q \times Q$ be an equivalence relation such that, for all $x_1, x_2 \in Q$ such that $x_1 \sim x_2$ it holds that, if $x_1 \xrightarrow{\gamma} x_2$ then $\gamma \in \Upsilon$, if $x_2 \xrightarrow{\sigma} x_3$ then $\sigma \in \Sigma_c$ and if $x_1 \xrightarrow{\sigma} x_3$ then $x_2 \xrightarrow{\sigma} x_3$. Then $G \lesssim_{synth,\Upsilon} G/\sim$.

Proof: It must be shown that for any deterministic automaton $T = \langle \Sigma_T, Q_T, \rightarrow_T, q_T^i, Q_T^m \rangle$ such that $\Sigma_T \cap \Upsilon = \emptyset$, (9) holds.

1) Let $s \in \mathcal{L}(G \parallel T \parallel \sup\mathcal{CN}(G \parallel T))$. This means that $G \parallel T \parallel \sup\mathcal{CN}(G \parallel T) \xrightarrow{s} (x_G, x_T, x'_G, x'_T)$ and since G and T are deterministic $x'_G = x_G$ and $x'_T = x_T$. Let $s = \sigma_1 \cdots \sigma_n$, then $(x_0^G, x_0^T) \xrightarrow{\sigma_1}_{|\hat{\Theta}_{G\parallel T}} (x_1^G, x_1^T) \xrightarrow{\sigma_2}_{|\hat{\Theta}_{G\parallel T}} \cdots \xrightarrow{\sigma_n}_{|\hat{\Theta}_{G\parallel T}} (x_n^G, x_n^T)$ and $(x_k^G, x_k^T) \in \text{gfp } \Theta_{G\parallel T}$ for $k = 0, ..., n$. By Lemma 3, $([x_k^G], x_k^T) \in \text{gfp } \Theta_{G/\sim\parallel T}$ for $k = 0, ..., n$ and $([x_0^G], x_0^T) \xrightarrow{\sigma_1}_{|\hat{\Theta}_{G/\sim\parallel T}} ([x_1^G], x_1^T) \xrightarrow{\sigma_2}_{|\hat{\Theta}_{G/\sim\parallel T}} \cdots \xrightarrow{\sigma_n}_{|\hat{\Theta}_{G/\sim\parallel T}} ([x_n^G], x_n^T)$. Therefore $G \parallel T \parallel \sup\mathcal{CN}(G/\sim \parallel T) \xrightarrow{s} (x_G, x_T, [x_G], x_T)$ which means that $s \in G \parallel T \parallel \sup\mathcal{CN}(G/\sim \parallel T)$.

(2) Let $s \in \mathcal{L}(G \parallel T \parallel \sup\mathcal{CN}(G/\sim \parallel T))$. This means that $G \parallel T \parallel \sup\mathcal{CN}(G/\sim \parallel T) \xrightarrow{s} (x_G, x_T, [x_G], x'_T)$, where $s = \sigma_1 \cdots \sigma_n$. Since T is deterministic, $x_T = x'_T$ and therefore $G \parallel T \parallel \sup\mathcal{CN}(G/\sim \parallel T) \xrightarrow{\sigma_1} (x_1^G, x_1^T, [x_1^G], x_1^T) \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} (x_n^G, x_n^T, [x_n^G], x_n^T)$. Since $([x_k^G], x_k^T) \in \text{gfp } \Theta_{G/\sim\parallel T}$ for $k = 0, ..., n$ by Lemma 3, $(x_k^G, x_k^T) \in \text{gfp } \Theta_{G\parallel T}$ for $k = 0, ..., n$. Therefore $G \parallel T \parallel \sup\mathcal{CN}(G \parallel T) \xrightarrow{\sigma_1} (x_1^G, x_1^T, x_1^G, x_1^T) \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} (x_n^G, x_n^T, x_n^G, x_n^T)$ and thus it can be concluded that $s \in \mathcal{L}(G \parallel T \parallel \sup\mathcal{CN}(G \parallel T))$. ■

3) Silent Uncontrollable Loop Rule: States in a local uncontrollable loop can be merged, and the deterministic abstracted automaton is a synthesis abstraction of the original automaton. Fig. 2 shows an example application of this rule.

Theorem 4: Let $G = \langle \Sigma, Q, \rightarrow, Q^i, Q^m \rangle$ be an automaton and let $\Upsilon \subseteq \Sigma_u$. Let $\sim \subseteq Q \times Q$ be an equivalence relation such that, for all $x_1, x_2 \in Q$ such that $x_1 \sim x_2$ it holds that, there exists $\upsilon \in \Upsilon^*$ such that $x_1 \xrightarrow{\upsilon} x_2$. Then $G \lesssim_{synth,\Upsilon} G/\sim$.

Proof: Same as the proof for Theorem 3, but using Lemma 4 instead of Lemma 3 in the appendix. ■

IV. Example

A simple manufacturing system [9] consists of two machines and a buffer. The first machine (M_1) starts processing workpieces (start$_1$) and puts them into the buffer (B) when it finishes (finish$_1$). In the beginning the buffer is empty and it becomes full after M_1 finishes. The second machine (M_2) removes the workpieces from the buffer (start$_2$) and completes the task (finish$_2$). Using a switch (W) can suspend (suspend) and resume (resume) production, and M_1 must not start if the switch is in suspend mode. The starting and repairing of machines are controllable events, while finishing, breakdown, suspend and resume are uncontrollable. In the case that both machines M_1 and M_2 are broken M_2 must be repaired first (R). Fig. 3 shows the automata of plants and the plantified specifications.

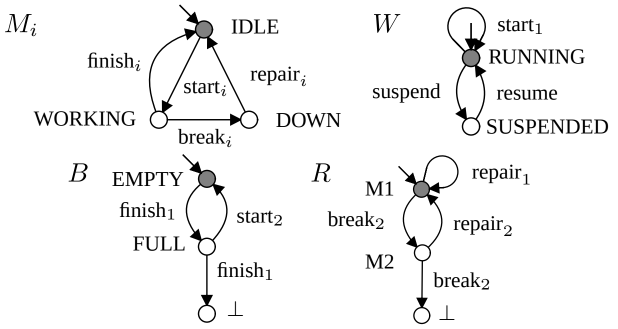

Fig. 3. Automata of small factory example.

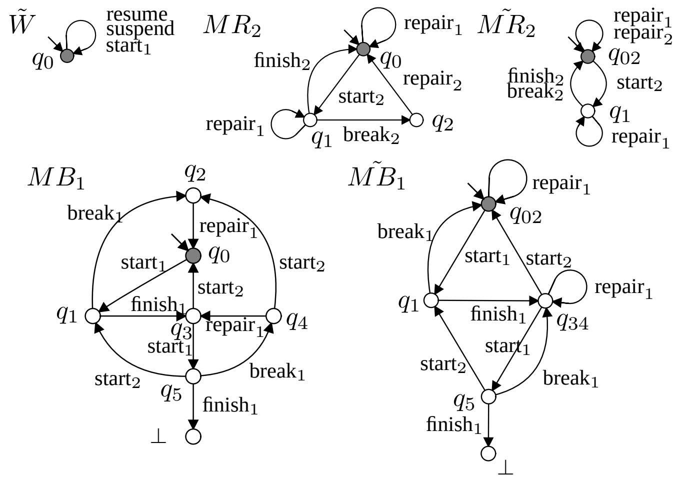

Fig. 4. Abstracted automata of small factory example.

First two events suspend and resume in W are uncontrollable local events therefore silent uncontrollable loop rule can be applied on W resulting in $\tilde{W}$ which is shown in Fig. 4. Since no more abstraction is possible some automata should be composed.

The composition of M_2 and R results in MR_2. Now controllable event repair$_2$ is a local event and no active event rule becomes applicable on MR_2. Merging q_0 and q_2 in MR_2 results in $\tilde{MR_2}$ which is shown in Fig. 4.

By composing B and M_1, automaton MB_1 can be obtained and repair$_1$ becomes a local event. Now no active event rule can be used to replace MB_1 by an abstracted automaton. By merging q_0 and q_2 first no active event rule can be applied one more time and q_3 and q_4 can be merged and the abstracted automaton $\tilde{MB_1}$ can be obtained. Fig. 4 shows MB_1 and also $\tilde{MB_1}$.

The last stage is to synthesise a supervisor for $\tilde{MB_1}$ and $\tilde{MR_2}$, which is S' and consists of 6 states. The modular supervisor for the system is $S' \parallel B \parallel R$. Composing the modular supervisor with the system results in the least restrictive monolithic supervisor for the system which consists of 24 states and is larger than S' which is the largest component of the modular supervisor.

V. CONCLUSIONS AND FUTURE WORKS

A new type of abstraction called synthesis abstraction is introduced and three rules are proposed to calculate the abstracted automaton of a given automaton. Using these rules, an algorithm for synthesizing a modular supervisor for large discrete event systems is proposed. This supervisor, in combination with the original specifications, produces the least restrictive controllable and nonblocking solution of the original control problem.

The proposed algorithm overcomes weaknesses of previous approaches to compositional synthesis. It results in the least restrictive supervisor for the system, without the need of an additional nonblocking check as in [7], or state labels as in [5].

In future work, the authors would like to develop more reduction rules. Presently, the abstraction rules apply only if the produced abstracted automata are deterministic. It would be an interesting research to consider nondeterminism after abstraction, which is likely to make more minimization possible.

REFERENCES

[1] K. Åkesson, H. Flordal, and M. Fabian. Exploiting modularity for synthesis and verification of supervisors. In *Proc. 15th IFAC World Congress on Automatic Control*, Barcelona, Spain, 2002.

[2] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer, September 1999.

[3] Lei Feng and W. M. Wonham. Computationally efficient supervisor design: Abstraction and modularity. In *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*, pages 3–8, Ann Arbor, MI, USA, July 2006.

[4] Hugo Flordal and Robi Malik. Modular nonblocking verification using conflict equivalence. In *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*, pages 100–106, Ann Arbor, MI, USA, July 2006.

[5] Hugo Flordal, Robi Malik, Martin Fabian, and Knut Åkesson. Compositional synthesis of maximally permissive supervisors using supervision equivalence. *Discrete Event Dyn. Syst.*, 17(4):475–504, 2007.

[6] Petra Malik, Robi Malik, David Streader, and Steve Reeves. Modular synthesis of discrete controllers. In *Proc. 12th IEEE Int. Conf. Engineering of Complex Computer Systems, ICECCS '07*, pages 25–34, Auckland, New Zealand, 2007.

[7] Robi Malik and Hugo Flordal. Yet another approach to compositional synthesis of discrete event systems. In *Proc. 9th Int. Workshop on Discrete Event Systems, WODES '08*, pages 16–21, Göteborg, Sweden, May 2008.

[8] Robin Milner. *Communication and concurrency*. Series in Computer Science. Prentice-Hall, 1989.

[9] Peter J. G. Ramadge and W. Murray Wonham. The control of discrete event systems. *Proc. IEEE*, 77(1):81–98, January 1989.

[10] Klaus Schmidt and Christian Breindl. On maximal permissiveness of hierarchical and modular supervisory control approaches for discrete event systems. In *Proc. 9th Int. Workshop on Discrete Event Systems, WODES '08*, pages 462–467, Göteborg, Sweden, May 2008.

[11] Raoguang Song and Ryan J. Leduc. Symbolic synthesis and verification of hierarchical interface-based supervisory control. In *Proc. 8th Int. Workshop on Discrete Event Systems, WODES '06*, pages 419–426, Ann Arbor, MI, USA, July 2006.

[12] Rong Su, Jan H. van Schuppen, and Jacobus E. Rooda. Model abstraction of nondeterministic finite-state automata in supervisor synthesis. *IEEE Trans. Automat. Contr.*, 55(11):2527–2541, November 2010.

[13] K. C. Wong and W. M. Wonham. Modular control and coordination of discrete-event systems. *Discrete Event Dyn. Syst.*, 8(3):247–297, October 1998.

APPENDIX

To simplify the proofs for the presented rules in III-B the following lemmas are used.

Lemma 1: Let $\approx$ be a bisimulation between G_1 and G_2 and let $x_1 \approx x_2$, states of G_1 and G_2 respectively. Then for all $n \geq 0$, $x_1 \in \Theta^n_{G_1}(Q_1) = X^n_1$ if and only if $x_2 \in \Theta^n_{G_2}(Q_2) = X^n_2$.

Proof: This can be proven by induction on n.

Base case. The base case holds since $x_1 \in Q_1 = \Theta^0_{G_1}(Q_1)$ and $x_2 \in Q_2 = \Theta^0_{G_2}(Q_2)$.

Inductive step. Assume the statement holds for $n \in \mathbb{N}$, i.e., for all $x_1 \approx x_2$, $x_1 \in X^n_1$ if and only if $x_2 \in X^n_2$. Now let $x_1 \in \Theta^{n+1}_{G_1}(Q_1)$. This implies $x_1 \in \Theta_{G_1}(X^n_1) = \Theta^{nonb}_{G_1}(X^n_1) \cap \Theta^{cont}_{G_1}(X^n_1)$.

$x_1 \in \Theta^{nonb}_{G_1}(X^n_1)$ means there exists $s = \sigma_1\sigma_2 \cdots \sigma_k \in \Sigma^*$ such that $x_1 = x^0_1 \xrightarrow{\sigma_1}_{|X^n_1} x^1_1 \xrightarrow{\sigma_2}_{|X^n_1} \cdots \xrightarrow{\sigma_k}_{|X^n_1} x^k_1 \in Q^m_1$. Since $x_1 \approx x_2$, there exists x^1_2 such that $x_2 = x^0_2 \xrightarrow{\sigma_1} x^1_2$ and $x^1_1 \approx x^1_2$, and by inductive assumption $x_2 = x^0_2 \xrightarrow{\sigma_1}_{|X^n_2} x^1_2$. By induction on k, it follows that $x_2 = x^0_2 \xrightarrow{\sigma_1}_{|X^n_2} x^1_2 \xrightarrow{\sigma_2}_{|X^n_2} \cdots \xrightarrow{\sigma_k}_{|X^n_2} x^k_2 \in Q^m_2$. Therefore it can be concluded that $x_2 \in \Theta^{nonb}_{G_2}(X^n_2)$.

Now assume $x_1 \in \Theta^{cont}_{G_1}(X^n_1)$. Let $\sigma \in \Sigma_u$ and $x_2 \xrightarrow{\sigma} y_2$. Since $x_1 \approx x_2$, thus there exists $y_1 \in Q_1$ such that $x_1 \xrightarrow{\sigma} y_1$ and $y_1 \approx y_2$. Since $x_1 \in \Theta^{cont}_{G_1}(X^n_1)$ therefore $\forall \sigma \in \Sigma_u$, $x_1 \xrightarrow{\sigma} y_1$ implies $y_1 \in X^n_1$. Since $y_1 \approx y_2$ by inductive assumption $y_2 \in X^n_2$. This implies $x_2 \in \Theta^{cont}_{G_2}(X^n_2)$. Thus $x_2 \in \Theta^{nonb}_{G_2}(X^n_2) \cap \Theta^{cont}_{G_2}(X^n_2) = \Theta^{n+1}_{G_2}(Q_2)$.

The second implication is analogous to the first one. ■

Lemma 2: Let G_1 and G_2 be two finite-state automata such that $G_1 \approx G_2$. Then $\sup\mathcal{CN}(G_1) \approx \sup\mathcal{CN}(G_2)$.

Proof: Assume $\approx$, be bisimulation between G_1 and G_2. Let $x_1 \xrightarrow{\sigma} y_1$ in $\sup\mathcal{CN}(G_1)$, then $x_1, y_1 \in \Theta^n_{G_1}(Q_1)$ for all $n \geq 0$ and $x_1 \xrightarrow{\sigma} y_1$. Since $G_1 \approx G_2$ there exists $y_2 \in Q_2$ such that $y_1 \approx y_2$ and $x_2 \xrightarrow{\sigma} y_2$. Since $y_1 \in \Theta^n_{G_1}(Q_1)$ for all $n \geq 0$, by Lemma 1, $y_2 \in \Theta^n_{G_2}(Q_2)$ for all $n \geq 0$, thus $x_2 \xrightarrow{\sigma} y_2$ in $\sup\mathcal{CN}(G_2)$.

The proof for the second condition of bisimulation is analogous to the first one.

The proof for the third condition of bisimulation follows immediately since $\approx$ is a bisimulation between G_1 and G_2, and marking in $\sup\mathcal{CN}(G_1)$ and $\sup\mathcal{CN}(G_2)$ is the same as marking in and G_1 and G_2. ■

Lemma 3: Let $G = \langle \Sigma, Q, \rightarrow, Q^i, Q^m \rangle$ and $T = \langle \Sigma_T, Q_T, \rightarrow_T, Q^i_T, Q^m_T \rangle$ be two automata. Let $\Upsilon \subseteq \Sigma$ be a set of local events for G, i.e. $\Sigma_T \cap \Upsilon = \emptyset$. Let $\sim\, \subseteq Q \times Q$ be an equivalence relation such that, for all $p, q \in Q$ such that $p \sim q$ it holds that, if $p \xrightarrow{\gamma} q$ then $\gamma \in \Upsilon$, if $q \xrightarrow{\sigma} r$ then $\sigma \in \Sigma_c$ and if $p \xrightarrow{\sigma} r$ then $q \xrightarrow{\sigma} r$. Then for all $x \in Q$ and for all $[x] \in Q/{\sim}$ the following two conditions hold

(i) if $(x, x_T) \in \text{gfp}\ \Theta_{G\|T}$, then $([x], x_T) \in \text{gfp}\ \Theta_{G/{\sim}\|T}$,

(ii) if $([x], x_T) \in \text{gfp}\ \Theta_{G/{\sim}\|T}$, then $(x, x_T) \in \text{gfp}\ \Theta_{G\|T}$,

Proof: *1)* Let $(x, x_T) \in \text{gfp}\ \Theta_{G\|T}$, it must be shown that $([x], x_T) \in \Theta^n_{G/{\sim}\|T}(Q/{\sim} \times Q_T) = \tilde{X}^n$ for all $n \geq 0$. This can be proven by induction on n.

Base case. $([x], x_T) \in Q/{\sim} \times Q_T \in \tilde{X}^0$.

Inductive step. Assume the statement holds for $n \in \mathbb{N}$, i.e., if $(x, y_T) \in \text{gfp}\ \Theta_{G\|T}$ then, $([x], x_T) \in \tilde{X}^n$. Now it must be shown that $([x], x_T) \in \Theta^{n+1}_{G/\sim\|T}(Q/\sim \times Q_T) = \Theta^{cont}_{G/\sim\|T}(\tilde{X}^n) \;\cap\; \Theta^{nonb}_{G/\sim\|T}(\tilde{X}^n)$.

Let $\sigma \in \Sigma_u$ and $([x], x_T) \xrightarrow{\sigma} ([y], y_T)$. Then $[x] \xrightarrow{\sigma} [y]$ and $x_T \xrightarrow{\sigma} y_T$. This implies that $x' \xrightarrow{\sigma} y'$ for some $x' \in [x]$, $y' \in [y]$ and $x', y' \in Q$. Since according to assumption the active events of states in $[p]$ are all controllable events, it can be concluded that $x = x'$. Thus $x \xrightarrow{\sigma} y'$ and $(x, x_T) \xrightarrow{\sigma} (y', y_T)$. If $x = p$ and $\sigma \in \Upsilon$ then since $\sigma \notin \Sigma_T$ we have $([x], x_T) = ([y], y_T)$. Since $(x, x_T) \in \text{gfp}\ \Theta_{G\|T}$ it follows that $(y', y_T) \in \text{gfp}\ \Theta_{G\|T}$. By inductive assumption $([y], y_T) = ([y'], y_T) \in \tilde{X}^n$. Therefore $([x], x_T) \in \Theta^{cont}_{G/\sim\|T}(\tilde{X}^n)$.

Since $(x, x_T) \in \text{gfp}\ \Theta_{G\|T}$, there exists a path $(x, x_T) = (x_0, x_0^T) \xrightarrow{\sigma_1}_{\hat{\Theta}_{G\|T}} (x_1, x_1^T) \xrightarrow{\sigma_2}_{\hat{\Theta}_{G\|T}} \cdots \xrightarrow{\sigma_k}_{\hat{\Theta}_{G\|T}} (x_k, x_k^T) \in Q^m \times Q_T^m$. Then $(x_l, x_l^T) \in \text{gfp}\ \Theta_{G\|T}$ for $l = 0, ..., k$. By inductive assumption $([x_l], x_l^T) \in \tilde{X}^n$ for $l = 0, ..., k$. Thus $([x], x_T) = ([x_0], x_0^T) \xrightarrow{\sigma_1}_{|\tilde{X}^n} \cdots \xrightarrow{\sigma_k}_{|\tilde{X}^n} ([x_k], x_k^T) \in Q^m/\sim \times Q_T^m$. Therefore $([x], x_T) \in \Theta^{nonb}_{G/\sim\|T}(\tilde{X}^n)$.

Therefore $([x], x_T) \in \Theta^{n+1}_{G/\sim\|T}(Q/\sim \times Q_T)$.

2) Let $([x], x_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$, it must be shown that $(x, x_T) \in \Theta^n_{G\|T}(Q \times Q_T) = X^n$ for all $n \geq 0$. This can be proven by induction.

Base case. $(x, x_T) \in Q \times Q_T = X^0$.

Inductive step. Assume the statement holds for $n \in \mathbb{N}$, i.e, if $([x], y_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$ then $(x, x_T) \in X^n$. Now it must be shown that $(x, x_T) \in \Theta^{n+1}_{G\|T}(Q \times Q_T) = \Theta^{cont}_{G\|T}(X^n) \;\cap\; \Theta^{nonb}_{G\|T}(X^n)$.

Let $\sigma \in \Sigma_u$ and $(x, x_T) \xrightarrow{\sigma} (y, y_T)$. Then $x \xrightarrow{\sigma} y$ and $x_T \xrightarrow{\sigma} y_T$. This implies that $[x] \xrightarrow{\sigma} [y]$ and $([x], x_T) \xrightarrow{\sigma} ([y], y_T)$. Since $([x], x_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$ and $\sigma \in \Sigma_u$, therefore $([y], y_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$. By inductive assumption $(y, y_T) \in X^n$ and thus $(x, x_T) \in \Theta^{cont}_{G\|T}(X^n)$.

Since $([x], x_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$, there exists a path $([x], x_T) = ([x_0], x_0^T) \xrightarrow{\sigma_1}_{\hat{\Theta}_{G/\sim\|T}} ([x_1], x_1^T) \xrightarrow{\sigma_2}_{\hat{\Theta}_{G/\sim\|T}} \cdots \xrightarrow{\sigma_k}_{\hat{\Theta}_{G/\sim\|T}} ([x_k], x_k^T) \in Q^m/\sim \times Q_T^m$ and without loss of generality the selfloop $[p] \xrightarrow{\gamma} [p] = [q]$ is not in the path. Clearly $([x_l], x_l^T) \in \text{gfp}\ \Theta_{G/\sim\|T}$ for $l = 0, ..., k$. By inductive assumption $(x_l', x_l^T) \in X^n$ for $l = 0, ..., k$ and for all $x_l' \in [x_l]$. Since $[x_0] \xrightarrow{\sigma_1} [x_1]$, there exists $x_0' \in [x_0]$ and $x_1'' \in [x_1]$ such that $x_0' \xrightarrow{\sigma_1} x_1''$. Since $[x_1] \xrightarrow{\sigma_2} [x_2]$, there exists $x_1' \in [x_1]$ and $x_2'' \in [x_2]$ such that $x_1' \xrightarrow{\sigma_2} x_2''$, and so on. Since $x_1', x_1'' \in [x_1]$, there are three possibilities,

(i) $x_1' = x_1''$, then $x_0' \xrightarrow{\sigma_1} x_1'' = x_1' \xrightarrow{\sigma_2} x_2''$,

(ii) $x_1'' = p$ and $x_1' = q$. Since $p \xrightarrow{\beta} q$, also $x_0' \xrightarrow{\sigma_1} x_1'' = p \xrightarrow{\beta} q = x_1' \xrightarrow{\sigma_2} x_2''$,

(iii) $x_1' = p$ and $x_1'' = q$. Since $p = x_1' \xrightarrow{\sigma_2} x_2''$, according to the assumption $q = x_1'' \xrightarrow{\sigma_2} x_2''$. Thus $x_0' \xrightarrow{\sigma_1} x_1'' \xrightarrow{\sigma_2} x_2''$.

Also since $([x_k], x_T) \in Q^m/\sim \times Q_T^m$ then $x_k' = x_k''$. By induction, it can be shown that

(i) $(x_0', x_0^T) \xrightarrow{\sigma_1} (x_1'', x_1^T) = (x_1', x_1^T) \xrightarrow{\sigma_2} (x_2'', x_2^T) \cdots \xrightarrow{\sigma_k} (x_k', x_k^T)$, and $(x_k', x_k^T) \in Q^m \times Q_T^m$,

(ii) $(x_0', x_0^T) \xrightarrow{\sigma_1} (x_1'', x_1^T) \xrightarrow{\beta} (x_1', x_1^T) \xrightarrow{\sigma_2} (x_2'', x_2^T) \cdots \xrightarrow{\sigma_k} (x_k', x_k^T)$, and $(x_k', x_k^T) \in Q^m \times Q_T^m$,

(iii) $(x_0', x_0^T) \xrightarrow{\sigma_1} (x_1'', x_1^T) \xrightarrow{\sigma_2} (x_2'', x_2^T) \cdots \xrightarrow{\sigma_k} (x_k', x_k^T)$, and $(x_k', x_k^T) \in Q^m \times Q_T^m$.

Thus $(x, x_T) \in \Theta^{nonb}_{G\|T}(X^n)$ and therefore $(x, x_T) \in \Theta^{n+1}_{G\|T}(Q \times Q_T)$. ■

Lemma 4: Let $G = \langle \Sigma, Q, \rightarrow, Q^i, Q^m \rangle$ be an automaton and let $\Upsilon \subseteq \Sigma_u$. Let $\sim \subseteq Q \times Q$ be an equivalence relation such that, for all $x_1, x_2 \in Q$ such that $x_1 \sim x_2$ it holds that, there exists $\upsilon \in \Upsilon^*$ such that $x_1 \xrightarrow{\upsilon} x_2$. Then the following two conditions hold,

(i) if $(x, x_T) \in \text{gfp}\ \Theta_{G\|T}$, then $([x], x_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$,

(ii) if $([x], x_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$, then $(x, x_T) \in \text{gfp}\ \Theta_{G\|T}$.

Proof: 1) Let $(x, x_T) \in \text{gfp}\ \Theta_{G\|T}$, it must be shown that $([x], x_T) \in \Theta^n_{G/\sim\|T}(Q/\sim \times Q_T) = \tilde{X}^n$ for all $n \geq 0$. This can be proven by induction on n.

Base case. $([x], x_T) \in Q/\sim \times Q_T \in \tilde{X}^0$.

Inductive step. Assume the statement holds for $n \in \mathbb{N}$, i.e., if $(x, y_T) \in \text{gfp}\ \Theta_{G\|T}$ then $([x], x_T) \in \tilde{X}^n$. Now it must be shown that $([x], x_T) \in \Theta^{n+1}_{G/\sim\|T}(Q/\sim \times Q_T) = \Theta^{cont}_{G/\sim\|T}(\tilde{X}^n) \;\cap\; \Theta^{nonb}_{G/\sim\|T}(\tilde{X}^n)$.

Let $\sigma \in \Sigma_u$ and $([x], x_T) \xrightarrow{\sigma} ([y], y_T)$. Then $[x] \xrightarrow{\sigma} [y]$ and $x_T \xrightarrow{\sigma} y_T$. This implies that $x' \xrightarrow{\sigma} y'$ for some $x' \in [x]$, $y' \in [y]$ and $x', y' \in Q$. Since $x' \in [x]$, it holds that $x \xrightarrow{u} x'$ for some $u \in \Upsilon$. Therefore $(x, x_T) \xrightarrow{u} (x', x_T) \xrightarrow{\sigma} (y', y_T)$. Since $(x, x_T) \in \text{gfp}\ \Theta_{G\|T}$ and $u\sigma \in \Sigma_u^*$ it follows that $(y', y_T) \in \text{gfp}\ \Theta_{G\|T}$. By inductive assumption $([y], y_T) = ([y'], y_T) \in \tilde{X}^n$. Therefore $([x], x_T) \in \Theta^{cont}_{G/\sim\|T}(\tilde{X}^n)$.

For blocking the same proof as in Lemma 3 holds here.

Therefore it can be concluded that $([x], x_T) \in \Theta^{cont}_{G/\sim\|T}(\tilde{X}^n) \;\cap\; \Theta^{nonb}_{G/\sim\|T}(\tilde{X}^n)$.

2) Let $([x], x_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$, it must be shown that $(x, x_T) \in \Theta^n_{G\|T}(Q \times Q_T) = X^n$ for all $n \geq 0$. This can be proven by induction.

Base case. $(x, x_T) \in Q \times Q_T \in X^0$.

Inductive step. Assume the statement holds for $n \in \mathbb{N}$, i.e., if $([x], y_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$ then $(x, x_T) \in X^n$. Now it must be shown that $(x, x_T) \in \Theta^{n+1}_{G\|T}(Q \times Q_T) = \Theta^{cont}_{G\|T}(X^n) \;\cap\; \Theta^{nonb}_{G\|T}(X^n)$.

For controllability the same proof as in 3 holds here.

Since $([x], x_T) \in \text{gfp}\ \Theta_{G/\sim\|T}$, there exists a path $([x], x_T) = ([x_0], x_0^T) \xrightarrow{\sigma_1}_{|\hat{\Theta}_{G/\sim\|T}} ([x_1], x_1^T) \xrightarrow{\sigma_2}_{|\hat{\Theta}_{G/\sim\|T}} \cdots \xrightarrow{\sigma_k}_{|\hat{\Theta}_{G/\sim\|T}} ([x_k], x_k^T) \in Q^m/\sim \times Q_T^m$. Then $([x_l], x_l^T) \in \text{gfp}\ \Theta_{G/\sim\|T}$ for $l = 0, ..., k$. By inductive assumption $(x_l', x_l^T) \in X^n$ for $l = 0, ..., k$ and for all $x_l' \in [x_l]$. Since $[x_0] \xrightarrow{\sigma_1} [x_1]$, there exists $x_0' \in [x_0]$ and $x_1'' \in [x_1]$ such that $x_0' \xrightarrow{\sigma_1} x_1''$. Since $[x_1] \xrightarrow{\sigma_2} [x_2]$, there exists $x_1' \in [x_1]$ and $x_2'' \in [x_2]$ such that $x_1' \xrightarrow{\sigma_2} x_2''$, and so on. Since $x_1', x_1'' \in [x_1]$ there exists $u_1 \in \Upsilon$ such that $x_1'' \xrightarrow{u_1}_{|X^n} x_1'$. Also since $x, x_0' \in [x_0]$ there exists $u_0 \in \Upsilon$ such that $x \xrightarrow{u_0}_{|X^n} x_0'$. Also since $x_k'' \in [x_k]$ and $[x_k] \in Q^m/\sim$ there exists $x_k' \in [x_k]$ such that $x_k' \in Q^m/\sim$ and $x_k'' \xrightarrow{u_k}_{|X^n} x_k'$ for some $u_k \in \Upsilon$. Therefore $(x, x_T) = (x_0, x_0^T) \xrightarrow{u_0}_{|X^n} (x_0', x_0^T) \xrightarrow{\sigma_1}_{|X^n} (x_1'', x_1^T)) \xrightarrow{u_1}_{|X^n} (x_1', x_1^T) \xrightarrow{\sigma_2}_{|X^n} \cdots \xrightarrow{\sigma_k}_{|X^n} (x_k'', x_k^T) \xrightarrow{u_k}_{|X^n} (x_k', x_k^T) \in Q^m \times Q_T^m$. Therefore $(x, x_T) \in \Theta^{nonb}_{G\|T}(X^n)$.

Therefore the claim is proven. ■