Team automata : a formal approach to the modeling of collaboration between system components

Beek, M.H. ter

4. Synchronized Automata

In the previous chapter we have introduced automata as the basic components underlying team automata. In this chapter we define precisely how *automata* can be combined in order to form a *synchronized automaton*. Within such a synchronized automaton its constituting automata interact by synchronizing on certain occurrences of shared actions. We also define how to obtain a *subautomaton* from a synchronized automaton by focusing on a subset of its constituting automata, and we study the relation between synchronized automata and their subautomata in terms of computations. Consequently, we show how to iteratively obtain synchronized automata from synchronized automata.

We then characterize three basic and natural ways of synchronizing. We also define *maximal-syn synchronized automata* as the unique synchronized automata being maximal with respect to a given type of synchronization *syn*. Through the formulation of *predicates of synchronization* we furthermore provide direct descriptions of such synchronized automata. Finally, we conclude this chapter with a study of the effect that synchronizations have on the inheritance of the automata-theoretic properties introduced in Section 3.2 from synchronized automata to their (sub)automata, and vice versa.

Notation 1. *In this chapter we assume a fixed, but arbitrary and possibly infinite index set $\mathcal{I} \subseteq \mathbb{N}$, which we will use to index the automata involved. For each $i \in \mathcal{I}$, we let $\mathcal{A}_i = (Q_i, \Sigma_i, \delta_i, I_i)$ be a fixed automaton. Moreover, we let $\mathcal{S} = \{\mathcal{A}_i \mid i \in \mathcal{I}\}$ be a fixed set of automata. Note that $\mathcal{I} \subseteq \mathbb{N}$ implies that $\mathcal{I}$ is ordered by the usual $\leq$ relation on $\mathbb{N}$, thus inducing an ordering on $\mathcal{S}$. Also note that the $\mathcal{A}_i$ are not necessarily different.* □

4.1 Definitions

We begin this section by defining synchronized automata as composite automata. Consequently, we consider also the dual approach by defining how to extract (sub)automata from a given synchronized automaton.

4.1.1 Synchronized Automata

Consider the set $\mathcal{S} = \{\mathcal{A}_i \mid i \in \mathcal{I}\}$ of automata, as fixed above. Then a state q of any synchronized automaton over $\mathcal{S}$ describes the states that each of the automata is in. The state space of any synchronized automaton $\mathcal{T}$ formed from $\mathcal{S}$ is thus the product $\prod_{i\in\mathcal{I}} Q_i$ of the state spaces of the automata of $\mathcal{S}$, with the product $\prod_{i\in\mathcal{I}} I_i$ of their initial states forming the set of initial states of $\mathcal{T}$.

The transition relation of such $\mathcal{T}$ is defined by allowing certain "synchronizations" and excluding others and is based solely on the transition relations of the automata forming the synchronized automaton.

Definition 4.1.1. *Let $a \in \bigcup_{i\in\mathcal{I}} \Sigma_i$. Then the* complete transition space *of a in $\mathcal{S}$ is denoted by $\Delta_a(\mathcal{S})$ and is defined as*

$$\Delta_a(\mathcal{S}) = \{(q,q') \in \prod_{i\in\mathcal{I}} Q_i \times \prod_{i\in\mathcal{I}} Q_i \mid \exists j \in \mathcal{I} :\ proj_j{}^{[2]}(q,q') \in \delta_{j,a} \wedge (\forall i \in \mathcal{I} :\ proj_i{}^{[2]}(q,q') \in \delta_{i,a} \vee proj_i(q) = proj_i(q'))\}.$$ □

The complete transition space $\Delta_a(\mathcal{S})$ thus consists of all possible combinations of a-transitions from automata of $\mathcal{S}$, with all non-participating automata remaining idle. It is an explicit requirement that at least one automaton is active, i.e. executes an a-transition. The transitions in $\Delta_a(\mathcal{S})$ are referred to as *synchronizations* (on a).

This $\Delta_a(\mathcal{S})$ is called the *complete* transition space of a in $\mathcal{S}$ because whenever a synchronized automaton $\mathcal{T}$ is constructed from $\mathcal{S}$, then for each action a, all a-transitions of $\mathcal{T}$ come from $\Delta_a(\mathcal{S})$. The transformation of a state of $\mathcal{T}$ is defined by the local state changes of the automata participating in the action of $\mathcal{T}$ being executed. When defining $\mathcal{T}$, for each action a, a specific subset δ_a of $\Delta_a(\mathcal{S})$ has to be chosen. By restricting the set of allowed transitions in this way, a certain kind of interaction between the automata constituting the synchronized automaton can be enforced.

Definition 4.1.2. *A* synchronized automaton over $\mathcal{S}$ *is a construct* $\mathcal{T} = (Q, \Sigma, \delta, I)$*, where*

$Q = \prod_{i\in\mathcal{I}} Q_i$,
$\Sigma = \bigcup_{i\in\mathcal{I}} \Sigma_i$,
$\delta \subseteq Q \times \Sigma \times Q$ *is such that for all* $a \in \Sigma$,

$\delta_a \subseteq \Delta_a(\mathcal{S})$, *and*

$I = \prod_{i\in\mathcal{I}} I_i$. □

All synchronized automata over a given set of automata thus have the same set of states, the same alphabet of actions, and the same set of initial states. They only differ by the choice of their transition relation, which is based on but not fixed by the transition relations of the individual automata. Due to this freedom of choosing a δ_a for each action a, a set of automata does not uniquely define a single synchronized automaton. Instead, a flexible framework is provided within which one can construct a variety of synchronized automata, all of which differ solely by the choice of the transition relation.

In the literature, automata are mostly composed according to some fixed strategy, thus leading to a uniquely defined synchronized automaton. In fact, the strategy that is prevalent in the literature (cf. the Introduction) is the rule to include, for all actions a, all and only those a-transitions in which all automata from $\mathcal{S}$ participate that have a as one of their actions. This leaves no choice for the transition relation and thus leads to a unique synchronized automaton. In Section 4.5 we will describe this and other fixed strategies for choosing transition relations in a predetermined way. Within our framework, however, it is precisely the freedom to choose transition relations which provides the flexibility to distinguish even the smallest nuances in the meaning of one's design.

The following example illustrates the definition of synchronized automata. Recall that vectors may be written vertically, even though in the text they are written horizontally.

Example 4.1.3. (Example 3.1.8 continued) Consider the automaton $W_2 = (\{s_2, t_2\}, \{a, b\}, \delta_2, \{s_2\})$, with $\delta_2 = \{(s_2, b, s_2), (s_2, a, t_2), (t_2, a, t_2), (t_2, b, s_2)\}$, modeling the second **w**heel of a car. Since W_2 in essence is just a copy of W_1 its structure is the same as that of W_1, depicted in Figure 3.1.

Now we show how W_1 and W_2 can form a synchronized automaton (an axle). The synchronized automaton $\mathcal{T}_{\{1,2\}}$ over $\{W_1, W_2\}$ is depicted in Figure 4.1(a). It has four states of which (s_1, s_2) is its only initial state. It has no other actions than a and b. We require the two wheels W_1 and W_2 to accelerate and break in unison, so we choose $\delta_{\{1,2\}} = \{((s_1, s_2), b, (s_1, s_2)), ((s_1, s_2), a, (t_1, t_2)), ((t_1, t_2), a, (t_1, t_2)), ((t_1, t_2), b, (s_1, s_2))\}$. We note that only the transition relation had to be chosen, all other elements follow from Definition 4.1.2.

Note that $\mathcal{T}_{\{1,2\}}$ is action reduced and transition reduced but not state reduced, since its states (s_1, t_2) and (t_1, s_2) are not reachable.

By choosing a different transition relation such as, e.g., $\delta'_{\{1,2\}} = \{((s_1, s_2), a, (s_1, t_2)), ((t_1, t_2), b, (s_1, s_2))\}$, another synchronized automaton over $\{W_1, W_2\}$ is defined, which we denote by $\mathcal{T}'_{\{1,2\}}$. Apart from its transition relation, $\mathcal{T}'_{\{1,2\}}$ contains the same elements as $\mathcal{T}_{\{1,2\}}$. $\mathcal{T}'_{\{1,2\}}$ is depicted in Figure 4.1(b).

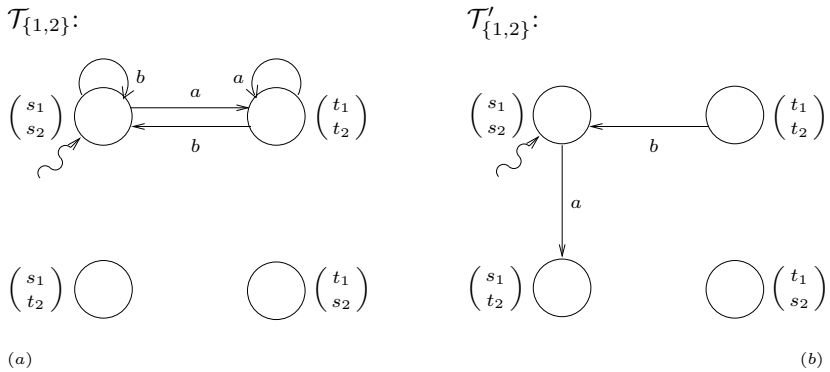

Fig. 4.1. Synchronized automata $\mathcal{T}_{\{1,2\}}$ and $\mathcal{T}'_{\{1,2\}}$.

If we assume that a flat tire is modeled by a wheel that cannot accelerate, then in $\mathcal{T}'_{\{1,2\}}$ the wheel W_1 has a flat tire. $\mathcal{T}'_{\{1,2\}}$ ends up in a deadlock (i.e. in a state where no action is enabled) after the execution of a, since one doesn't drive far with a flat tire. Furthermore, $\mathcal{T}'_{\{1,2\}}$ is not even action reduced nor is it transition reduced, because action b can never be executed in $\mathcal{T}'_{\{1,2\}}$ due to the fact that state (t_1, t_2) is not reachable. □

Definition 4.1.2 immediately implies the following result.

Theorem 4.1.4. *Every synchronized automaton is an automaton.* □

Since every synchronized automaton is again an automaton, it could in its turn be used as a constituting automaton of a new synchronized automaton.

Note, however, that even though a synchronized automaton over just one automaton $\{\mathcal{A}_j\}$ is again an automaton, such a synchronized automaton is different from its only constituting automaton. Even when Q_j and $\prod_{\{j\}} Q_j$ are identified, the transition relation of the synchronized automaton may be properly included in the transition relation of the automaton. This is due to the fact that the freedom in choosing the transition relation of a synchronized automaton, allows one to omit transitions from $\mathcal{A}_j$ in the transition relation of a synchronized automaton over $\{\mathcal{A}_j\}$.

Example 4.1.5. (Example 4.1.3 continued) We now show how to form a synchronized automaton (a car) over three automata (an axle and two wheels).

For $i \in \{3, 4\}$, let $W_i = (\{s_i, t_i\}, \{a, b\}, \delta_i, \{s_i\})$, where $\delta_i = \{(s_i, b, s_i), (s_i, a, t_i), (t_i, a, t_i), (t_i, b, s_i)\}$, be two automata modeling the third and the fourth **w**heel of a car. Since W_3 and W_4 (like W_2) are in essence just copies of W_1, their structure is the same as that of W_1, depicted in Figure 3.1.

Any synchronized automaton over $\{\mathcal{T}_{\{1,2\}}, W_3, W_4\}$ has alphabet $\{a, b\}$ and 16 states, among which the initial state $((s_1, s_2), s_3, s_4)$. We choose synchronized automaton $\hat{\mathcal{T}}$ by defining $\hat{\delta} = \{(((s_1, s_2), s_3, s_4), b, ((s_1, s_2), s_3, s_4)), (((s_1, s_2), s_3, s_4), a, ((t_1, t_2), t_3, t_4)), (((t_1, t_2), t_3, t_4), a, ((t_1, t_2), t_3, t_4)), (((t_1, t_2), t_3, t_4), b, ((s_1, s_2), s_3, s_4))\}$ as its transition relation. Its state-reduced version $\hat{\mathcal{T}}_S$ is depicted in Figure 4.2. □

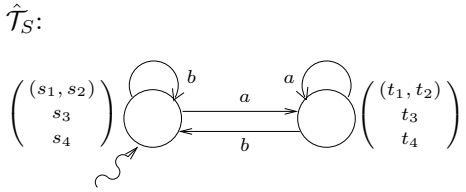

Fig. 4.2. State-reduced synchronized automaton $\hat{\mathcal{T}}_S$.

We conclude this section with two additional observations.

First it should be noted that in the definition of a synchronized automaton over $\mathcal{S} = \{\mathcal{A}_i \mid i \in \mathcal{I}\}$ we have implicitly used the ordering on $\mathcal{S}$ induced by $\mathcal{I}$. Every synchronized automaton over $\mathcal{S}$ has $\prod_{i \in \mathcal{I}} Q_i$ as its set of states and thus, if $\mathcal{I} = \{i_1, i_2, \dots\}$ with $i_1 < i_2 < \cdots$, then every state q of $\mathcal{T}$ is a tuple $(q_1, q_2, \dots)$ with $q_j \in Q_{i_j}$ for $j \geq 1$. This is convenient in concrete situations, but note that changing the order of the automata in $\mathcal{S}$ leads to formally different state spaces. As an example, consider two automata $\mathcal{A}_4$ and $\mathcal{A}_7$ with sets of states Q_4 and Q_7, respectively. Let $\mathcal{S} = \{\mathcal{A}_i \mid i \in \{4, 7\}\}$ and let $\mathcal{S}' = \{D_j \mid j \in \{1, 2\}\}$ with $D_1 = \mathcal{A}_7$ and $D_2 = \mathcal{A}_4$. Synchronized automata over $\mathcal{S}$ have $Q_4 \times Q_7$ as their state space, whereas synchronized automata over $\mathcal{S}'$ have $Q_7 \times Q_4$ as their state space. In Section 4.3 we will come back to the ordering within state spaces in a more general setup.

Secondly, neither in the definition of an automaton nor in the definition of a synchronized automaton, have we required a priori that states have to be reachable, that actions have to be active, or that transitions have to be useful in at least one computation starting from the initial state of the system. The lack of such extra conditions allows for a smooth and general definition of a synchronized automaton, with the full cartesian product of the sets of states of its constituting automata as the synchronized automaton's state space, the full union of the sets of actions of its constituting automata as its alphabet of actions, and an arbitrary selection of synchronizations as its transitions. Moreover, recall that in general no effective procedures exist

to obtain the reduced versions of synchronized automata defined in Definitions 3.2.8, 3.2.9, and 3.2.27.

4.1.2 Subautomata

Given a synchronized automaton $\mathcal{T}$ over $\mathcal{S}$, by focusing on a subset of the automata in $\mathcal{S}$, a subautomaton within $\mathcal{T}$ can be distinguished. Its transitions are restrictions of the transitions of $\mathcal{T}$ to the automata in the subset, while its actions of course are the actions of these automata.

Definition 4.1.6. *Let $\mathcal{T} = (Q, \Sigma, \delta, I)$ be a synchronized automaton over $\mathcal{S}$ and let $J \subseteq \mathcal{I}$. Then the* subautomaton of $\mathcal{T}$ determined by J *is denoted by $SUB_J(\mathcal{T})$ and is defined as $SUB_J(\mathcal{T}) = (Q_J, \Sigma_J, \delta_J, I_J)$, where*

$Q_J = \prod_{j \in J} Q_j$,
$\Sigma_J = \bigcup_{j \in J} \Sigma_j$,
$\delta_J \subseteq Q_J \times \Sigma_J \times Q_J$ *is such that for all $a \in \Sigma_J$,*

$$(\delta_J)_a = proj_J^{[2]}(\delta_a) \cap \Delta_a(\{\mathcal{A}_j \mid j \in J\}), \text{ and}$$

$I_J = \prod_{j \in J} I_j$. □

We write SUB_J instead of $SUB_J(\mathcal{T})$ if the synchronized automaton $\mathcal{T}$ is clear from the context. In Figure 4.3 we have sketched a subautomaton of a synchronized automaton.

The transition relation of a subautomaton SUB_J of a synchronized automaton $\mathcal{T}$ (over $\mathcal{S}$) determined by some $J \subseteq \mathcal{I}$, is obtained by restricting the transition relation of $\mathcal{T}$ to synchronizations between the automata in $\{\mathcal{A}_j \mid j \in J\}$. Hence in each transition of the subautomaton at least one of the automata from $\{\mathcal{A}_j \mid j \in J\}$ is actively involved. This is formalized by the intersection of $\text{proj}_J^{[2]}(\delta_a)$ with $\Delta_a(\{\mathcal{A}_j \mid j \in J\})$, for each action a, as in each transition in this complete transition space at least one automaton from $\{\mathcal{A}_j \mid j \in \mathcal{J}\}$ is active.

Note that if $J = \varnothing$, then SUB_J is the trivial automaton.

Example 4.1.7. (Example 4.1.5 continued) Subautomaton $SUB_{\{1\}}(\mathcal{T}_{\{1,2\}}) = (\{(s_1), (t_1)\}, \{a, b\}, \delta_{\{1\}}, \{(s_1)\})$, where $\delta_{\{1\}} = \{((s_1), b, (s_1)), ((s_1), a, (t_1)), ((t_1), a, (t_1)), ((t_1), b, (s_1))\}$, is depicted in Figure 4.4(a).

Note that $SUB_{\{1\}}(\mathcal{T}_{\{1,2\}})$ differs from W_1 in the sense that it has (s_1) and (t_1) as states rather than s_1 and t_1. Obviously, $SUB_{\{1\}}(\mathcal{T}_{\{1,2\}})$ and W_1 do exhibit the same behavior.

$\mathcal{T}$ over $\mathcal{S} = \{\mathcal{A}_i \mid i \in \mathcal{I}\}$ with $\mathcal{I} = [n]$ for some even $n \geq 1$

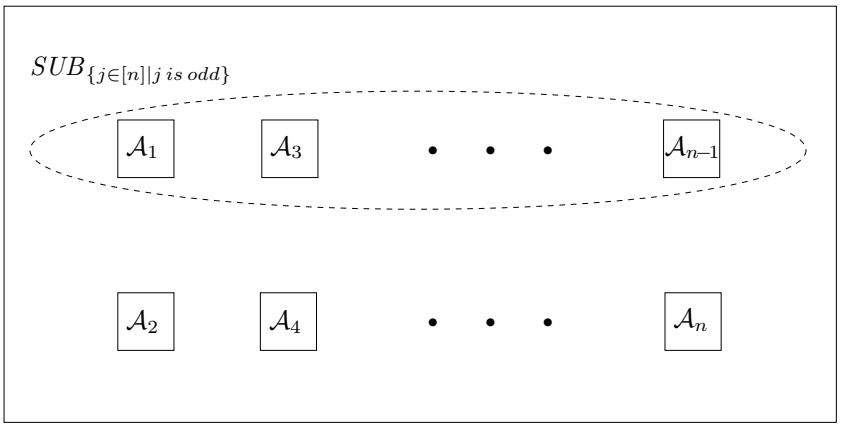

Fig. 4.3. Subautomaton $SUB_{\{j\in[n]|j\ is\ odd\}}$ of synchronized automaton $\mathcal{T}$.

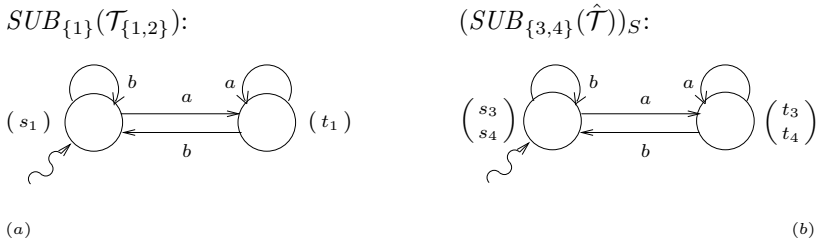

Fig. 4.4. Subautomaton $SUB_{\{1\}}(\mathcal{T}_{\{1,2\}})$ and automaton $(SUB_{\{3,4\}}(\hat{\mathcal{T}}))_S$.

Subautomaton $SUB_{\{3,4\}}(\hat{\mathcal{T}}) = (\{(s_3,s_4),(s_3,t_4),(t_3,s_4),(t_3,t_4)\},\{a,b\}, \hat{\delta}_{\{3,4\}}, \{(s_3,s_4)\})$, where $\hat{\delta}_{\{3,4\}} = \{((s_3,s_4),b,(s_3,s_4)),((s_3,s_4),a,(t_3,t_4)), ((t_3,t_4),a,(t_3,t_4)),((t_3,t_4),b,(s_3,s_4))\}$, has as its state-reduced version the automaton $(SUB_{\{3,4\}}(\hat{\mathcal{T}}))_S$ depicted in Figure 4.4(b). □

It is not hard to see that subautomata satisfy the requirements of a synchronized automaton.

Theorem 4.1.8. *Let $\mathcal{T} = (Q,\Sigma,\delta,I)$ be a synchronized automaton over $\mathcal{S}$ and let $J \subseteq \mathcal{I}$. Then*

SUB_J is a synchronized automaton over $\{\mathcal{A}_j \mid j \in J\}$.

Proof. The states, alphabet, and initial states of SUB_J as given in Definition 4.1.6 satisfy the requirements of Definition 4.1.2 for synchronized automata over $\{\mathcal{A}_j \mid j \in J\}$. Finally, $(\delta_J)_a \subseteq \Delta_a(\{\mathcal{A}_j \mid j \in J\})$ by Definition 4.1.6. □

According to this theorem a subautomaton of a synchronized automaton is again a synchronized automaton and thus, by Theorem 4.1.4, also an automaton. In Section 4.3 we will consider the dual approach and use synchronized automata as automata in "larger" synchronized automata. It will be shown that subautomata can be used as automata to iteratively define the synchronized automaton they are derived from.

We conclude this section by comparing the set of transitions and computations of a singleton subautomaton $SUB_{\{j\}}$ of a synchronized automaton $\mathcal{T}$ over $\mathcal{S}$ with those of the single automaton $\mathcal{A}_j$ from $\mathcal{S}$, where $j \in \mathcal{I}$. Due to the fact that $SUB_{\{j\}}$ has vectors (of one element) as states, whereas $\mathcal{A}_j$ does not, $SUB_{\{j\}}$ never equals $\mathcal{A}_j$ (see, e.g., Example 4.1.7). This is a purely syntactic reason, though. Therefore, in order to compare the set of transitions and computations of $\mathcal{A}_j$ with those of $SUB_{\{j\}}$, we identify $\prod_{\{j\}} Q_j$ and Q_j. To this end we define, for $j \in \mathcal{I}$, the homomorphism $\upsilon_j : (\Sigma \cup \prod_{\{j\}} Q_j)^\infty \to (\Sigma \cup Q_j)^\infty$ by

$$\upsilon_j(x) = \begin{cases} x & \text{if } x \in \Sigma \text{ and} \\ \mathrm{proj}_j(x) & \text{if } x \in \prod_{\{j\}} Q_j. \end{cases}$$

Consequently, we now show that for all $j \in \mathcal{I}$, the set of transitions (computations) of the subautomaton $SUB_{\{j\}}$ of a synchronized automaton $\mathcal{T}$ over $\mathcal{S}$ is included in the set of transitions (computations) of the single automaton $\mathcal{A}_j$ from $\mathcal{S}$. However, as shown in the example directly following this result, these inclusions can be proper.

Lemma 4.1.9. *Let $\mathcal{T} = (Q, \Sigma, \delta, I)$ be a synchronized automaton over $\mathcal{S}$ and let $j \in \mathcal{I}$. Then*

(1) $proj_j{}^{[2]}((\delta_{\{j\}})_a) \subseteq \delta_{j,a}$, for all $a \in \Sigma$, and

(2) $\upsilon_j(\mathbf{C}^\infty_{SUB_{\{j\}}}) \subseteq \mathbf{C}^\infty_{\mathcal{A}_j}$.

Proof. (1) Let $a \in \Sigma$ and let $(p, p') \in (\delta_{\{j\}})_a$. From Definition 4.1.6 then follows that $(p, p') \in \Delta_a(\{\mathcal{A}_j\}) = \{(q, q') \in \prod Q_j \times \prod Q_j \mid \mathrm{proj}_j{}^{[2]}(q, q') \in \delta_{j,a}\}$. Consequently, $\mathrm{proj}_j{}^{[2]}(p, p') \in \delta_{j,a}$.

(2) Let $\alpha \in \mathbf{C}^\infty_{SUB_{\{j\}}}$. First consider the finitary case, i.e. let $\alpha \in \mathbf{C}_{SUB_{\{j\}}}$. If $\alpha \in I_j$, then $\alpha = \prod_{\{j\}} q$ for some $q \in I_j$. Hence $\mathrm{proj}_j(\alpha) = q \in I_j$ and $\upsilon_j(\alpha) = q \in \mathbf{C}_{\mathcal{A}_j}$.

If $\alpha = \beta q a q'$ for some $\beta q \in \mathbf{C}_{SUB_{\{j\}}}$, $q, q' \in \prod_{\{j\}} Q_j$, and $a \in \Sigma_{\{j\}}$, with $(q, q') \in (\delta_{\{j\}})_a$, then we proceed with an inductive argument and assume that $\upsilon_j(\beta q) \in \mathbf{C}_{\mathcal{A}_j}$. From (1) follows that $\mathrm{proj}_j{}^{[2]}(q, q') \in \delta_{j,a}$ and we thus conclude $\upsilon_j(\alpha) = \upsilon_j(\beta)\mathrm{proj}_j(q)a\mathrm{proj}_j(q') \in \mathbf{C}_{\mathcal{A}_j}$.

Consequently consider the infinitary case, i.e. let $\alpha \in \mathbf{C}^{\omega}_{SUB_{\{j\}}}$. Let $\alpha_1 \leq \alpha_2 \leq \cdots \in \mathbf{C}_{SUB_{\{j\}}}$ be such that $\alpha = \lim_{n\to\infty} \alpha_n$. By the same reasoning as above $\upsilon_j(\alpha_n) \in \mathbf{C}_{\mathcal{A}_j}$, for all $n \geq 1$. Since υ_j is a letter-to-letter homomorphism we have $\upsilon_j(\alpha_1) \leq \upsilon_j(\alpha_2) \leq \cdots$ and $\lim_{n\to\infty} \upsilon_j(\alpha_n)$ is an infinite word. Furthermore $\lim_{n\to\infty} \upsilon_j(\alpha_n) = \upsilon_j(\lim_{n\to\infty} \alpha_n)$.

Hence $\upsilon_j(\alpha) = \upsilon_j(\lim_{n\to\infty} \alpha_n) = \lim_{n\to\infty} \upsilon_j(\alpha_n) \in \mathbf{C}^{\omega}_{\mathcal{A}_j}$. □

Given a synchronized automaton $\mathcal{T} = (Q, \Sigma, \delta, I)$ over $\mathcal{S}$, the following example shows that it can be the case that there exists a $j \in \mathcal{I}$ for which $\mathrm{proj}_j{}^{[2]}((\delta_{\{j\}})_a) \subset \delta_{j,a}$, for all $a \in \Sigma$, and $\upsilon_j(\mathbf{C}^{\infty}_{SUB_{\{j\}}}) \subset \mathbf{C}^{\infty}_{\mathcal{A}_j}$.

Example 4.1.10. Let $\mathcal{A}_1 = (\{q_1, q_1'\}, \{a\}, \{(q_1, a, q_1'), (q_1', a, q_1')\}, \{q_1\})$ and $\mathcal{A}_2 = (\{q_2, q_2'\}, \{a\}, \{(q_2, a, q_2')\}, \{q_2\})$ be the automata depicted in Figure 4.5(a).

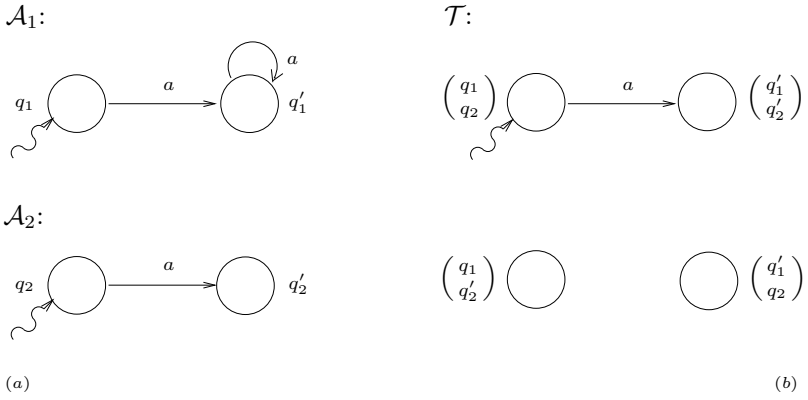

Fig. 4.5. Automata $\mathcal{A}_1$ and $\mathcal{A}_2$, and synchronized automaton $\mathcal{T}$.

Consider the synchronized automaton $\mathcal{T} = (Q, \{a\}, \{((q_1, q_2), a, (q_1', q_2'))\}, \{(q_1, q_2)\})$, in which $Q = \{(q_1, q_2), (q_1, q_2'), (q_1', q_2), (q_1', q_2')\}$, over $\{\mathcal{A}_1, \mathcal{A}_2\}$. It is depicted in Figure 4.5(b).

Let $j = 1$. It is clear that $(\delta_{\{1\}})_a = \{((q_1), (q_1'))\}$. Thus $\mathrm{proj}_1{}^{[2]}((\delta_{\{1\}})_a) = \{(q_1, q_1')\} \subset \{(q_1, q_1'), (q_1', q_1')\} = \delta_{1,a}$. Clearly, $\mathbf{C}_{SUB_{\{1\}}} = \{(q_1), (q_1)a(q_1')\}$. Hence $\upsilon_1(\mathbf{C}_{SUB_{\{1\}}}) = \{q_1, q_1 a q_1'\} \subset \{q_1, q_1 a q_1', q_1 a q_1' a q_1', \dots\} \cup \{q_1 (a q_1')^{\omega}\} = \mathbf{C}^{\infty}_{\mathcal{A}_1}$. □

4.2 Projecting

In this section we want to extract the computations of any one of the (sub)automata constituting a synchronized automaton from the computations of this synchronized automaton. Note, however, that within the formalization of a synchronized automaton, no explicit information on loops is provided. That is to say, in general one cannot distinguish whether or not an automaton with a loop on a in its current local state participates in the synchronized automaton's synchronization on a. This automaton may have been idle or, after having participated in the action a starting from the global state, it may have returned to its original local state.

Example 4.2.1. Consider the three automata $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$, as depicted in Figure 4.6(a).

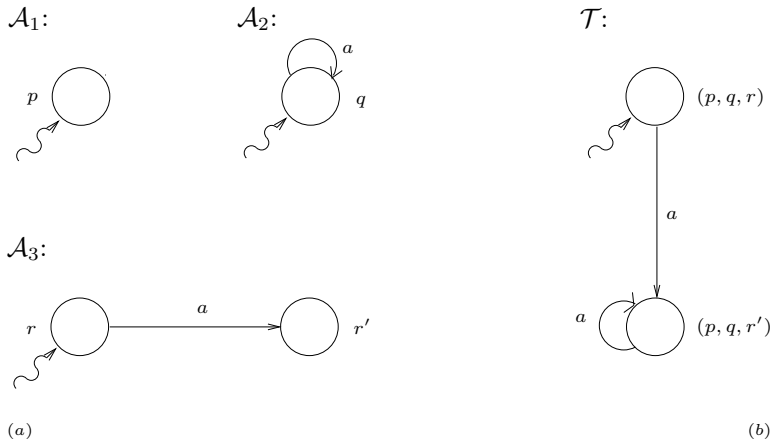

Fig. 4.6. Automata $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$, and synchronized automaton $\mathcal{T}$.

$\mathcal{A}_1$ and $\mathcal{A}_2$ each have only one state, p and q, respectively, which are their initial states. $\mathcal{A}_3$ has two states, r and r', of which r is its initial state. $\mathcal{A}_1$ has an empty alphabet, while both $\mathcal{A}_2$ and $\mathcal{A}_3$ have $\{a\}$ as their alphabet. Finally, $\mathcal{A}_1$ has no transitions at all, the transition relation of $\mathcal{A}_2$ consists solely of the loop (q, a, q), and that of $\mathcal{A}_3$ is $\{(r, a, r')\}$.

Now consider the synchronized automaton $\mathcal{T} = (\{(p, q, r), (p, q, r')\}, \{a\}, \delta, \{(p, q, r)\})$, where $\delta_a = \Delta_a(\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}) \setminus \{((p, q, r), a, (p, q, r))\}$, over $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$. It is depicted in Figure 4.6(b). Now one might wonder which automata participate when the a-transitions of $\mathcal{T}$ are executed.

First consider the execution of the loop on a at (p, q, r') in $\mathcal{T}$. Clearly $\mathcal{A}_1$ does not participate as it cannot execute a at all. Also $\mathcal{A}_3$ does not participate since a is not enabled in r'. However, since in every transition of a synchronized automaton at least one component is required to participate, it must thus be the case that $\mathcal{A}_2$ executes its loop on a.

Secondly, consider the execution of the a-transition from (p, q, r) to (p, q, r') in $\mathcal{T}$. Clearly $\mathcal{A}_1$ is not involved. On the other hand, $\mathcal{A}_3$ is responsible for the local state change from r to r' and thus participates by executing a. But what about $\mathcal{A}_2$ — does it execute its loop on a or does it remain idle during this execution of a by $\mathcal{T}$? □

In spite of the fact that Example 4.2.1 shows that information on the actual execution of loops by the constituting automata is lacking in the definition of a synchronized automaton, in order to relate the computations of a synchronized automaton to those taking place in its constituting automata we simply apply projections.

Recall that computations of a synchronized automaton are determined by the consecutive execution of transitions, starting from the initial state. Consider a transition (q, a, q') of a synchronized automaton over $\mathcal{S}$. We now assume that the j-th automaton participates in this transition by executing $(\mathrm{proj}_j(q), a, \mathrm{proj}_j(q'))$ whenever $\mathrm{proj}^{[2]}(q, q') \in \delta_{j,a}$. Otherwise no transition takes place in the j-th automaton. We thus resolve the lacking of information on loops by assuming that the presence of an automaton's loop in a transition of a synchronized automaton implies execution of that loop. This may be considered as a "maximal" interpretation of the participation of its constituting automata in transitions of synchronized automata, in the sense that we assume that if an automaton could have participated in an a-transition of the synchronized automaton by executing a loop on this action a, then it indeed has done so.

Example 4.2.2. (Example 4.2.1 continued) We consider the abovementioned maximal interpretation of the automata's participation in transitions of the synchronized automaton. Then $\mathcal{A}_2$ is thus assumed to execute its loop on a at q during the execution of a at (p, q, r) by means of the a-transition $((p, q, r), (p, q, r'))$ of $\mathcal{T}$. □

Using the maximal interpretation we define the projection on (sub)automata of the computations of a synchronized automaton. Because of the results at the end of Section 4.1 we define separately the projection on the subautomaton defined by $\{j\}$ of a synchronized automaton and the projection on its j-th automaton. The formal reason behind this is the fact that Q_j and $\prod_{\{j\}} Q_j$

are not identified. In fact, as we will show shortly, the two separate definitions are the same whenever Q_j and $\prod_{\{j\}} Q_j$ are identified.

Finally, one could think of other interpretations of the participation of constituting (sub)automata in transitions of synchronized automata in case of loops.

Definition 4.2.3. *Let $\mathcal{T} = (Q, \Sigma, \delta, I)$ be a synchronized automaton over $\mathcal{S}$. Let $J \subseteq \mathcal{I}$. Then*

(1) the projection on subautomaton SUB_J of a finite computation $\alpha \in \mathbf{C}_\mathcal{T}$ *is denoted by $\pi_{SUB_J}(\alpha)$ and is defined as*
(a) if $\alpha = q \in I$, then $\pi_{SUB_J}(\alpha) = proj_J(q)$, and
(b) if $\alpha = \beta q a q'$, for some $\beta q \in \mathbf{C}_\mathcal{T}$, $q, q' \in Q$, and $a \in \Sigma$, then

$$\pi_{SUB_J}(\alpha) = \begin{cases} \pi_{SUB_J}(\beta q) & \text{if } proj_J{}^{[2]}(q, q') \notin (\delta_J)_a \text{ and} \\ \pi_{SUB_J}(\beta q) a proj_J(q') & \text{if } proj_J{}^{[2]}(q, q') \in (\delta_J)_a, \end{cases}$$

and

(2) the projection on subautomaton SUB_J of an infinite computation $\alpha \in \mathbf{C}^\omega_\mathcal{T}$ *is denoted by $\pi_{SUB_J}(\alpha)$ and is defined as*

$$\pi_{SUB_J}(\alpha) = \lim_{n\to\infty} \pi_{SUB_J}(\alpha_n) \text{ whenever } \alpha = \lim_{n\to\infty} \alpha_n \text{ for } \alpha_1 \leq \alpha_2 \leq \cdots \in \mathbf{C}_\mathcal{T}.$$

Let $j \in \mathcal{I}$. Then

(3) the projection on automaton $\mathcal{A}_j$ of a finite computation $\alpha \in \mathbf{C}_\mathcal{T}$ *is denoted by $\pi_{\mathcal{A}_j}(\alpha)$ and is defined as*
(a) if $\alpha = q \in I$, then $\pi_{\mathcal{A}_j}(\alpha) = proj_j(q)$, and
(b) if $\alpha = \beta q a q'$, for some $\beta q \in \mathbf{C}_\mathcal{T}$, $q, q' \in Q$, and $a \in \Sigma$, then

$$\pi_{\mathcal{A}_j}(\alpha) = \begin{cases} \pi_{\mathcal{A}_j}(\beta q) & \text{if } proj_j{}^{[2]}(q, q') \notin \delta_{j,a} \text{ and} \\ \pi_{\mathcal{A}_j}(\beta q) a proj_j(q') & \text{if } proj_j{}^{[2]}(q, q') \in \delta_{j,a}, \end{cases}$$

and

(4) the projection on automaton $\mathcal{A}_j$ of an infinite computation $\alpha \in \mathbf{C}^\omega_\mathcal{T}$ *is denoted by $\pi_{\mathcal{A}_j}(\alpha)$ and is defined as*

$$\pi_{\mathcal{A}_j}(\alpha) = \lim_{n\to\infty} \pi_{\mathcal{A}_j}(\alpha_n) \text{ whenever } \alpha = \lim_{n\to\infty} \alpha_n \text{ for } \alpha_1 \leq \alpha_2 \leq \cdots \in \mathbf{C}_\mathcal{T}. \square$$

Recall that every prefix of odd length of an infinite computation α of a synchronized automaton $\mathcal{T}$ is a finite computation. Thus α is the limit of any prefix-ordered infinite subset of its finite prefixes. Moreover, if $\alpha_1 \leq \alpha_2$ for finite computations α_1 and α_2 of $\mathcal{T}$, then $\pi_{SUB_J}(\alpha_1) \leq \pi_{SUB_J}(\alpha_2)$, for all

$J \subseteq \mathcal{I}$, and $\pi_{\mathcal{A}_j}(\alpha_1) \leq \pi_{\mathcal{A}_j}(\alpha_2)$, for all $j \in \mathcal{I}$. Hence the projection $\pi_{SUB_J}(\alpha)$ on subautomaton $SUB_J(\mathcal{T})$ and the projection $\pi_{\mathcal{A}_j}(\alpha)$ on automaton $\mathcal{A}_j$ are well defined for any computation $\alpha \in \mathbf{C}_{\mathcal{T}}^{\infty}$. Furthermore, $\pi_{SUB_J}(\lim_{n\to\infty} \alpha_n) = \lim_{n\to\infty} \pi_{SUB_J}(\alpha_n)$ and $\pi_{\mathcal{A}_j}(\lim_{n\to\infty} \alpha_n) = \lim_{n\to\infty} \pi_{\mathcal{A}_j}(\alpha_n)$.

Note that $\pi_{SUB_J}(\alpha)$ and $\pi_{\mathcal{A}_j}(\alpha)$ can be finite sequences. This happens if subautomaton $SUB_J(\mathcal{T})$ or automaton $\mathcal{A}_j$, respectively, no longer participates in α after a finite number k of steps. In that case, if $\alpha = q_0a_1q_1a_2q_2\cdots$, then $\pi_{SUB_J}(q_0a_1q_1a_2q_2\cdots a_nq_n) = \pi_{SUB_J}(q_0a_1q_1a_2q_2\cdots a_nq_na_{n+1}q_{n+1})$, for all $n \geq k$, and hence $\pi_{SUB_J}(\alpha) = \pi_{SUB_J}(q_0a_1q_1a_2q_2\cdots a_kq_k)$. Likewise $\pi_{\mathcal{A}_j}(\alpha) = \pi_{\mathcal{A}_j}(q_0a_1q_1a_2q_2\cdots a_kq_k)$ in that case.

Contrary to what one might expect from Example 4.1.10, we indeed see that for each computation of a synchronized automaton its projection on an automaton "agrees" with its projection on the corresponding singleton subautomaton, in the sense that they are equal whenever Q_j and $\prod_{\{j\}} Q_j$ are identified.

Theorem 4.2.4. *Let $\mathcal{T} = (Q, \Sigma, \delta, I)$ be a synchronized automaton over $\mathcal{S}$ and let $j \in \mathcal{I}$. Then*

$$\upsilon_j(\pi_{SUB_{\{j\}}}(\mathbf{C}_{\mathcal{T}}^{\infty})) = \pi_{\mathcal{A}_j}(\mathbf{C}_{\mathcal{T}}^{\infty}).$$

Proof. Let $\alpha \in \mathbf{C}_{\mathcal{T}}^{\infty}$. First consider the finitary case, i.e. let $\alpha \in \mathbf{C}_{\mathcal{T}}$. We proceed by induction on the length of w. If $\alpha = q$, then $\alpha \in \prod_{i\in\mathcal{I}} I_i$. By Definition 4.2.3, $\pi_{\mathcal{A}_j}(\alpha) = \mathrm{proj}_j(\alpha)$ and $\pi_{SUB_{\{j\}}}(\alpha) = \mathrm{proj}_{\{j\}}(\alpha)$. Consequently $\upsilon_j(\pi_{SUB_{\{j\}}}(\alpha)) = \mathrm{proj}_j(\mathrm{proj}_{\{j\}}(\alpha)) = \mathrm{proj}_j(\alpha) = \pi_{\mathcal{A}_j}(\alpha)$.

Next assume that $\alpha = \beta qaq'$ for some $\beta \in (\Sigma \cup Q)^*$, $q, q' \in Q$, and $a \in \Sigma$, such that $\beta q \in \mathbf{C}_{\mathcal{T}}$ and $(q, q') \in \delta_a$. It is not difficult to see that $\mathrm{proj}_j{}^{[2]}(q, q') \in \delta_{j,a}$ if and only if $\mathrm{proj}_{\{j\}}{}^{[2]}(q, q') \in (\delta_{\{j\}})_a$. Indeed we already know from Lemma 4.1.9 that $\mathrm{proj}_{\{j\}}{}^{[2]}((\delta_{\{j\}})_a) \subseteq \delta_{j,a}$ and hence $\mathrm{proj}_{\{j\}}{}^{[2]}(q, q') \in (\delta_{\{j\}})_a$ implies $\mathrm{proj}_j{}^{[2]}(\mathrm{proj}_{\{j\}}{}^{[2]}(q, q')) = \mathrm{proj}_j{}^{[2]}(q, q') \in \delta_{j,a}$. Conversely, if $\mathrm{proj}_j{}^{[2]}(q, q') \in \delta_{j,a}$ then $\mathrm{proj}_{\{j\}}{}^{[2]}(q, q') \in (\delta_{\{j\}})_a$ provided that $(q, q') \in \delta_a$, which is the case. Returning to our computation α we now obtain the following.

If $\mathrm{proj}_j{}^{[2]}(q, q') \notin \delta_{j,a}$, then by induction $\pi_{\mathcal{A}_j}(\alpha) = \pi_{\mathcal{A}_j}(\beta q)$ and $\pi_{\mathcal{A}_j}(\beta q) = \upsilon_j(\pi_{SUB_{\{j\}}}(\beta q))$. As $\mathrm{proj}_{\{j\}}{}^{[2]}(q, q') \notin (\delta_{\{j\}})_a$ it follows that $\pi_{SUB_{\{j\}}}(\alpha) = \pi_{SUB_{\{j\}}}(\beta q)$. Consequently $\pi_{\mathcal{A}_j}(\alpha) = \upsilon_j(\pi_{SUB_{\{j\}}}(\alpha))$.

If $\mathrm{proj}_j{}^{[2]}(q, q') \in \delta_{j,a}$, then by induction $\pi_{\mathcal{A}_j}(\alpha) = \pi_{\mathcal{A}_j}(\beta q)a\mathrm{proj}_j(q') = \upsilon_j(\pi_{SUB_{\{j\}}}(\beta q))a\mathrm{proj}_j(q')$. As $\mathrm{proj}_{\{j\}}{}^{[2]}(q, q') \in (\delta_{\{j\}})_a$, then $\pi_{SUB_{\{j\}}}(\alpha) = \pi_{SUB_{\{j\}}}(\beta q)a\mathrm{proj}_{\{j\}}(q')$. Hence $\pi_{\mathcal{A}_j}(\alpha) = \upsilon_j(\pi_{SUB_{\{j\}}}(\beta q)a\mathrm{proj}_{\{j\}}(q')) = \upsilon_j(\pi_{SUB_{\{j\}}}(\alpha))$. This concludes the proof for the finitary case.

Now consider the infinitary case, i.e. let $\alpha \in \mathbf{C}_{\mathcal{T}}^{\omega}$. Let $\alpha_1 \leq \alpha_2 \leq \cdots \in \mathbf{C}_{\mathcal{T}}$ be such that $\alpha = \lim_{n\to\infty} \alpha_n$. Then by definition $\pi_{\mathcal{A}_j}(\alpha) = \lim_{n\to\infty} \pi_{\mathcal{A}_j}(\alpha_n)$ and $\pi_{SUB_{\{j\}}}(\alpha) = \lim_{n\to\infty} \pi_{SUB_{\{j\}}}(\alpha_n)$. By the same reasoning as above $\pi_{\mathcal{A}_j}(\alpha_n) = \upsilon_j(\pi_{SUB_{\{j\}}}(\alpha_n))$ and since υ_j is a homomorphism we thus obtain $\pi_{\mathcal{A}_j}(\alpha) = \lim_{n\to\infty} \upsilon_j(\pi_{SUB_{\{j\}}}(\alpha_n)) = \upsilon_j(\lim_{n\to\infty} \pi_{SUB_{\{j\}}}(\alpha_n)) = \upsilon_j(\pi_{SUB_{\{j\}}}(\alpha))$. □

Example 4.2.5. (Example 4.1.10 continued) It is easy to see that $\mathbf{C}_{\mathcal{T}} = \{(q_1, q_2), (q_1, q_2)a(q_1', q_2')\}$. Now recall that $j = 1$. Then $\upsilon_1(\pi_{SUB_{\{1\}}}(\mathbf{C}_{\mathcal{T}})) = \upsilon_1(\{(q_1), (q_1)a(q_1')\}) = \{q_1, q_1aq_1'\} = \pi_{\mathcal{A}_1}(\mathbf{C}_{\mathcal{T}})$. □

We conclude this section by showing that if we take the set of computations of a synchronized automaton and consequently project on a (sub)automaton of that synchronized automaton, then the result is always included in the set of computations of that (sub)automaton. However, these inclusions may be proper.

Lemma 4.2.6. *Let $\mathcal{T} = (Q, \Sigma, \delta, I)$ be a synchronized automaton over $\mathcal{S}$ and let $J \subseteq \mathcal{I}$. Then*

$$\pi_{SUB_J}(\mathbf{C}_{\mathcal{T}}^{\infty}) \subseteq \mathbf{C}_{SUB_J}^{\infty}.$$

Proof. Let $\alpha \in \mathbf{C}_{\mathcal{T}}^{\infty}$. First consider the finitary case, i.e. let $\alpha \in \mathbf{C}_{\mathcal{T}}$. Hence $\alpha = q_0a_1q_1a_2\cdots a_nq_n$ for some $n \geq 0$, $q_\ell \in Q$ for $0 \leq \ell \leq n$, and $a_\ell \in \Sigma$ for $1 \leq \ell \leq n$. By Definition 4.2.3 we have $\pi_{SUB_J}(\alpha) = p_0b_1p_1b_2\cdots b_mp_m$ for some $m \geq 0$, $p_\ell \in Q_J$ for $0 \leq \ell \leq m$, and $b_\ell \in \Sigma_J$ for $1 \leq \ell \leq m$.
We prove by induction on n that $\pi_{SUB_J}(\alpha) \in \mathbf{C}_{SUB_J}$ and, furthermore, that $\mathrm{proj}_J(q_n) = p_m$.

If $n = 0$, then $\alpha = q_0 \in I$. Thus by Definition 4.2.3 we have $\pi_{SUB_J}(\alpha) = \mathrm{proj}_J(q_0) \in I_J$, which implies that $\pi_{SUB_J}(\alpha) \in \mathbf{C}_{SUB_J}$. Moreover, $m = 0$ and $\mathrm{proj}_J(q_0) = p_0$.

Now assume that the statement holds for some $k \geq 0$. Let $n = k + 1$. Then by Definition 4.2.3 we have $\pi_{SUB_J}(\alpha) = \pi_{SUB_J}(q_0a_1q_1a_2\cdots a_kq_k)\gamma$, where $\gamma = \lambda$ if $\mathrm{proj}_J{}^{[2]}(q_k, q_{k+1}) \notin (\delta_J)_{a_{k+1}}$ and $\gamma = a_{k+1}\mathrm{proj}_J(q_{k+1})$ otherwise.
First consider the case $\gamma = \lambda$. Then $\pi_{SUB_J}(\alpha) \in \mathbf{C}_{SUB_J}$ by the induction hypothesis. Moreover, since $\mathrm{proj}_J{}^{[2]}(q_k, q_{k+1}) \notin (\delta_J)_{a_{k+1}}$, Definition 4.1.1 implies that $\mathrm{proj}_J(q_k) = \mathrm{proj}_J(q_{k+1})$. By the induction hypothesis $\mathrm{proj}_J(q_k) = p_m$, and hence $\mathrm{proj}_J(q_{k+1}) = p_m$.
Secondly, consider the case $\gamma \neq \lambda$. Then $\pi_{SUB_J}(\alpha) = p_0b_1p_1b_2\cdots b_mp_m = \pi_{SUB_J}(q_0a_1q_1a_2\cdots a_kq_k)a_{k+1}\mathrm{proj}_J(q_{k+1})$. Thus in this case $b_m = a_{k+1}$ and $p_m = \mathrm{proj}_J(q_{k+1})$.

The only thing left to prove is that $\pi_{SUB_J}(\alpha) \in \mathbf{C}_{SUB_J}$. We already have that $\text{proj}_J{}^{[2]}(q_k, q_{k+1}) \in (\delta_J)_{a_{k+1}}$. From the induction hypothesis above it now follows that $p_0 b_1 p_1 b_2 \cdots b_{m-1} p_{m-1} \in \mathbf{C}_{SUB_J}$ and $p_{m-1} = \text{proj}_J(q_k)$. Thus $\text{proj}_J{}^{[2]}(p_{m-1}, p_m) = \text{proj}_J{}^{[2]}(q_k, q_{k+1}) \in (\delta_J)_{b_m}$, which implies $\pi_{SUB_J}(\alpha) = p_0 b_1 p_1 b_2 \cdots b_m p_m \in \mathbf{C}_{SUB_J}$.

Now consider the infinitary case, i.e. let $\alpha \in \mathbf{C}^{\omega}_{\mathcal{T}}$. Hence $\alpha = \lim_{n\to\infty} \alpha_n$ for finite computations $\alpha_1 \leq \alpha_2 \leq \cdots \in \mathbf{C}_{\mathcal{T}}$. Then $\pi_{SUB_J}(\alpha_1) \leq \pi_{SUB_J}(\alpha_2) \leq \cdots$ and $\pi_{SUB_J}(\alpha_n) \in \mathbf{C}_{SUB_J}$, for all $n \geq 1$. Thus $\pi_{SUB_J}(\alpha) = \lim_{n\to\infty} \pi_{SUB_J}(\alpha_n) \in \mathbf{C}^{\infty}_{SUB_J}$. □

Corollary 4.2.7. *Let $\mathcal{T}$ be a synchronized automaton over $\mathcal{S}$ and let $j \in \mathcal{I}$. Then*

$$\pi_{\mathcal{A}_j}(\mathbf{C}^{\infty}_{\mathcal{T}}) \subseteq \mathbf{C}^{\infty}_{\mathcal{A}_j}.$$

Proof. Directly from Theorem 4.2.4 and Lemmata 4.2.6 and 4.1.9. □

In the following example we show that, given a synchronized automaton $\mathcal{T}$ over $\mathcal{S}$, it can be the case that there exists a subset $J \subseteq \mathcal{I}$ or a $j \in \mathcal{I}$ for which $\pi_{SUB_J}(\mathbf{C}^{\infty}_{\mathcal{T}}) \subset \mathbf{C}^{\infty}_{SUB_J}$ or $\pi_{\mathcal{A}_j}(\mathbf{C}^{\infty}_{\mathcal{T}}) \subset \mathbf{C}^{\infty}_{\mathcal{A}_j}$, respectively.

Example 4.2.8. Let $\mathcal{A}_1 = (\{q_1, q_1'\}, \{a, b\}, \{(q_1, a, q_1), (q_1, b, q_1')\}, \{q_1\})$ and $\mathcal{A}_2 = (\{q_2, q_2'\}, \{a\}, \{(q_2, a, q_2')\}, \{q_2\})$ be the automata depicted in Figure 4.7(a).

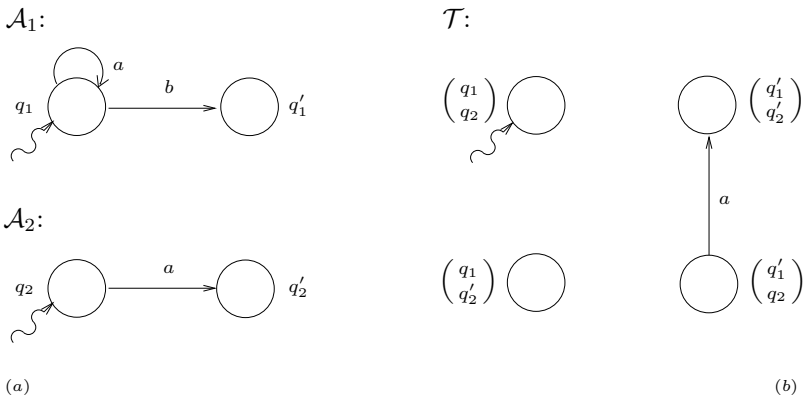

Fig. 4.7. Automata $\mathcal{A}_1$ and $\mathcal{A}_2$, and synchronized automaton $\mathcal{T}$.

Consider synchronized automaton $\mathcal{T} = (Q, \{a, b\}, \{((q_1', q_2), a, (q_1', q_2'))\}, \{(q_1, q_2)\})$, in which $Q = \{(q_1, q_2), (q_1, q_2'), (q_1', q_2), (q_1', q_2')\}$, over $\{\mathcal{A}_1, \mathcal{A}_2\}$. It is depicted in Figure 4.7(b).

It is clear that (q_1, q_2) is the only computation of $\mathcal{T}$, whereas $SUB_{\{2\}}$ has the two computations (q_2) and $(q_2)a(q_2')$. Hence we have $\pi_{SUB_{\{2\}}}(\mathbf{C}^\infty_{\mathcal{T}}) = \mathrm{proj}_{\{2\}}((q_1, q_2)) = (q_2) \subset \{(q_2), (q_2)a(q_2')\} = \mathbf{C}^\infty_{SUB_{\{2\}}}$ and, according to Lemma 4.1.9(2) and Theorem 4.2.4, $\pi_{\mathcal{A}_2}(\mathbf{C}^\infty_{\mathcal{T}}) = \upsilon_2(\pi_{SUB_{\{2\}}}(\mathbf{C}^\infty_{\mathcal{T}})) = \upsilon_2((q_2)) = q_2 \subset \{q_2, q_2aq_2'\} = \upsilon_2(\{(q_2), (q_2)a(q_2')\}) = \upsilon_2(\mathbf{C}^\infty_{SUB_{\{2\}}}) \subseteq \mathbf{C}^\infty_{\mathcal{A}_2}$.

As a further example we consider the synchronized automaton $\mathcal{T}' = (Q, \{a, b\}, \{((q_1, q_2), a, (q_1, q_2'))\}, \{(q_1, q_2)\})$ over $\{\mathcal{A}_1, \mathcal{A}_2\}$. It is depicted in Figure 4.8.

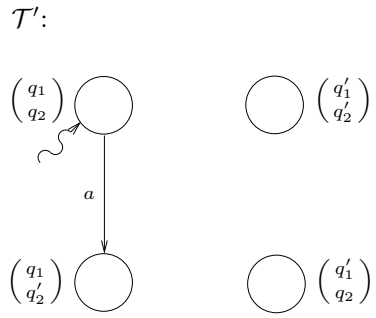

Fig. 4.8. Synchronized automaton $\mathcal{T}'$.

It is clear that $\mathbf{C}^\infty_{\mathcal{T}'} = \{(q_1, q_2), (q_1, q_2)a(q_1, q_2')\}$, whereas we have $\mathbf{C}^\infty_{\mathcal{A}_1} = \{q_1, q_1aq_1, q_1bq_1', q_1aq_1aq_1, q_1aq_1bq_1', \dots\} \cup \{q_1(aq_1)^\omega\}$. Hence we now see that $\pi_{\mathcal{A}_1}(\mathbf{C}^\infty_{\mathcal{T}'}) = \{q_1, q_1aq_1\} \subset \mathbf{C}^\infty_{\mathcal{A}_1}$. □

4.3 Iterated Composition

In this section we show that synchronized automata are naturally suited to describe hierarchical systems. We do this by demonstrating how to iteratively build synchronized automata from synchronized automata, and how to consider subautomata as constituting automata in an iterated definition of a synchronized automaton.

Given a set of automata $\mathcal{S}$, there may be several ways of forming a synchronized automaton over $\mathcal{S}$. Until now we directly defined synchronized automata over $\mathcal{S}$, but other routes are also feasible. We might first (iteratively) form synchronized automata from (disjoint) subsets of $\mathcal{S}$ and then use these as automata for a higher-level synchronized automaton, until after a finite number of such iterations all automata from $\mathcal{S}$ have been used. This is shown in Example 4.1.5 and Figure 4.2, where four wheels are combined by first

connecting two of them (to form an axle) and then attaching the other two to the result. This section shows that whatever route chosen, the resulting *iterated* synchronized automaton can always be regarded as a synchronized automaton over $\mathcal{S}$: it will always have the same alphabet of actions and it will have essentially the same state space, transition space, and set of initial states as any synchronized automaton formed directly over $\mathcal{S}$.

Example 4.3.1. Let $\mathcal{S} = \{A_i \mid i \in [7]\}$, with $\mathcal{A}_i = (Q_i, \Sigma_i, \delta_i, I_i)$, for $i \in [7]$. Let $\mathcal{T}_{1-7} = (\prod_{i\in[7]} Q_i, \bigcup_{i\in[7]} \Sigma_i, \delta, \prod_{i\in[7]} I_i)$ be a synchronized automaton over $\mathcal{S}$. As δ is not relevant for the moment, it is not specified any further. Recall that all other parameters of $\mathcal{T}_{1-7}$ are uniquely defined by Definition 4.1.2. The structure of this synchronized automaton relative to $\mathcal{S}$, is depicted in the tree of Figure 4.9(a).

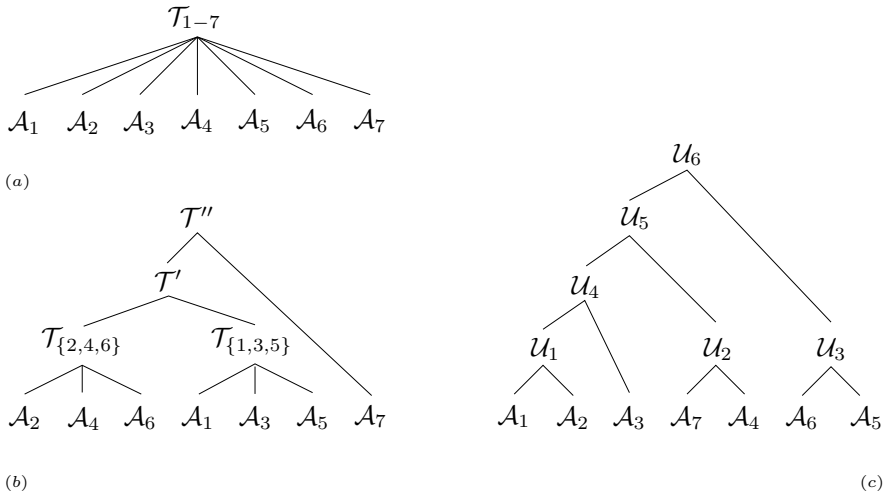

Fig. 4.9. Three synchronized automata constructed from $\{A_i \mid i \in [7]\}$.

Next consider the synchronized automaton $\mathcal{T}_{\{2,4,6\}}$ over $\{\mathcal{A}_2, \mathcal{A}_4, \mathcal{A}_6\}$ and the synchronized automaton $\mathcal{T}_{\{1,3,5\}}$ over $\{\mathcal{A}_1, \mathcal{A}_3, \mathcal{A}_5\}$. Let $\mathcal{T}_{\{2,4,6\}}$ be specified as $\mathcal{T}_{\{2,4,6\}} = (P_1, \Gamma_1, \gamma_1, J_1)$ and let $\mathcal{T}_{\{1,3,5\}}$ be specified as $\mathcal{T}_{\{1,3,5\}} = (P_2, \Gamma_2, \gamma_2, J_2)$.

Let $\mathcal{T}'$ be a synchronized automaton over $\mathcal{S}' = \{\mathcal{A}'_1, \mathcal{A}'_2\}$, with $\mathcal{A}'_1 = \mathcal{T}_{\{2,4,6\}}$ and $\mathcal{A}'_2 = \mathcal{T}_{\{1,3,5\}}$. Let $\mathcal{T}'$ be specified as $\mathcal{T}' = (P', \Gamma', \gamma', J')$.

Let $\mathcal{T}''$ be a synchronized automaton over $\mathcal{S}'' = \{\mathcal{A}''_1, \mathcal{A}''_2\}$, with $\mathcal{A}''_1 = \mathcal{T}'$ and $\mathcal{A}''_2 = \mathcal{A}_7$. Let $\mathcal{T}''$ be specified as $\mathcal{T}'' = (P'', \Gamma'', \gamma'', J'')$, for some $\gamma'' \subseteq P'' \times \Gamma'' \times P''$. By Definition 4.1.2 we have $P'' = P' \times Q_7 = (\prod_{i\in\{1,2\}} P_i) \times$

$Q_7 = ((\prod_{i\in\{2,4,6\}} Q_i) \times (\prod_{i\in\{1,3,5\}} Q_i)) \times Q_7 = ((Q_2 \times Q_4 \times Q_6) \times (Q_1 \times Q_3 \times Q_5)) \times Q_7$. Similarly, $J'' = ((I_2 \times I_4 \times I_6) \times (I_1 \times I_3 \times I_5)) \times I_7$. Furthermore, $\Gamma'' = \Gamma' \cup \Sigma_7 = (\bigcup_{i\in\{1,2\}} \Gamma_i) \cup \Sigma_7 = ((\bigcup_{i\in\{2,4,6\}} \Sigma_i) \cup (\bigcup_{i\in\{1,3,5\}} \Sigma_i)) \cup \Sigma_7 = \bigcup_{i\in[7]} \Sigma_i$.

Thus $\mathcal{T}''$ has the same actions as any synchronized automaton formed directly over $\mathcal{S}$. Its set of states, however, differs from the set of states of a synchronized automaton over $\mathcal{S}$ by its nested structure and its ordering. In Figure 4.9(b) the structure of $\mathcal{T}''$ relative to $\mathcal{S}$ is depicted.

In Figure 4.9(c) the structure relative to $\mathcal{S}$ of yet another route for constructing a synchronized automaton, starting from the automata in $\mathcal{S}$, is depicted. The set of states of this particular synchronized automaton $\mathcal{U}_6$ is $(((Q_1 \times Q_2) \times Q_3) \times (Q_7 \times Q_4)) \times (Q_6 \times Q_5)$. □

In order to describe in a precise way the relationship between a synchronized automaton obtained by iteratively composing synchronized automata and a synchronized automaton formed directly from a given set of automata, we need formal notions enabling us to describe the construction and the parsing of vectors with vectors as elements. Let $\mathcal{D} = \{D_j \mid j \in J\}$ be an indexed set, with $J \subseteq \mathbb{N}$ and $J \neq \varnothing$. Then $\mathcal{V}(\mathcal{D})$ is defined as consisting of all finitely nested combinations of elements from $\mathcal{D}$ provided each D_j is used at most once. The *domain* of an element V from $\mathcal{V}(\mathcal{D})$ consequently is defined to consist of the indices of the sets in $\mathcal{D}$ combined to form V. This leads to the following recursive definition of $\mathcal{V}(\mathcal{D})$ and the accompanying notion of domain.

Definition 4.3.2. *$\mathcal{V}(\mathcal{D})$ is the smallest set $\mathcal{V}$ such that*

(1) $D_j \in \mathcal{V}$, for each $j \in J$;
Set $dom(D_j) = \{j\}$, and
(2) if $\{V_\ell \mid \ell \in L\} \subseteq \mathcal{V}$, with $L \subseteq \mathbb{N}$ and $L \neq \varnothing$, then $\prod_{\ell\in L} V_\ell \in \mathcal{V}$ provided that for all $k \neq \ell \in L$, $dom(V_k) \cap dom(V_\ell) = \varnothing$;
Set $dom(\prod_{\ell\in L} V_\ell) = \bigcup_{\ell\in L} dom(V_\ell)$. □

This definition provides a description of how to construct products of products of indexed sets. Every element of $\mathcal{V}(\mathcal{D})$ describes a finitely nested cartesian product of sets from $\mathcal{D}$, while its domain gives the information as to which D_j have been used.

Note that according to step (2) of Definition 4.3.2 each product may combine an infinite number of sets. In the construction of any product in $\mathcal{V}$, however, step (2) is applied only a finite number of times. This corresponds to the intuition that a synchronized automaton is constructed by a finite iteration.

Example 4.3.3. (Example 4.3.1 continued) Let $\mathcal{Q} = \{Q_i \mid i \in [7]\}$. The set of states $P_2 = \prod_{i\in\{1,3,5\}} Q_i$ is an element of $\mathcal{V}(\mathcal{Q})$ with domain $\{1,3,5\}$. Also $P' = P_1 \times P_2 = \prod_{i\in\{2,4,6\}} Q_i \times \prod_{i\in\{1,3,5\}} Q_i$ is an element of $\mathcal{V}(\mathcal{Q})$. Its domain is $\{2,4,6\} \cup \{1,3,5\} = \{1,2,3,4,5,6\}$. Finally, for $P'' = P' \times Q_7 \in \mathcal{V}(\mathcal{Q})$, we have $\mathrm{dom}\,(P' \times Q_7) = \{1,2,3,4,5,6,7\}$. □

Given an element v of a nested cartesian product V from $\mathcal{V}(\mathcal{D})$ with domain $\mathrm{dom}\,(V)$, we want to *unpack* and *reorder* v in such a way that the "corresponding" element of $\prod_{j\in\mathrm{dom}\,(V)} D_j$ results. To this end we define the function u_V which recursively, for each $j \in \mathrm{dom}\,(V)$, locates in v the element in the position of D_j according to the construction of V. Note that since each D_j with $j \in \mathrm{dom}\,(V)$ is used exactly once in the construction of V, its position in V is unique. Thus u_V **u**npacks v and on basis of this unpacking the resulting elements of $\bigcup_{j\in\mathrm{dom}\,(V)} D_j$ are ordered in $\langle v\rangle_V$ according to $\mathrm{dom}\,(V)$.

Definition 4.3.4. *Let $V \in \mathcal{V}(\mathcal{D})$ be such that $dom\,(V) = J'$ for some $J' \subseteq J$. Then*

(1) the function $u_V : V \times J' \to \bigcup_{j\in J'} D_j$ is defined as follows:
  (a) if $J' = \{j\}$ and $V = D_j$, then $u_V(v,j) = v$ for all $v \in V$ and
  (b) if $V = \prod_{\ell\in L} V_\ell$, with $V_\ell \in \mathcal{V}(\mathcal{D})$ for all $\ell \in L$, then, for all $v \in V$ and $j \in J'$, $u_V(v,j) = u_{V_k}(proj_k(v), j)$, where $k \in L$ is such that $j \in dom\,(V_k)$, and

(2) the reordering *of an element $v \in V$ relative to the construction of V is denoted by $\langle v\rangle_V$ and is defined as*

$$\langle v\rangle_V = \prod_{j\in J'} u_V(v,j).$$ □

Example 4.3.5. (Example 4.3.3 continued) Assume that we know that $q = (((x,m,\ell),(e,a,p)),e) \in P''$. With the above definition we now reorder q relative to the construction of P'': $\langle q\rangle_{P''} = \prod_{i\in[7]} u_{P''}(q,i)$. Here, e.g., $u_{P''}(q,3) = a$. This follows from the fact that $u_{P''}((((x,m,\ell),(e,a,p)),e),3) = u_{P'}(((x,m,\ell),(e,a,p)),3)$ since $3 \in \mathrm{dom}\,(P')$, $u_{P'}(((x,m,\ell),(e,a,p)),3) = u_{P_2}((e,a,p),3)$ as $3 \in \mathrm{dom}\,(P_2)$, and $u_{P_2}((e,a,p),3) = u_{Q_3}(a,3) = a$. Each $u_{P''}(q,i)$ can thus be determined, leading to $\langle q\rangle_{P''} = (e,x,a,m,p,\ell,e)$. □

Definition 4.3.4 may seem unnecessarily complicated but, as illustrated in the next example, the information about the construction of $V \in \mathcal{V}(\mathcal{D})$ is necessary in order to obtain a faithful reordering of the entries from $\bigcup_{j\in J} D_j$ in $\mathcal{V}$.

Example 4.3.6. Let $\mathcal{Q} = \{Q_i \mid i \in [3]\}$. Let $a \in Q_1$ and let $b, c \in Q_2 \cap Q_3$. Now assume we want to reorder $q = (a, (b, c))$. Then we need to know whether we are dealing with a construction $Q_1 \times (Q_2 \times Q_3) \in \mathcal{V}(\mathcal{Q})$, which would mean that the faithful reordering of q is (a, b, c), or with a construction $Q_1 \times (Q_3 \times Q_2) \in \mathcal{V}(\mathcal{Q})$, which would result in (a, c, b) as the faithful reordering of q. □

Only if $D_i \cap D_j = \varnothing$ for any two sets of states of a set of automata, the above definitions could be simplified. This has never been a condition though.

Unpacking and reordering all elements of a nested cartesian product V over sets from $\mathcal{D}$ (relative to the construction of V) results in the cartesian product (over sets from $\mathcal{D}$) according to J. This is formally stated in the following lemma.

Lemma 4.3.7. *If $V \in \mathcal{V}(\mathcal{D})$ and $\mathrm{dom}(V) = J'$, then $\{\langle v \rangle_V \mid v \in V\} = \prod_{j \in J'} D_j$.*

Proof. Let $V \in \mathcal{V}(\mathcal{D})$ and let $\mathrm{dom}(V) = J'$.

($\subseteq$) Let $v \in V$. By Definition 4.3.4 we have $\langle v \rangle_V = \prod_{j \in J'} u_V(v, j)$. Now we only have to prove that $u_V(v, j) \in D_j$, for all $j \in J'$. We do this by structural induction.
If $J' = \{j\}$ and $V = D_j$, then $u_V(v, j) = v \in V = D_j$.
Next assume that $V = \prod_{\ell \in L} V_\ell$, with $V_\ell \in \mathcal{V}(\mathcal{D})$ for all $\ell \in L$. Then, by Definition 4.3.4, for all $j \in J'$, $u_V(v, j) = u_{V_k}(\mathrm{proj}_k(v), j)$, where k is such that $j \in \mathrm{dom}(V_k)$. Since each $V_k \in \mathcal{V}(\mathcal{D})$, the depth of its nesting is strictly less than the depth of the nesting in V. Thus by the induction hypothesis, $u_{V_k}(\mathrm{proj}_k(v), j) \in D_j$, for all $j \in \mathrm{dom}(V_k)$, which completes this direction of the proof.

($\supseteq$) Let $d \in \prod_{j \in J'} D_j$. Then we only have to prove that there exists a $v \in V$ such that $\langle v \rangle_V = d$ or, equivalently, that there exists a $v \in V$ such that for all $j \in J'$, $u_V(v, j) = \mathrm{proj}_j(d)$. We do this by structural induction.
Assume that $J' = \{j\}$ and $V = D_j$. Now set $v = \mathrm{proj}_j(d)$. Then $u_V(v, j) = v = \mathrm{proj}_j(d)$.
Next assume that $V = \prod_{\ell \in L} V_\ell$. Then from the induction hypothesis it follows that for all $\ell \in L$, $\{\langle v_\ell \rangle_{V_\ell} \mid v_\ell \in V_\ell\} = \prod_{j \in J_\ell} D_j$ where $J_\ell = \mathrm{dom}(V_\ell)$. Hence for all $\ell \in L$ and for all $j \in J_\ell$ we have a $v_\ell \in V_\ell$ such that $u_{V_\ell}(v_\ell, j) = \mathrm{proj}_j(d) \in D_j$. Let $v \in V$ be such that for all $\ell \in L$, $\mathrm{proj}_\ell(v) = v_\ell$ with $v_\ell \in V_\ell$. Then for all $j \in J'$, $u_V(v, j) = u_{V_\ell}(\mathrm{proj}_\ell(v), j)$, where ℓ is such that $j \in \mathrm{dom}(V_\ell)$. Since for all $\ell \in L$, $u_{V_\ell}(\mathrm{proj}_\ell(v), j) = u_{V_\ell}(v_\ell, j) = \mathrm{proj}_j(d)$, this completes also this direction of the proof. □

Now we are ready to return to the issue of iteratively forming a synchronized automaton, given a set of synchronized automata. We begin by generalizing the notion of a synchronized automaton.

Definition 4.3.8. *$\mathcal{T}$ is an* iterated synchronized automaton over $\mathcal{S}$ *if either*

(1) $\mathcal{T}$ is a synchronized automaton over $\mathcal{S}$, or

(2) $\mathcal{T}$ is a synchronized automaton over $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$, where each $\mathcal{T}_j$ is an iterated synchronized automaton over $\{\mathcal{A}_i \mid i \in \mathcal{I}_j\}$, for some $\mathcal{I}_j \subseteq \mathcal{I}$, and $\{\mathcal{I}_j \mid j \in \mathcal{J}\}$ forms a partition of $\mathcal{I}$. □

We see that iterated synchronized automata indeed are a generalization of synchronized automata: every synchronized automaton over a given set of automata may also be viewed as an iterated synchronized automaton over that set. But, as announced in the beginning of this section, synchronized automata formed iteratively over a set of automata are essentially synchronized automata over that set. Intuitively the only difference lies in the ordering and grouping of the elements from the set of automata. In the remainder of this section, we will formalize this statement.

The following lemma shows that the set of (initial) states of an iterated synchronized automaton over a set of automata is — upto a reordering — the same as the set of (initial) states of any synchronized automaton over that set.

Lemma 4.3.9. *Let $\mathcal{T} = (P, \Gamma, \gamma, J)$ be an iterated synchronized automaton over $\mathcal{S}$. Let $\mathcal{Q} = \{Q_i \mid i \in \mathcal{I}\}$. Then*

(1) $P \in \mathcal{V}(\mathcal{Q})$ and $dom(P) = \mathcal{I}$,

(2) $\{\langle q \rangle_P \mid q \in P\} = \prod_{i \in \mathcal{I}} Q_i$, and

(3) $\{\langle q \rangle_P \mid q \in J\} = \prod_{i \in \mathcal{I}} I_i$.

Proof. If $\mathcal{T}$ is a synchronized automaton over $\mathcal{S}$, then $P = \prod_{i \in \mathcal{I}} Q_i$ and $J = \prod_{i \in \mathcal{I}} I_i$.
By Definition 4.3.2(2) we have $P \in \mathcal{V}(\mathcal{Q})$ and $\mathrm{dom}(P) = \bigcup_{i \in \mathcal{I}} \mathrm{dom}(Q_i) = \mathcal{I}$.
By Lemma 4.3.7 we have $\{\langle q \rangle_P \mid q \in P\} = \prod_{i \in \mathcal{I}} Q_i$.
Since according to Definition 4.3.4 for all $q \in P$, $\langle q \rangle_P = \prod_{i \in \mathcal{I}} u_P(q, i) = \prod_{i \in \mathcal{I}} u_{Q_i}(\mathrm{proj}_i(q), i) = \prod_{i \in \mathcal{I}} \mathrm{proj}_i(q) = q$, it follows that $\{\langle q \rangle_P \mid q \in J\} = \{q \mid q \in \prod_{i \in \mathcal{I}} I_i\} = \prod_{i \in \mathcal{I}} I_i$.

Now assume that $\mathcal{T}$ is an iterated synchronized automaton over $\mathcal{S}$. Hence $\mathcal{T}$ is a synchronized automaton over a set of automata $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$, where $\mathcal{J} \subseteq \mathbb{N}$, $\{\mathcal{I}_j \mid j \in \mathcal{J}\}$ forms a partition of $\mathcal{I}$, and each $\mathcal{T}_j$ is an iterated synchronized automaton over $\{\mathcal{A}_i \mid i \in \mathcal{I}_j\}$. Let, for $j \in \mathcal{J}$, $\mathcal{T}_j$ be specified as $\mathcal{T}_j = (P_j, \Gamma_j, \gamma_j, J_j)$. Hence $P = \prod_{j \in \mathcal{J}} P_j$ and $J = \prod_{j \in \mathcal{J}} J_j$. As induction hypothesis we assume that for all $j \in \mathcal{J}$, $P_j \in \mathcal{V}(\mathcal{Q})$ with $\mathrm{dom}(P_j) = \mathcal{I}_j$, and

$\{\langle q\rangle_{P_j} \mid q \in J_j\} = \prod_{i\in\mathcal{I}_j} I_i$.
Since $\{\mathcal{I}_j \mid j \in \mathcal{J}\}$ forms a partition of $\mathcal{I}$, we immediately have $P = \prod_{j\in\mathcal{J}} P_j \in \mathcal{V}(\mathcal{Q})$ and $\mathrm{dom}\,(P) = \bigcup_{j\in\mathcal{J}} \mathrm{dom}\,(P_j) = \bigcup_{j\in\mathcal{J}} \mathcal{I}_j = \mathcal{I}$.
By Lemma 4.3.7 we have $\{\langle q\rangle_P \mid q \in P\} = \prod_{i\in\mathcal{I}} Q_i$.
Furthermore, $q \in J$ if and only if $\mathrm{proj}_j(q) \in J$, for all $j \in \mathcal{J}$. By the induction hypothesis, for all $j \in \mathcal{J}$, $\mathrm{proj}_j(q) \in J_j$ if and only if $\langle \mathrm{proj}_j(q)\rangle_{P_j} = \prod_{i\in\mathcal{I}_j} u_{P_j}(\mathrm{proj}_j(q), i) \in \prod_{i\in\mathcal{I}_j} I_i$. Thus $q \in J$ if and only if for all $j \in \mathcal{J}$ and for all $i \in \mathcal{I}_j$, $u_{P_j}(\mathrm{proj}_j(q), i) \in I_i$. Since for all $q \in P$, $\langle q\rangle_P = \prod_{i\in\mathcal{I}} u_P(q, i) = \prod_{i\in\mathcal{I}} u_{P_{k_i}}(\mathrm{proj}_{k_i}(q), i)$, where $k_i \in \mathcal{J}$ is such that $i \in \mathrm{dom}\,(P_{k_i})$, it follows that $\{\langle q\rangle_P \mid q \in J\} = \prod_{i\in\mathcal{I}} I_i$. □

Next we consider the actions and transitions of iterated synchronized automata. The actions of an iterated synchronized automaton over a set of automata $\mathcal{S}$ are the same as the actions of any synchronized automaton over $\mathcal{S}$. Furthermore, the transitions of any synchronized automaton over $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$ are — after reordering — the transitions of a synchronized automaton over $\mathcal{S}$.

Lemma 4.3.10. *Let $\mathcal{T} = (P, \Gamma, \gamma, J)$ be an iterated synchronized automaton over $\mathcal{S}$. Then*

(1) $\Gamma = \bigcup_{i\in\mathcal{I}} \Sigma_i$ and

(2) $\{(\langle q\rangle_P, \langle q'\rangle_P) \mid (q, q') \in \gamma_a\} \subseteq \Delta_a(\mathcal{S})$, for all $a \in \Gamma$.

Proof. If $\mathcal{T}$ is a synchronized automaton over $\mathcal{S}$, then (1) follows immediately from Definition 4.1.2. In that case also (2) follows immediately from Definition 4.1.2 because, as in the proof of Lemma 4.3.9, $\langle q\rangle_P = q$, for all $q \in P$.

Now assume that $\mathcal{T}$ is a synchronized automaton over $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$, where $\mathcal{J} \subseteq \mathbb{N}$, and each $\mathcal{T}_j = (P_j, \Gamma_j, \gamma_j, J_j)$ is an iterated synchronized automaton over $\{\mathcal{A}_i \mid i \in \mathcal{I}_j\}$, with $\{\mathcal{I}_j \mid j \in \mathcal{J}\}$ forming a partition of $\mathcal{I}$. Assume furthermore inductively that for all $j \in \mathcal{J}$, $\Gamma_j = \bigcup_{i\in\mathcal{I}_j} \Sigma_i$. Then $\Gamma = \bigcup_{j\in\mathcal{J}} \Gamma_j = \bigcup_{j\in\mathcal{J}} \bigcup_{i\in\mathcal{I}_j} \Sigma_i = \bigcup_{i\in\mathcal{I}} \Sigma_i$, by Definition 4.1.2, and because $\{\mathcal{I}_j \mid j \in \mathcal{J}\}$ forms a partition of $\mathcal{I}$.

Consequently we consider the transitions of $\mathcal{T}$. Let $a \in \Gamma$. Since $\mathcal{T}$ is a synchronized automaton over $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$, we know that $\gamma_a \subseteq \Delta_a(\{\mathcal{T}_j \mid j \in \mathcal{J}\})$. We have to prove that — upto the reordering relative to the construction of P — every a-transition of $\mathcal{T}$ is an element of the complete transition space of a in $\mathcal{S}$. In order to prove this we make inductively the following assumption. For all $j \in \mathcal{J}$, $\{(\langle p\rangle_{P_j}, \langle p'\rangle_{P_j}) \mid (p, p') \in \gamma_{j,a}\} \subseteq \Delta_a(\{\mathcal{A}_i \mid i \in \mathcal{I}_j\})$.

Before we turn to the proof we make the following auxiliary observation. Let $q \in P$. By Lemma 4.3.9 we have $\langle q\rangle_P \in \prod_{i\in\mathcal{I}} Q_i$ and thus $\langle q\rangle_P =$

$\prod_{i\in\mathcal{I}} \mathrm{proj}_i(\langle q\rangle_P)$. Let $i \in \mathcal{I}$. By Definition 4.3.4 we have $\mathrm{proj}_i(\langle q\rangle_P) = u_P(q,i) = u_{P_j}(\mathrm{proj}_j(q), i)$, where j is such that $i \in \mathcal{I}_j$. Now $\mathrm{proj}_j(q) \in P_j$ and hence, again by Lemma 4.3.9, $\langle \mathrm{proj}_j(q)\rangle_{P_j} \in \prod_{i\in\mathcal{I}_j} Q_i$. By Definition 4.3.4 once again we have $\mathrm{proj}_i(\langle \mathrm{proj}_j(q)\rangle_{P_j}) = u_{P_j}(\mathrm{proj}_j(q), i)$, whenever $i \in \mathcal{I}_j$. Hence $\mathrm{proj}_i(\langle q\rangle_P) = \mathrm{proj}_i(\langle \mathrm{proj}_j(q)\rangle_{P_j})$, for all $q \in P$, $i \in \mathcal{I}_j$, and $j \in \mathcal{J}$. This ends the observation.

Now let $(q, q') \in \gamma_a$. In order to prove that $(\langle q\rangle_P, \langle q'\rangle_P) \in \Delta_a(\mathcal{S})$ we verify the two conditions in Definition 4.1.1.

First we prove that there exists an $i \in \mathcal{I}$ such that $\mathrm{proj}_i{}^{[2]}(\langle q\rangle_P, \langle q'\rangle_P) \in \delta_{i,a}$. Let $j \in \mathcal{J}$ be such that $\mathrm{proj}_j{}^{[2]}(q, q') \in \gamma_{j,a}$. Such a j exists because $\gamma_a \subseteq \Delta_a(\{\mathcal{T}_j \mid j \in \mathcal{J}\})$. By the induction hypothesis we have $(\langle \mathrm{proj}_j(q)\rangle_{P_j}, \langle \mathrm{proj}_j(q')\rangle_{P_j}) \in \Delta_a(\{\mathcal{A}_i \mid i \in \mathcal{I}_j\})$. Hence by Definition 4.1.1 there exists an $i \in \mathcal{I}_j$ such that $\mathrm{proj}_i{}^{[2]}(\langle \mathrm{proj}_j(q)\rangle_{P_j}, \langle \mathrm{proj}_j(q')\rangle_{P_j}) \in \delta_{i,a}$. Thus, by our observation above, for this i we have $\mathrm{proj}_i{}^{[2]}(\langle q\rangle_P, \langle q'\rangle_P) \in \delta_{i,a}$, as desired.

Secondly, we prove that for all $i \in \mathcal{I}$, either $\mathrm{proj}_i{}^{[2]}(\langle q\rangle_P, \langle q'\rangle_P) \in \delta_{i,a}$ or $\mathrm{proj}_i(\langle q\rangle_P) = \mathrm{proj}_i(\langle q'\rangle_P)$. Let $i \in \mathcal{I}$ and let $j \in \mathcal{J}$ be such that $i \in \mathcal{I}_j$. Because $\{\mathcal{I}_j \mid j \in \mathcal{J}\}$ forms a partition of $\mathcal{I}$ such a j exists and is unique. Since $\gamma_a \subseteq \Delta_a(\{\mathcal{T}_j \mid j \in \mathcal{J}\})$, Definition 4.1.1 implies that either $\mathrm{proj}_j^{[2]}(q, q') \in \gamma_{j,a}$ or $\mathrm{proj}_j(q) = \mathrm{proj}_j(q')$.

If $\mathrm{proj}_j^{[2]}(q, q') \in \gamma_{j,a}$, then $(\langle \mathrm{proj}_j(q)\rangle_{P_j}, \langle \mathrm{proj}_j(q')\rangle_{P_j}) \in \Delta_a(\{\mathcal{A}_i \mid i \in \mathcal{I}_j\})$ by the induction hypothesis. Hence by Definition 4.1.1, we get that either $\mathrm{proj}_i{}^{[2]}(\langle \mathrm{proj}_j(q)\rangle_{P_j}, \langle \mathrm{proj}_j(q')\rangle_{P_j}) \in \delta_{i,a}$, which — by the above auxiliary observation — implies that $\mathrm{proj}_i{}^{[2]}(\langle q\rangle_P, \langle q'\rangle_P) \in \delta_{i,a}$, or $\mathrm{proj}_i(\langle \mathrm{proj}_j(q)\rangle_{P_j}) = \mathrm{proj}_i(\langle \mathrm{proj}_j(q')\rangle_{P_j})$, which — again by the above auxiliary observation — implies that $\mathrm{proj}_i(\langle q\rangle_P) = \mathrm{proj}_i(\langle q'\rangle_P)$.

If $\mathrm{proj}_j(q) = \mathrm{proj}_j(q')$, then $\mathrm{proj}_i(\langle q\rangle_P) = u_{P_j}(\mathrm{proj}_j(q), i) = u_{P_j}(\mathrm{proj}_j(q'), i) = \mathrm{proj}_i(\langle q'\rangle_P)$, which completes the proof. □

Note that this lemma states that for each action a its complete transition space in $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$ is included — after reordering — in its complete transition space in $\mathcal{S}$. Iteration in the construction of a synchronized automaton thus does not lead to an increase of the number of possibilities for synchronization. In other words, every iterated synchronized automaton over a set of automata can be interpreted as a synchronized automaton over that set, by reordering its state space and transition space.

Definition 4.3.11. *Let $\mathcal{T} = (Q, \Sigma, \delta, I)$ be an iterated synchronized automaton over $\mathcal{S}$. Then the* reordered version of $\mathcal{T}$ w.r.t. $\mathcal{S}$ *is denoted by $\langle\langle \mathcal{T}\rangle\rangle_{\mathcal{S}}$ and is defined as*

$$\langle\langle\mathcal{T}\rangle\rangle_{\mathcal{S}} = (\{\langle q\rangle_Q \mid q \in Q\}, \Sigma,$$
$$\{(\langle q\rangle_Q, a, \langle q'\rangle_Q) \mid q, q' \in Q,\ (q, a, q') \in \delta\}, \{\langle q\rangle_I \mid q \in I\}).\ \square$$

From Lemmata 4.3.9 and 4.3.10 we conclude that $\langle\langle\mathcal{T}\rangle\rangle_{\mathcal{S}}$ is indeed a synchronized automaton over $\mathcal{S}$ whenever $\mathcal{T}$ is an iterated synchronized automaton over $\mathcal{S}$. In fact, $\langle\langle\mathcal{T}\rangle\rangle_{\mathcal{S}}$ is the interpretation of $\mathcal{T}$ as a synchronized automaton over $\mathcal{S}$ by reordering. Since their only difference is the ordering of the elements of their state spaces, it is immediate that $\langle\langle\mathcal{T}\rangle\rangle_{\mathcal{S}}$ and $\mathcal{T}$ have — upto a reordering — the same set of computations and thus the same behavior.

Theorem 4.3.12. *Let $\mathcal{T} = (Q, \Sigma, \delta, I)$ be an iterated synchronized automaton over $\mathcal{S}$ and let Θ be an alphabet disjoint from Q. Then*

(1) $\mathbf{C}^{\infty}_{\langle\langle\mathcal{T}\rangle\rangle_{\mathcal{S}}} = \{\langle q_0\rangle_Q a_1 \langle q_1\rangle_Q a_2 \langle q_2\rangle_Q \cdots \mid q_0 a_1 q_1 a_2 q_2 \cdots \in \mathbf{C}^{\infty}_{\mathcal{T}}\}$ *and*

(2) $\mathbf{B}^{\Theta,\infty}_{\langle\langle\mathcal{T}\rangle\rangle_{\mathcal{S}}} = \mathbf{B}^{\Theta,\infty}_{\mathcal{T}}$. $\square$

Clearly the converse of the inclusion of Lemma 4.3.10(2) in general does not hold, since synchronized automata — and hence also iterated synchronized automata — are equipped with only a subset of all possible synchronizations. Moreover, a given intermediate synchronized automaton $\mathcal{T}_j$ over a subset $\mathcal{S}_j$ of $\mathcal{S}$ may have a transition relation that is properly included in the complete transition space of $\mathcal{S}_j$. As a consequence, $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$ may provide less transitions for the forming of a synchronized automaton than $\{\mathcal{A}_i \mid i \in \mathcal{I}\}$ does. However, there is a natural condition that guarantees that for a given arbitrary synchronized automaton $\mathcal{T}$ over $\mathcal{S}$ and given iterated synchronized automata $\mathcal{T}_j$ over subsets $\mathcal{S}_j = \{\mathcal{A}_i \mid i \in \mathcal{I}_j\}$, where the $\mathcal{I}_j$ form a partition of $\mathcal{I}$, one can still obtain a synchronized automaton $\widehat{\mathcal{T}}$ over the set consisting of the $\mathcal{T}_j$, such that $\langle\langle\widehat{\mathcal{T}}\rangle\rangle_{\mathcal{S}} = \mathcal{T}$. This condition requires that each of the $\mathcal{T}_j$ has at least all transitions — after reordering — of the corresponding subautomaton of $\mathcal{T}$ determined by $\mathcal{I}_j$. In fact, when loops are ignored this is a necessary and sufficient condition for obtaining an iterated version of a given synchronized automaton over $\mathcal{S}$. Formally, we have the following result, where we recall $\delta_{\mathcal{I}_j}$ to be the transition relation of $SUB_{\mathcal{I}_j}(\mathcal{T})$.

Theorem 4.3.13. *Let $\mathcal{T} = (Q, \Sigma, \delta, I)$ be a synchronized automaton over $\mathcal{S}$ and let $\{\mathcal{I}_j \mid j \in \mathcal{J}\}$, where $\mathcal{J} \subseteq \mathbb{N}$, form a partition of $\mathcal{I}$. Let, for each $j \in \mathcal{J}$, $\mathcal{T}_j = (P_j, \Gamma_j, \gamma_j, J_j)$ be an iterated synchronized automaton over $\{\mathcal{A}_i \mid i \in \mathcal{I}_j\}$. Then*

(1) if $(\delta_{\mathcal{I}_j})_a \subseteq \{(\langle q\rangle_{P_j}, \langle q'\rangle_{P_j}) \mid (q, q') \in \gamma_{j,a}\}$, for all $a \in \Gamma_j$ for all $j \in \mathcal{J}$, then there exists a synchronized automaton $\widehat{\mathcal{T}}$ over $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$ such that $\langle\langle\widehat{\mathcal{T}}\rangle\rangle_{\mathcal{S}} = \mathcal{T}$, and

(2) if $\widehat{\mathcal{T}}$ is a synchronized automaton over $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$, then $\langle\langle\widehat{\mathcal{T}}\rangle\rangle_{\mathcal{S}} = \mathcal{T}$ implies that $(\delta_{\mathcal{I}_j})_a \setminus \{(p,p) \mid (p,p) \in \Delta_a(\{\mathcal{A}_i \mid i \in \mathcal{I}_j\})\} \subseteq \{(\langle q\rangle_{P_j}, \langle q'\rangle_{P_j}) \mid (q,q') \in \gamma_{j,a}\}$, for all $a \in \Gamma_j$ for all $j \in \mathcal{J}$.

Proof. Let $\widehat{\mathcal{T}} = (P, \Gamma, \gamma, J)$ be an arbitrary synchronized automaton over $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$. First we make an auxiliary observation similar to the one in the proof of Lemma 4.3.10. Let $q \in P$ and let $j \in \mathcal{J}$. Then $\mathrm{proj}_{\mathcal{I}_j}(\langle q\rangle_P) = \langle \mathrm{proj}_j(q)\rangle_{P_j}$, since $P = \prod_{j\in\mathcal{J}} P_j$ and, by Lemma 4.3.9(2), $\prod_{i\in\mathcal{I}_j} Q_i = \{\langle q\rangle_{P_j} \mid q \in P_j\}$.

(1) Assume that $(\delta_{\mathcal{I}_j})_a \subseteq \{(\langle q\rangle_{P_j}, \langle q'\rangle_{P_j}) \mid (q,q') \in \gamma_{j,a}\}$. By Lemmata 4.3.9(2), 4.3.10(1), and 4.3.9(3) we know that $Q = \{\langle q\rangle_P \mid q \in P\}$, $\Sigma = \Gamma$, and $I = \{\langle q\rangle_J \mid q \in J\}$, respectively. Thus it only remains to prove that the transition relation γ for $\widehat{\mathcal{T}}$ can be chosen in such a way that $\delta = \{(\langle q\rangle_P, a, \langle q'\rangle_P) \mid q, q' \in P,\ (q,a,q') \in \gamma\}$. Thus using the injectivity of reordering we define γ simply by $\gamma_a = \{(q,q') \in \prod_{j\in\mathcal{J}} P_j \times \prod_{j\in\mathcal{J}} P_j \mid (\langle q\rangle_P, \langle q'\rangle_P) \in \delta_a\}$, for all $a \in \Gamma$ and prove that this is indeed the transition relation of a synchronized automaton over $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$.

Let $(p,p') \in \gamma_a$. We prove there exists a $j \in \mathcal{J}$ so that ${\mathrm{proj}_j}^{[2]}(p,p') \in \gamma_{j,a}$. As $(\langle p\rangle_P, \langle p'\rangle_P) \in \delta_a$ there exists an $i \in \mathcal{I}$ such that ${\mathrm{proj}_j}^{[2]}(\langle p\rangle_P, \langle p'\rangle_P) \in \delta_{i,a}$. Let j be such that $i \in \mathcal{I}_j$. Then it follows that ${\mathrm{proj}_{\mathcal{I}_j}}^{[2]}(\langle p\rangle_P, \langle p'\rangle_P) \in (\delta_{\mathcal{I}_j})_a$. Since $(\delta_{\mathcal{I}_j})_a \subseteq \{(\langle q\rangle_{P_j}, \langle q'\rangle_{P_j}) \mid (q,q') \in \gamma_{j,a}\}$ there exists an $(r,r') \in \gamma_{j,a}$ such that $(\langle r\rangle_{P_j}, \langle r'\rangle_{P_j}) = {\mathrm{proj}_{\mathcal{I}_j}}^{[2]}(\langle p\rangle_P, \langle p'\rangle_P)$. Thus by the observation above we have $(\langle r\rangle_{P_j}, \langle r'\rangle_{P_j}) = (\langle \mathrm{proj}_j(p)\rangle_{P_j}, \langle \mathrm{proj}_j(p')\rangle_{P_j})$. Since reordering is an injective operation it follows that $r = \mathrm{proj}_j(p)$ and $r' = \mathrm{proj}_j(p')$, and thus ${\mathrm{proj}_j}^{[2]}(p,p') = (r,r') \in \gamma_{j,a}$.

It now remains to prove that for all $j \in \mathcal{J}$, either $\mathrm{proj}_j(p) = \mathrm{proj}_j(p')$ or ${\mathrm{proj}_j}^{[2]}(p,p') \in \gamma_{j,a}$. Let $j \in \mathcal{J}$ be such that $\mathrm{proj}_j(p) \neq \mathrm{proj}_j(p')$. Then we only have to prove that ${\mathrm{proj}_j}^{[2]}(p,p') \in \gamma_{j,a}$. Since $(p,p') \in \gamma_a$ we have $(\langle p\rangle_P, \langle p'\rangle_P) \in \delta_a$. By the observation above we have $\mathrm{proj}_{\mathcal{I}_j}(\langle p\rangle_P) = \langle \mathrm{proj}_j(p)\rangle_{P_j}$ and $\mathrm{proj}_{\mathcal{I}_j}(\langle p'\rangle_P) = \langle \mathrm{proj}_j(p')\rangle_{P_j}$. From the fact that reordering is an injective operation we infer that $\mathrm{proj}_{\mathcal{I}_j}(\langle p\rangle_P) \neq \mathrm{proj}_{\mathcal{I}_j}(\langle p'\rangle_P)$. Hence ${\mathrm{proj}_{\mathcal{I}_j}}^{[2]}(\langle p\rangle_P, \langle p'\rangle_P) \in (\delta_{\mathcal{I}_j})_a$. Since $(\delta_{\mathcal{I}_j})_a \subseteq \{(\langle q\rangle_{P_j}, \langle q'\rangle_{P_j}) \mid (q,q') \in \gamma_{j,a}\}$ it follows that ${\mathrm{proj}_j}^{[2]}(p,p') \in \gamma_{j,a}$.

(2) Now assume that $\langle\langle\widehat{\mathcal{T}}\rangle\rangle_{\mathcal{S}} = \mathcal{T}$. Let $j \in \mathcal{J}$ and $a \in \Gamma$ be fixed. Let $(p,p') \in (\delta_{\mathcal{I}_j})_a$ be such that $p \neq p'$. By Definition 4.1.6 there is a pair $(r,r') \in \delta_a$ such that ${\mathrm{proj}_{\mathcal{I}_j}}^{[2]}(r,r') = (p,p')$. Since $\langle\langle\widehat{\mathcal{T}}\rangle\rangle_{\mathcal{S}} = \mathcal{T}$ there are $(\hat{r},\hat{r}') \in \gamma_a$ such that $(\langle\hat{r}\rangle_P, \langle\hat{r}'\rangle_P) = (r,r')$. By the observation above we have $(p,p') = {\mathrm{proj}_{\mathcal{I}_j}}^{[2]}(r,r') = (\langle \mathrm{proj}_j(\hat{r})\rangle_{P_j}, \langle \mathrm{proj}_j(\hat{r}')\rangle_{P_j})$ and thus the only thing left to prove here is that $(\mathrm{proj}_j(\hat{r}), \mathrm{proj}_j(\hat{r}')) \in \gamma_{j,a}$. Assume to the contrary that this is not the case. Then the fact that $\widehat{\mathcal{T}}$ is a synchronized automaton

over $\{\mathcal{T}_j \mid j \in \mathcal{J}\}$, together with $(\hat{r}, \hat{r}') \in \gamma_a$, implies that $\mathrm{proj}_j(\hat{r}) = \mathrm{proj}_j(\hat{r}')$ and thus $p = p'$, a contradiction. Hence $(\mathrm{proj}_j(\hat{r}), \mathrm{proj}_j(\hat{r}')) \in \gamma_{j,a}$. □

Thus, not only can every iterated synchronized automaton over $\mathcal{S}$ be considered as a synchronized automaton directly constructed from $\mathcal{S}$ by Definition 4.3.11, but according to Theorem 4.3.13 also every synchronized automaton can be iteratively constructed from its subautomata. Consequently, both subautomata and iterated synchronized automata can be treated as synchronized automata — including the considerations concerning their computations and behavior — and it thus suffices to study only the relationship between (sub)automata and synchronized automata in the sequel, i.e. without considering iterated synchronized automata explicitly.

4.4 Synchronizations

As said before, the high level of flexibility that is obtained by leaving the set of transitions of a synchronized automaton as a modeling choice is an important — perhaps even the most important — feature of the team automata framework we are introducing. The choice for a specific interconnection strategy (which automata synchronize on what actions, and when) is based on the system one wants to model.

In this section we provide the basis for the introduction of a broad variety of often complex interconnection strategies for team automata in Section 5.3. We do so by introducing some basic and natural types of synchronization that can be expressed already within the synchronized automata underlying team automata.

We focus on the individual actions of a synchronized automaton and we distinguish several different ways of synchronizing on shared actions. We consider actions that are never used in synchronizations between multiple automata, as well as actions on which all automata having these actions have to synchronize. The latter case is weakened by requiring participation only if an automaton is in a state at which that action is enabled.

Recall that information on the actual execution of loops is missing in the transition relation of a synchronized automaton. In the coming definitions and their intuitive explanation, the presence of loops on action a in automata is treated as if a is actually executed, which is in accordance with the maximal interpretation of the participation of automata adopted in Section 4.2.

Notation 2. *For the remainder of this chapter we assume $\mathcal{T} = (Q, \Sigma, \delta, I)$ is an arbitrary but fixed synchronized automaton over our fixed set $\mathcal{S}$ of au-*

tomata. Note that Σ is the alphabet of any synchronized automaton over $\mathcal{S}$ (i.e. not only of $\mathcal{T}$). □

4.4.1 Free

Intuitively, an action a is a *free* action of $\mathcal{T}$ if no a-transition of $\mathcal{T}$ is brought about by a simultaneous execution of a by two or more automata. Thus, whenever a is executed by $\mathcal{T}$ only one automaton is active in this execution.

Definition 4.4.1. *The set of* free *actions of $\mathcal{T}$ is denoted by $Free(\mathcal{T})$ and is defined as*

$$Free(\mathcal{T}) = \{a \in \Sigma \mid (q, q') \in \delta_a \Rightarrow \\ \#\{i \in \mathcal{I} \mid a \in \Sigma_i \wedge proj_i{}^{[2]}(q, q') \in \delta_{i,a}\} = 1\}.$$ □

Example 4.4.2. (Example 4.1.3 continued) Actions a and b both are not *free* in synchronized automaton $\mathcal{T}_{\{1,2\}}$. This can be concluded from the fact that the a-transition $((s_1, s_2), a, (t_1, t_2))$ and the b-transition $((t_1, t_2), b, (s_1, s_2))$ can serve as an example of a simultaneous execution of a and b, respectively, by two automata. In synchronized automaton $\mathcal{T}'_{\{1,2\}}$, however, action a is *free* while action b is not *free*. □

4.4.2 Action-Indispensable

If an action a is *action-indispensable*, then all automata which have a as one of their actions are involved in every execution of a by $\mathcal{T}$. This means that $\mathcal{T}$ cannot execute an a if there is an automaton to which a belongs but in which it is not enabled at the current local state.

Definition 4.4.3. *The set of* action-indispensable *(ai for short) actions of $\mathcal{T}$ is denoted by $AI(\mathcal{T})$ and is defined as*

$$AI(\mathcal{T}) = \{a \in \Sigma \mid \forall i \in \mathcal{I} : (a \in \Sigma_i \wedge (q, q') \in \delta_a) \Rightarrow \\ proj_i{}^{[2]}(q, q') \in \delta_{i,a}\}.$$ □

Example 4.4.4. (Example 4.4.2 continued) Actions a and b both are *ai* in the synchronized automaton $\mathcal{T}_{\{1,2\}}$. This follows directly from the fact that in all of the a-transitions and in all of the b-transitions of $\mathcal{T}_{\{1,2\}}$, both W_1 and W_2 participate. Hence b is also *ai* in $\mathcal{T}'_{\{1,2\}}$, while a however is not *ai* in $\mathcal{T}'_{\{1,2\}}$. This difference stems from the fact that in the a-transition $((s_1, s_2), a, (s_1, t_2))$ only W_2 participates while also W_1 has a in its alphabet. □

4.4.3 State-Indispensable

State-indispensable, finally, is a weak version of action-indispensable: if an action a is state-indispensable, then all executions of a by $\mathcal{T}$ involve all automata in which a is enabled at the current local state. In this case $\mathcal{T}$ does not have to "wait" with the execution of a until a is enabled in all automata to which it belongs.

Definition 4.4.5. *The set of* state-indispensable *(si for short) actions of* $\mathcal{T}$ *is denoted by* $SI(\mathcal{T})$ *and is defined as*

$$SI(\mathcal{T}) = \{a \in \Sigma \mid \forall i \in \mathcal{I} : (a \in \Sigma_i \wedge (q, q') \in \delta_a \wedge a \ en_{\mathcal{A}_i} \ proj_i(q)) \Rightarrow proj_i^{[2]}(q, q') \in \delta_{i,a}\}. \ \square$$

Example 4.4.6. (Example 4.4.4 continued) Actions a and b both are *si* in the synchronized automaton $\mathcal{T}_{\{1,2\}}$. This follows immediately from the fact that in all of the a-transitions as well as in all of the b-transitions of $\mathcal{T}_{\{1,2\}}$, both W_1 and W_2 participate. Hence b is also *si* in $\mathcal{T}'_{\{1,2\}}$, whereas a is not *si* in $\mathcal{T}'_{\{1,2\}}$. This is due to the fact that in the a-transition $((s_1, s_2), a, (s_1, t_2))$ only W_2 participates, while at state (s_1, s_2) action a is also enabled at the local state s_1 of W_1. $\square$

4.4.4 Free, Action-Indispensable, and State-Indispensable

We now compare the three types of synchronization introduced in this section.

It is immediate that all *ai* actions in $\mathcal{T}$ also satisfy the weaker requirement of being *si* actions.

Lemma 4.4.7. $AI(\mathcal{T}) \subseteq SI(\mathcal{T})$.

In fact, as we show next, this lemma describes the only dependency among *free*, *ai*, and *si* actions.

The combination of the properties of being *free*, *ai*, and *si* leads in principle to eight different types of actions in a synchronized automaton. However, by Lemma 4.4.7, *ai* implies *si*, which eliminates the combinations ⟨*free*, *ai*, not *si*⟩ and ⟨not *free*, *ai*, not *si*⟩. Each of the remaining six combinations is feasible, as we demonstrate in the following example.

Example 4.4.8. Consider the automata $\mathcal{A}_1 = (\{q, q'\}, \{a\}, \{(q, a, q')\}, \{q\})$ and $\mathcal{A}_2 = (\{r, r'\}, \{a\}, \{(r, a, r')\}, \{r\})$, as depicted in Figure 4.10.

From $\{\mathcal{A}_1, \mathcal{A}_2\}$ we construct the following five synchronized automata $\mathcal{T}^i = (\{(q, r), (q, r'), (q', r), (q', r')\}, \{a\}, \delta^i, \{(q, r)\})$, with $i \in [5]$, where

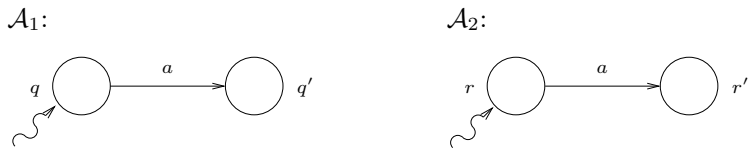

Fig. 4.10. Automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

$\delta^1 = \{((q,r),a,(q,r')),((q,r),a,(q',r'))\}$; now a is not *free* since both automata execute a in the second transition, while a is not *si* (and thus also not *ai*) since $\mathcal{A}_1$ does not execute a in the first transition, even though it is in a state at which a is enabled,

$\delta^2 = \{((q,r),a,(q',r'))\}$; now a is not *free* since in the given transition a is executed by both automata, which implies that a is *ai* and thus *si*,

$\delta^3 = \{((q,r),a,(q',r))\}$; now a is *free* since only one automaton is involved in the a-transition, but a is not *si* (and thus also not *ai*) since $\mathcal{A}_2$ does not execute a even though it is in a state at which a is enabled,

$\delta^4 = \{((q,r'),a,(q',r'))\}$; now a is *free* for the same reason as in the previous case, a is not *ai* since $\mathcal{A}_2$ does have a in its alphabet but nevertheless does not execute a, and a is *si* since $\mathcal{C}_2$ cannot execute a in state r' (a is not enabled at state r'), and

$\delta^5 = \varnothing$; now a trivially is *free*, *ai*, and *si*.

These synchronized automata $\mathcal{T}^1$, $\mathcal{T}^2$, $\mathcal{T}^3$, $\mathcal{T}^4$, and $\mathcal{T}^5$ thus illustrate the cases $\langle$not *free*, not *ai*, not *si*$\rangle$, $\langle$not *free*, *ai*, *si*$\rangle$, $\langle$*free*, not *ai*, not *si*$\rangle$, $\langle$*free*, not *ai*, *si*$\rangle$, and $\langle$*free*, *ai*, *si*$\rangle$, respectively.

It is not difficult to check that action a is *si* but neither *free* nor *ai* in the synchronized automaton $\mathcal{T}$ of Example 4.2.1, depicted in Figure 4.6(b). This concludes our display of the remaining six combinations. □

We conclude by noting that the definitions of *free*, *ai*, and *si* synchronizations are based on the maximal interpretation adopted in Section 4.2. We will come back to this in Subsection 7.2.1, where we will reconsider *free*, *ai*, and *si* synchronizations in a context in which precise information on the participation of loops in synchronizations is available.

4.5 Predicates of Synchronizations

Our exposition until now has been analytical, in the sense that we have investigated transition relations to determine whether or not they satisfy the

conditions inherent to certain types of synchronization. These conditions in general do not lead to uniquely defined synchronized automata.

In this section we deal with the question of how to describe a unique synchronized automaton, given a set of automata and certain conditions to be satisfied by the synchronizations. Recall that all elements of a synchronized automaton, except for its set of transitions, are uniquely determined by the set of automata it is composed over.

We begin by describing specific synchronized automata satisfying certain constraints on synchronizations. Synchronization constraints for an action a are conditions on the a-transitions to be chosen from $\Delta_a(\mathcal{S})$, the complete transition space of a in $\mathcal{S}$. Together, these conditions should determine a unique subset $\mathcal{R}_a$, which will be the set of a-transitions in the synchronized automaton. We will refer to subsets of the complete transition space $\Delta_a(\mathcal{S})$ as *predicates* (*of synchronizations*) for a. Once predicates have been chosen for all actions, the synchronized automaton over $\mathcal{S}$ defined by these predicates is unique.

The following generic definition formalizes this setup.

Definition 4.5.1. *For all $a \in \Sigma$, let $\mathcal{R}_a(\mathcal{S}) \subseteq \Delta_a(\mathcal{S})$ and let $\mathcal{R} = \{\mathcal{R}_a(\mathcal{S}) \mid a \in \Sigma\}$. Then $\mathcal{T}$ is* the $\mathcal{R}$-synchronized automaton *over $\mathcal{S}$ if for all $a \in \Sigma$,*

$$\delta_a = \mathcal{R}_a(\mathcal{S}).$$ □

A natural way of fixing a predicate for a given type of synchronization is to apply a maximality principle. Since a predicate is a subset of the complete transition space, this amounts to including everything that is not forbidden, i.e. everything that is in accordance with the chosen type of synchronization. This is the intuitive approach of [Ell97] and generalizes the classical approach to define synchronized systems from *ai* to other types of synchronization (cf. the Introduction). Thus when a synchronized automaton is to be constructed according to a specification of synchronization conditions for its set of actions, the strategy is to include as many transitions as possible without violating the specification, while checking that the result is unique.

This leads to the following predicates.

Definition 4.5.2. *Let $a \in \Sigma$. Then*

(1) the predicate no-constraints *in $\mathcal{S}$ for a is denoted by $\mathcal{R}_a^{no}(\mathcal{S})$ and is defined as*

$$\mathcal{R}_a^{no}(\mathcal{S}) = \Delta_a(\mathcal{S}),$$

(2) the predicate is-free *in $\mathcal{S}$ for a is denoted by $\mathcal{R}_a^{free}(\mathcal{S})$ and is defined as*

$\mathcal{R}_a^{free}(\mathcal{S}) = \{(q,q') \in \Delta_a(\mathcal{S}) \,|\, \#\{i \in \mathcal{I} \mid a \in \Sigma_i \wedge proj_i{}^{[2]}(q,q') \in \delta_{i,a}\} = 1\}$,

(3) the predicate is-ai *in* $\mathcal{S}$ *for* a *is denoted by* $\mathcal{R}_a^{ai}(\mathcal{S})$ *and is defined as*

$\mathcal{R}_a^{ai}(\mathcal{S}) = \{(q,q') \in \Delta_a(\mathcal{S}) \mid \forall i \in \mathcal{I} : a \in \Sigma_i \Rightarrow proj_i{}^{[2]}(q,q') \in \delta_{i,a}\}$, *and*

(4) the predicate is-si *in* $\mathcal{S}$ *for* a *is denoted by* $\mathcal{R}_a^{si}(\mathcal{S})$ *and is defined as*

$\mathcal{R}_a^{si}(\mathcal{S}) = \{(q,q') \in \Delta_a(\mathcal{S}) \mid \forall i \in \mathcal{I} : (a \in \Sigma_i \wedge a \ en_{\mathcal{A}_i} \ proj_i(q)) \Rightarrow proj_i{}^{[2]}(q,q') \in \delta_{i,a}\}$. □

Each of these predicates selects, for a given action a, *all* transitions from its complete transition space $\Delta_a(\mathcal{S})$ that obey a certain type of synchronization. In the case of *no-constraints* for a, this means that all a-transitions are allowed since nothing is required (and thus no transition is forbidden). In the other three cases, *all and only* those a-transitions are included that respect the specified property of a.

Theorem 4.5.3. *Let* $a \in \Sigma$. *Then*

(1) $a \in Free(\mathcal{T})$ *if and only if* $\delta_a \subseteq \mathcal{R}_a^{free}(\mathcal{S})$,

(2) $a \in AI(\mathcal{T})$ *if and only if* $\delta_a \subseteq \mathcal{R}_a^{ai}(\mathcal{S})$, *and*

(3) $a \in SI(\mathcal{T})$ *if and only if* $\delta_a \subseteq \mathcal{R}_a^{si}(\mathcal{S})$.

Proof. Immediately from Definitions 4.4.1, 4.4.3, 4.4.5, and 4.5.2. □

The predicate $\mathcal{R}_a^{free}(\mathcal{S})$ ($\mathcal{R}_a^{ai}(\mathcal{S})$, $\mathcal{R}_a^{si}(\mathcal{S})$) thus defines the largest transition relation in $\Delta_a(\mathcal{S})$ in which an action a is *free* (*ai*, *si*). In other words, each of the types of synchronization introduced in the previous section gives rise to a predicate that is the unique maximal representative among all transition relations satisfying the type of synchronization.

Definition 4.5.4. *Let* $syn \in \{free, ai, si\}$. *Then*

(1) the $\{\mathcal{R}_a^{syn}(\mathcal{S}) \mid a \in \Sigma\}$*-synchronized automaton over* $\mathcal{S}$ *is called the* maximal-syn synchronized automaton *(over* $\mathcal{S}$*) and*

(2) an action $a \in \Sigma$ *is called* maximal-syn *in* $\mathcal{T}$ *if* $\delta_a = \mathcal{R}_a^{syn}(\mathcal{S})$. □

In case the automata from $\mathcal{S}$ have no shared actions, then the *maximal-free* (*maximal-ai*, *maximal-si*) synchronized automaton equals the $\mathcal{R}^{no}$-synchronized automaton (over $\mathcal{S}$).

Theorem 4.5.5. *Let* $a \in \Sigma_j \setminus (\bigcup_{i \in \mathcal{I} \setminus \{j\}} \Sigma_i)$. *Then*

$\mathcal{R}_a^{no}(\mathcal{S}) = \mathcal{R}_a^{syn}(\mathcal{S})$, *for all* $syn \in \{free, ai, si\}$. □

4.6 Effect of Synchronizations

In this section we study the effect that the types of synchronization introduced in the previous sections have on the inheritance of the automata-theoretic properties from Section 3.2. We investigate both top-down inheritance — from synchronized automata to their (sub)automata — and bottom-up preservation — from (sub)automata to synchronized automata.

Notation 3. *For the remainder of this chapter we fix an arbitrary $j \in \mathcal{I}$ and an arbitrary subset $J \subseteq \mathcal{I}$. The subautomaton SUB_J of $\mathcal{T}$ will be specified as $SUB_J = (Q_J, \Sigma_J, \delta_J, I_J)$. We moreover fix Θ to be an arbitrary alphabet disjoint from Q.* □

The properties whose inheritance we study are static, in the sense that they depend on the mere "presence" of transitions in (sub)automata and synchronized automata. We begin by introducing two useful auxiliary notions.

A transition (p, a, p') of automaton $\mathcal{A}_j$ defines the execution of an action a by taking $\mathcal{A}_j$ from a (local) state p to a (local) state p'. Such a transition is *present* in the synchronized automaton $\mathcal{T}$ if it participates in one or more of the transitions of $\mathcal{T}$. In other words, if $\mathcal{T}$ can execute a by going from a (global) state q such that $\mathrm{proj}_j(q) = p$ to a (global) state q' such that $\mathrm{proj}_j(q') = p'$. The transition (p, a, p') is *omnipresent* in $\mathcal{T}$ if for all (global) states q of $\mathcal{T}$ such that $\mathrm{proj}_j(q) = p$, it can always be executed by participating in an a-transition (q, a, q') of $\mathcal{T}$ with $\mathrm{proj}_j(q') = p'$. The presence and omnipresence of transitions of SUB_J is defined likewise.

Definition 4.6.1. *(1) Let $(p, a, p') \in \delta_J$. Then*

- *(a) (p, a, p') is* present *in $\mathcal{T}$ if there exists a $(q, a, q') \in \delta$ such that $(proj_J(q), a, proj_J(q')) = (p, a, p')$ and*
- *(b) (p, a, p') is* omnipresent *in $\mathcal{T}$ if for all $q \in Q$ such that $proj_J(q) = p$, there exists a $(q, a, q') \in \delta$ such that $proj_J(q') = p'$.*

(2) Let $(p, a, p') \in \delta_j$. Then

- *(a) (p, a, p') is* present *in $\mathcal{T}$ if there exists a $(q, a, q') \in \delta$ such that $(proj_j(q), a, proj_j(q')) = (p, a, p')$ and*
- *(b) (p, a, p') is* omnipresent *in $\mathcal{T}$ if for all $q \in Q$ such that $proj_j(q) = p$, there exists a $(q, a, q') \in \delta$ such that $proj_j(q') = p'$.* □

Note that any transition of a (sub)automaton that is omnipresent in $\mathcal{T}$ is also present in $\mathcal{T}$.

We now investigate which conditions guarantee the presence or even omnipresence of the transitions of (sub)automata in synchronizations of synchronized automata over these (sub)automata. We are particularly interested in the presence or omnipresence of transitions in case of *free*, *ai*, and *si* actions.

As the transitions of any subautomaton of $\mathcal{T}$ are obtained from transitions of $\mathcal{T}$ by projection, each transition of a subautomaton of $\mathcal{T}$ is present — but not necessarily omnipresent — in $\mathcal{T}$.

Theorem 4.6.2. *Each transition of SUB_J is present in $\mathcal{T}$.* □

Since the transition relation of $\mathcal{T}$ is *chosen* from the complete transition space, certain transitions of automata from $\mathcal{S}$ may not be present (and thus neither omnipresent) in $\mathcal{T}$. We now study the types of synchronized automata in which not too many transitions from the complete transition space have been left out, i.e. in which transitions are (omni)present.

In the *maximal-si* synchronized automaton $\mathcal{T}$ over $\mathcal{S}$, all executions of an action a by definition involve all automata in which a is enabled at the current local state. Hence it is not surprising that all transitions of (sub)automata from $\mathcal{S}$ are omnipresent — and thus present — in $\mathcal{T}$.

Theorem 4.6.3. *Let $a \in \Sigma$.*

if $\delta_a = \mathcal{R}_a^{si}(\mathcal{S})$, then each a-transition of SUB_J as well as each a-transition of $\mathcal{A}_j$ is omnipresent in $\mathcal{T}$.

Proof. We only prove the statement for SUB_J, as the other case is analogous. Let $\delta_a = \mathcal{R}_a^{si}(\mathcal{S})$ and let $(p, a, p') \in \delta_J$. Now let $q \in Q$ be such that $\mathrm{proj}_J(q) = p$ and let $q' \in Q$ be the state that is defined by $\mathrm{proj}_J(q') = p'$ and, for all $i \in \mathcal{I} \setminus J$, $\mathrm{proj}_i(q')$ is such that $(\mathrm{proj}_i(q), a, \mathrm{proj}_i(q')) \in \delta_i$ whenever $a \,\mathrm{en}_{\mathcal{A}_i}\, \mathrm{proj}_i(q)$. Then by Definitions 4.1.1 and 4.5.2(4), $(q, a, q') \in \mathcal{R}_a^{si}(\mathcal{S})$. Hence (p, a, p') is omnipresent in $\mathcal{T}$. □

It is clear that once a transition of an automaton is present or omnipresent in a synchronized automaton, adding more transitions to the latter will not affect that property. We may thus conclude from Theorem 4.6.3 that whenever $\mathcal{T}$ is such that $\delta_a = \mathcal{R}_a^{no}(\mathcal{S})$, for all $a \in \Sigma_{ext}$, then all transitions of the automata from $\mathcal{S}$ are omnipresent — and thus present — in $\mathcal{T}$. Moreover, if $\delta_a = \mathcal{R}_a^{no}(\mathcal{S})$, for all $a \in \Sigma_{ext}$, then for every transition (p, a, p') of SUB_J, we have that $(q, a, q') \in \mathcal{R}_a^{no}(\mathcal{S})$ for all $q \in Q$ such that $\mathrm{proj}_J(q) = p$, $\mathrm{proj}_J(q') = p'$, and for all $i \in \mathcal{I} \setminus J$, $\mathrm{proj}_i(q) = \mathrm{proj}_i(q')$.

Theorem 4.6.4. *Let $a \in \Sigma$. Then*

if $\delta_a = \mathcal{R}_a^{no}(\mathcal{S})$, then each a-transition of SUB_J as well as each a-transition of $\mathcal{A}_j$ is omnipresent in $\mathcal{T}$. □

In the following example we demonstrate that in the *maximal-free* (*maximal-ai*) synchronized automaton over $\mathcal{S}$, not all transitions of all automata from $\mathcal{S}$ need to be present — let alone omnipresent. Apparently the *is-free* (*is-ai*) predicate may contain too few transitions from the complete transition space.

Example 4.6.5. Consider automata $\mathcal{A}_1 = (\{p\}, \{a\}, \{(p, a, p)\}, \{p\})$, $\mathcal{A}_2 = (\{q, q'\}, \{a\}, \{(q, a, q), (q, a, q'), (q', a, q')\}, \{q\})$, and $\mathcal{A}_3 = (\{r\}, \{a\}, \varnothing, \{r\})$. They are depicted in Figure 4.11.

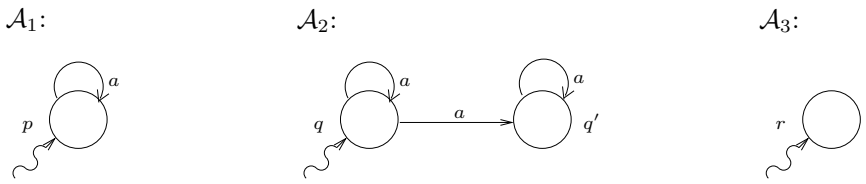

Fig. 4.11. Automata $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$.

It is not difficult to see that both the $\mathcal{R}^{free}$-synchronized automaton $\mathcal{T}_{1,2}^{free}$ over $\{\mathcal{A}_1, \mathcal{A}_2\}$ and the $\mathcal{R}^{ai}$-synchronized automaton $\mathcal{T}_{2,3}^{ai}$ over $\{\mathcal{A}_2, \mathcal{A}_3\}$ have an empty transition relation. We thus see that none of the a-transitions appearing in $\mathcal{A}_2$ is present — and thus neither omnipresent — in either $\mathcal{T}_{1,2}^{free}$ or $\mathcal{T}_{2,3}^{ai}$. □

By looking more closely at Example 4.6.5 we obtain some hints as to why some transitions of automata from $\mathcal{S}$ cannot be omnipresent in the *maximal-free* (*maximal-ai*) synchronized automaton over $\mathcal{S}$.

First consider the case that $\mathcal{T}$ is the *maximal-ai* synchronized automaton over $\mathcal{S}$. From Example 4.6.5 it follows immediately that no a-transition of $\mathcal{A}_j$ will be present in $\mathcal{T}$ if $\delta_a = \varnothing$. On the other hand, if $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S}) \neq \varnothing$, then every a-transition of $\mathcal{A}_j$ can be executed in $\mathcal{T}$ from every state in which a is enabled at the local states of all other automata that also have a as an action.

Theorem 4.6.6. *For all $a \in \Theta \cap \Sigma_j$, let $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S})$. Then*

$\mathcal{R}_a^{ai}(\mathcal{S}) \neq \varnothing$ if and only if $\delta_{j,a} \neq \varnothing$ and each a-transition of $\mathcal{A}_j$ is present in $\mathcal{T}$.

Proof. (If) Trivial.

(Only if) Let $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S}) \neq \varnothing$. Then for all $i \in \mathcal{I}$, if $a \in \Sigma_i$, then there exist q_i, q_i' such that $(q_i, a, q_i') \in \delta_i$. Now let $(p, a, p') \in \delta_j$ and let $q, q' \in Q$ be such that $\text{proj}_j(q) = p$ and $\text{proj}_j(q') = p'$, $\text{proj}_i(q) = q_i$ and $\text{proj}_i(q') = q_i'$, for all $i \in \mathcal{I}$ such that $a \in \Sigma_i$ and $i \neq j$, and $\text{proj}_k(q) = \text{proj}_k(q')$, for all $k \in \mathcal{I}$ such that $a \notin \Sigma_k$. This implies that $(q, a, q') \in \mathcal{R}_a^{ai}(\mathcal{S})$ and hence (p, a, p') is present in $\mathcal{T}$. □

Example 4.6.5 suggests furthermore that certain transitions of automata from $\mathcal{S}$ cannot be omnipresent in $\mathcal{T}$ in case the following situation exists. Let q be a state of $\mathcal{T}$ at which an action a is locally enabled — due to the existence of an a-transition t — in (at least) one of the automata from $\mathcal{S}$, while it is not locally enabled — due to the absence of an a-transition — in (at least) one other automaton from $\mathcal{S}$ *that does have* a *in its alphabet*. If this is the case, then a is not enabled at q in $\mathcal{T}$. The reason is that otherwise action a could be executed from q *without* the participation of all of the automata having this a as one of their actions, which would be contradicting the fact that $\mathcal{T}$ is the *maximal-ai* synchronized automaton over $\mathcal{S}$. Hence the a-transition t cannot be omnipresent in $\mathcal{T}$.

To avoid the situation sketched above from occurring when dealing with *maximal-ai* synchronized automata, we define a *Θ-enabling set of automata* as a set of automata with the property that each of its constituting automata is Θ-enabling. Recall Θ to be an arbitrary alphabet disjoint from Q.

Definition 4.6.7. *$\mathcal{S}$ is Θ-enabling if for all $i \in \mathcal{I}$, $\mathcal{A}_i$ is Θ-enabling.* □

If $\mathcal{S}$ is Σ-enabling, then we may also simply say that $\mathcal{S}$ is enabling. Note, however, that in that case the *maximal-ai* synchronized automaton over $\mathcal{S}$ and the *maximal-si* synchronized automaton over $\mathcal{S}$ are one and the same. In fact, if $\mathcal{S}$ is $\{a\}$-enabling and $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S})$, for an action a, then clearly $\delta_a = \mathcal{R}_a^{si}(\mathcal{S})$.

Theorem 4.6.8. *For all $a \in \Theta \cap \Sigma$, let $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S})$. Then*

> *if $\mathcal{S}$ is Θ-enabling, then for all $a \in \Theta$, each a-transition of SUB_J as well as each a-transition of $\mathcal{A}_j$ is omnipresent in $\mathcal{T}$.*

Proof. Let $\mathcal{S}$ be Θ-enabling. Together with the fact that $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma$, this implies that $\delta_a = \mathcal{R}_a^{si}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma$, after which the result follows directly from Theorem 4.6.3. □

We conclude that whenever $\mathcal{S}$ is enabling, all transitions of (sub)automata from $\mathcal{S}$ are omnipresent — and thus present — in the *maximal-ai* synchronized automaton $\mathcal{T}$ over $\mathcal{S}$.

Now consider the case that $\mathcal{T}$ is the *maximal-free* synchronized automaton over $\mathcal{S}$. Consequently, Example 4.6.5 suggests that certain transitions of automata from $\mathcal{S}$ cannot be omnipresent in $\mathcal{T}$ in case the following situation exists. Let q be a state of $\mathcal{T}$ at which an action a is locally enabled in (at least) two of the automata from $\mathcal{S}$, of which (at least) one time as a loop. Then the other a-transition that is locally enabled at q cannot be omnipresent in $\mathcal{T}$. The reason is that by our maximal interpretation the automaton with the loop on a participates in the execution of any a-transition in $\mathcal{T}$ from q. This would be contradicting the fact that $\mathcal{T}$ is the *maximal-free* synchronized automaton over $\mathcal{S}$.

To avoid the situation sketched above from occurring when studying *maximal-free* synchronized automata, we define a *Θ-J-loop-limited set of automata* as a set of automata with the property that whenever there is an a-transition, with $a \in \Theta$, in the *maximal-free* team automaton over $\mathcal{A}_k$, $k \in J$, then none of the other automata in the set has a loop on a.

Definition 4.6.9. *(1) $\mathcal{S}$ is* Θ-J-loop limited *if for all $a \in \Theta \cap \Sigma_J$, whenever there exists an $i \in \mathcal{I} \setminus J$ such that $(q,q) \in \delta_{i,a}$ for some $q \in Q_i$, then $\mathcal{R}_a^{free}(\{\mathcal{A}_k \mid k \in J\}) = \varnothing$, and*

(2) $\mathcal{S}$ is Θ-j-loop limited *if for all $a \in \Theta \cap \Sigma_j$, whenever there exists an $i \in \mathcal{I} \setminus \{j\}$ such that $(q,q) \in \delta_{i,a}$ for some $q \in Q_i$, then $\delta_{j,a} = \varnothing$.* □

We thus note that $\mathcal{S}$ being Θ-j-loop limited is the same as $\mathcal{S}$ being Θ-$\{j\}$-loop limited. If $\mathcal{S}$ is Σ_J-J-loop limited or Σ_j-j-loop limited, then we may also simply say that $\mathcal{S}$ is J-loop limited or j-loop limited, respectively. Finally, note that whenever $\Theta \subseteq \Sigma_J \setminus (\bigcup_{i \in \mathcal{I} \setminus J} \Sigma_i)$ or $\Theta \subseteq \Sigma_j \setminus (\bigcup_{i \in \mathcal{I} \setminus \{j\}} \Sigma_i)$, then $\mathcal{S}$ is Θ-J-loop limited or Θ-j-loop limited, respectively.

Loop limitedness is a sufficient *and* necessary condition on $\mathcal{S}$ for guaranteeing all transitions of (sub)automata from $\mathcal{S}$ to be omnipresent — and thus present — in the *maximal-free* synchronized automaton $\mathcal{T}$ over $\mathcal{S}$.

Theorem 4.6.10. *For all $a \in \Theta \cap \Sigma$, let $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$. Then*

(1) each a-transition of SUB_J, for all $a \in \Theta$, is omnipresent in $\mathcal{T}$ if and only if $\mathcal{S}$ is Θ-J-loop limited, and

(2) each a-transition of $\mathcal{A}_j$, for all $a \in \Theta$, is omnipresent in $\mathcal{T}$ if and only if $\mathcal{S}$ is Θ-j-loop limited.

Proof. (1) (If) Let $\mathcal{S}$ be Θ-J-loop limited, let $a \in \Theta$, and let $(p, a, p') \in \delta_J$. Now let $q \in Q$ be such that $\mathrm{proj}_J(q) = p$ and let $q' \in Q$ be the state that is defined by $\mathrm{proj}_J(q') = p'$ and, for all $i \in \mathcal{I} \setminus J$, $\mathrm{proj}_i(q') = \mathrm{proj}_i(q)$.

Then Definitions 4.1.1 and 4.5.2(2) together with the fact that $\mathcal{S}$ is Θ-J-loop limited imply that $(q, a, q') \in \mathcal{R}_a^{free}(\mathcal{S})$. Hence (p, a, p') is omnipresent in $\mathcal{T}$.

(Only if) Let each a-transition of SUB_J, for all $a \in \Theta$, be omnipresent in $\mathcal{T}$. Now assume that $\mathcal{S}$ is not Θ-J-loop limited. Then there exist an $a \in \Theta$, a $(p, a, p') \in \mathcal{R}_a^{free}(\{\mathcal{A}_k \mid k \in J\})$, and an $i \in \mathcal{I} \setminus J$ such that $(q, a, q) \in \delta_i$. Now let $r \in Q$ be such that $\mathrm{proj}_J(r) = p$ and $\mathrm{proj}_i(r) = q$. Since (p, a, p') is omnipresent in $\mathcal{T}$, there exists an $(r, a, r') \in \delta$ such that $\mathrm{proj}_J(r') = p'$. Moreover, because $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$ it must be the case that for all $\ell \in \mathcal{I} \setminus J$, $\mathrm{proj}_\ell(r') = \mathrm{proj}_\ell(r)$ and $(\mathrm{proj}_\ell(r), a, \mathrm{proj}_\ell(r')) \notin \delta_\ell$, which contradicts the fact that $(q, a, q) \in \delta_i$. Hence $\mathcal{S}$ is Θ-J-loop limited.

(2) Analogous. □

This concludes our intermezzo on the presence and omnipresence of transitions of (sub)automata in synchronized automata over these (sub)automata. In the next two subsections we investigate the inheritance of the automata-theoretic properties introduced in Section 3.2 from synchronized automata to their (sub)automata, and vice versa. While doing so we adhere to the order according to which these properties were introduced.

4.6.1 Top-Down Inheritance of Properties

Initially we search for sufficient conditions under which the automata-theoretic properties of Section 3.2 are inherited from synchronized automata to their (sub)automata.

Reduced Versions

In order to investigate the conditions under which action reducedness, transition reducedness, and state reducedness are inherited from a synchronized automaton to its (sub)automata, it is important to know whether or not the projection on a (sub)automaton of a state that is reachable in a synchronized automaton is itself reachable in that (sub)automaton.

Lemma 4.6.11. *Let $q \in Q$ be reachable in $\mathcal{T}$. Then*

(1) $proj_J(q)$ is reachable in SUB_J and

(2) $proj_j(q)$ is reachable in $\mathcal{A}_j$.

Proof. If $q \in Q$ is reachable in $\mathcal{T}$, then there exists a computation $\alpha q \in \mathbf{C}_{\mathcal{T}}$. Hence (1) and (2) follow directly from Lemma 4.2.6 and its Corollary 4.2.7, respectively. □

An immediate consequence of Lemma 4.6.11 is that the state reducedness of a synchronized automaton is inherited by all its (sub)automata.

Theorem 4.6.12. *Let $\mathcal{T}$ be state reduced. Then*

SUB_J as well as $\mathcal{A}_j$ is state reduced. □

Note that the statements of Lemma 4.6.11 cannot be reversed. This follows from Example 4.2.8. This also means that the Θ-action-reduced (Θ-transition-reduced) versions of the subautomata of a synchronized automaton in general are different from the subautomata of the Θ-action-reduced (Θ-transition-reduced) versions of that synchronized automaton. Hence in general $SUB_J(\mathcal{T}_A^\Theta) \neq (SUB_J(\mathcal{T}))_A^\Theta$ and $SUB_J(\mathcal{T}_T^\Theta) \neq (SUB_J(\mathcal{T}))_T^\Theta$, even if $\Theta \subseteq \Sigma_J$. The situation is different in case of state reducedness. In fact, since the state-reduced version $\mathcal{T}_S$ of a synchronized automaton $\mathcal{T}$ over $\mathcal{S}$ need not be a synchronized automaton over $\mathcal{S}$, subautomata of $\mathcal{T}_S$ are not defined unless $\mathcal{T}_S = \mathcal{T}$, i.e. $\mathcal{T}$ is state reduced. However, if $\mathcal{T}$ is state reduced, then Theorem 4.6.12 implies that $SUB_J(\mathcal{T}) = (SUB_J(\mathcal{T}))_S$.

In the following example we show that the fact that $\mathcal{T}$ is Θ-action reduced (Θ-transition reduced) in general does not imply that each of its constituting automata is Θ-action reduced (Θ-transition reduced). To construct a Θ-action-reduced synchronized automaton $\mathcal{T}$ over $\mathcal{S}$, it suffices to have just one Θ-action-reduced automaton in $\mathcal{S}$. By basing the transition relation of $\mathcal{T}$ solely on that Θ-action-reduced automaton, e.g., one obtains that $\mathcal{T}$ is Θ-action reduced while obviously not all automata from $\mathcal{S}$ need to be Θ-action reduced. It is even easier to construct a Θ-transition-reduced synchronized automaton $\mathcal{T}$ over $\mathcal{S}$, viz. by equipping $\mathcal{T}$ with only useful a-transitions, for all $a \in \Theta$.

Example 4.6.13. Consider automata $\mathcal{A}_1 = (\{q_1, q_1'\}, \{a\}, \{(q_1, a, q_1')\}, \{q_1\})$ and $\mathcal{A}_2 = (\{q_2, q_2'\}, \{a\}, \{(q_2', a, q_2)\}, \{q_2\})$, as depicted in Figure 4.12(a).

Consider the synchronized automaton $\mathcal{T} = (Q, \{a\}, \delta, \{(q_1, q_2)\})$, with $Q = \{(q_1, q_2), (q_1, q_2'), (q_1', q_2), (q_1', q_2')\}$ and $\delta = \{((q_1, q_2), a, (q_1', q_2))\}$, over $\{\mathcal{A}_1, \mathcal{A}_2\}$. It is depicted in Figure 4.12(b).

It is easy to see that $\mathcal{T}$ is both action reduced and transition reduced, whereas $\mathcal{A}_2$ clearly is neither action reduced nor transition reduced. □

The action reducedness of a synchronized automaton *is* inherited by each of its (sub)automata in case each of the latter's actions is *ai* in the synchronized automaton.

Theorem 4.6.14. *Let $\mathcal{T}$ be Θ-action reduced. Then*

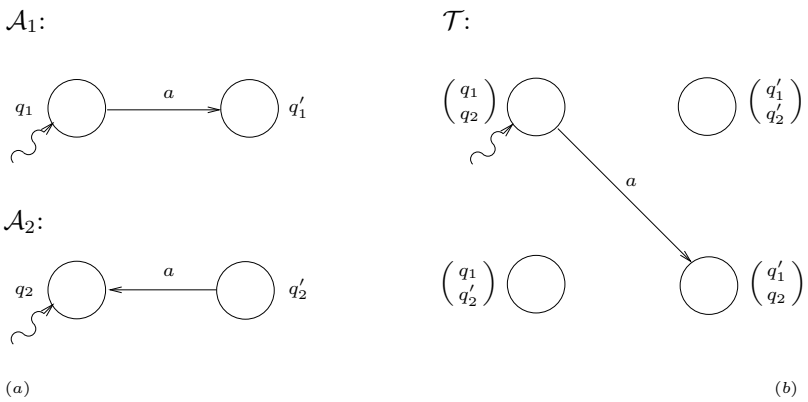

Fig. 4.12. Automata $\mathcal{A}_1$ and $\mathcal{A}_2$, and synchronized automaton $\mathcal{T}$.

(1) if $\delta_a \subseteq \mathcal{R}_a^{ai}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_J$, then SUB_J is Θ-action reduced, and

(2) if $\delta_a \subseteq \mathcal{R}_a^{ai}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_j$, then $\mathcal{A}_j$ is Θ-action reduced.

Proof. (1) Let $a \in \Theta \cap \Sigma_J$ and let $\delta_a \subseteq \mathcal{R}_a^{ai}(\mathcal{S})$. Since $\mathcal{T}$ is Θ-action reduced we know that there exists a computation $\alpha \in \mathbf{C}_{\mathcal{T}}$ such that $\alpha = \beta aq$ for some $\beta \in I(\Sigma Q)^*$ and $q \in Q$. Since $\delta_a \subseteq \mathcal{R}_a^{ai}(\mathcal{S})$, $\pi_{SUB_J}(\alpha) = \pi_{SUB_J}(\beta)a\mathrm{proj}_J(q) \in \mathbf{C}_{SUB_J}$ by Definition 4.2.3(1) and Lemma 4.2.6. Hence a is active in SUB_J and SUB_J is thus Θ-action reduced.

(2) Analogous, but now using Definition 4.2.3(3) and Corollary 4.2.7. □

It is worthwhile to notice that the requirement of every action being *ai* as condition in this theorem cannot be replaced by requiring each action to be *free* or *si* without invalidating the statement. In the following example we show this by demonstrating that the action reducedness of $\mathcal{T}$ in general is not inherited by each of its (sub)automata in case $\mathcal{T}$ is the *maximal-free* synchronized automaton nor in case $\mathcal{T}$ is the *maximal-si* synchronized automaton — and hence neither in case $\mathcal{T}$ is a synchronized automaton in which every action is *free* nor in case $\mathcal{T}$ is a synchronized automaton in which every action is *si*.

Example 4.6.15. (Example 4.6.13 continued) First we consider the $\mathcal{R}^{free}$-synchronized automaton $\mathcal{T}^{free} = (Q, \{a\}, \delta^{free}, \{(q_1, q_2)\})$, with $\delta^{free} = \delta \cup \{((q_1, q_2'), a, (q_1, q_2)), ((q_1, q_2'), a, (q_1', q_2')), ((q_1', q_2'), a, (q_1', q_2))\}$, over $\{\mathcal{A}_1, \mathcal{A}_2\}$. It is depicted in Figure 4.13(a).

Clearly, $\mathcal{T}^{free}$ is $\{a\}$-action reduced. Now note that $SUB_{\{2\}}(\mathcal{T}^{free})$ is essentially a copy of $\mathcal{A}_2$. It is easy to see that neither $SUB_{\{2\}}(\mathcal{T}^{free})$ nor $\mathcal{A}_2$ is $\{a\}$-action reduced.

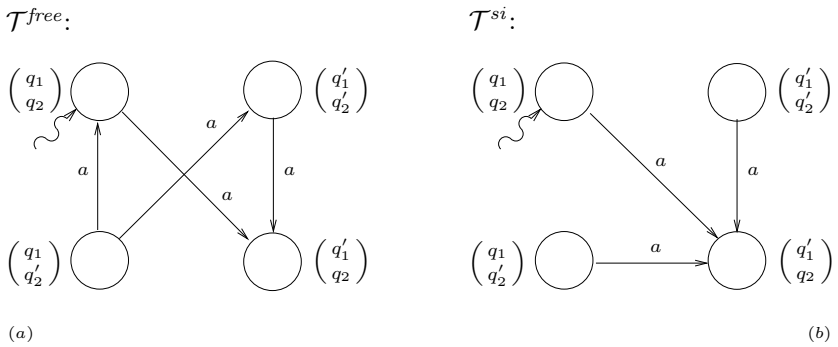

Fig. 4.13. Synchronized automata $\mathcal{T}^{free}$ and $\mathcal{T}^{si}$.

Next we consider the $\mathcal{R}^{si}$-synchronized automaton $\mathcal{T}^{si} = (Q, \{a\}, \delta^{si}, \{(q_1, q_2)\})$, with $\delta^{si} = \delta \cup \{((q_1, q_2'), a, (q_1', q_2)), ((q_1', q_2'), a, (q_1', q_2))\}$, over $\{\mathcal{A}_1, \mathcal{A}_2\}$. It is depicted in Figure 4.13(b).

Clearly, $\mathcal{T}^{si}$ is $\{a\}$-action reduced. Moreover, also $SUB_{\{2\}}(\mathcal{T}^{si})$ is essentially a copy of $\mathcal{A}_2$. Since we know that $\mathcal{A}_2$ is not $\{a\}$-action reduced, neither is $SUB_{\{2\}}(\mathcal{T}^{si})$.

Finally, we note that the $\mathcal{R}^{ai}$-synchronized automaton $\mathcal{T}^{ai} = (Q, \{a\}, \{((q_1, q_2'), a, (q_1', q_2))\}, \{(q_1, q_2)\})$ over $\{\mathcal{A}_1, \mathcal{A}_2\}$ is not $\{a\}$-action reduced. □

In Example 4.6.13 we have seen that the fact that $\mathcal{T}$ is transition reduced in general does not imply that $\mathcal{A}_j$ is transition reduced. As we show next, the transition reducedness of a synchronized automaton *is* inherited by each of its (sub)automata in case each of the latter's transitions is present in the synchronized automaton.

Theorem 4.6.16. *Let $\mathcal{T}$ be Θ-transition reduced. Then*

(1) SUB_J is Θ-transition reduced and

(2) if each a-transition of $\mathcal{A}_j$, for all $a \in \Theta$, is present in $\mathcal{T}$, then $\mathcal{A}_j$ is Θ-transition reduced.

Proof. (1) Let $a \in \Theta \cap \Sigma_J$ and let $(p, a, p') \in \delta_J$. Then Theorem 4.6.2 implies that there exists a transition $(q, a, q') \in \delta$ such that $(\mathrm{proj}_J(q), a, \mathrm{proj}_J(q')) = (p, a, p')$. Since $\mathcal{T}$ is Θ-transition reduced there furthermore exists a computation $\alpha q \in \mathbf{C}_{\mathcal{T}}$, i.e. q is reachable in $\mathcal{T}$. Lemma 4.6.11(1) now implies that p is reachable in SUB_J and thus (p, a, p') is useful in SUB_J. Hence SUB_J is Θ-transition reduced.

(2) Analogous. □

Together with Theorems 4.6.3, 4.6.4, 4.6.6, and 4.6.10(2) this implies the following result.

Corollary 4.6.17. *Let $\mathcal{T}$ be Θ-transition reduced and let syn $\in \{si, no\}$. Then*

(1) if $\delta_a = \mathcal{R}_a^{syn}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_j$, then $\mathcal{A}_j$ is Θ-transition reduced,

(2) if $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S}) \neq \varnothing$, for all $a \in \Theta \cap \Sigma_j$, then $\mathcal{A}_j$ is Θ-transition reduced, and

(3) if $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_j$, and $\mathcal{S}$ is Θ-j-loop limited, then $\mathcal{A}_j$ is Θ-transition reduced. □

Enabling

We now turn to the inheritance of enabling from synchronized automata to their (sub)automata. In the following example we show that when a synchronized automaton $\mathcal{T}$ over $\mathcal{S}$ is Θ-enabling, then this in general does not imply that each of its (sub)automata is Θ-enabling. We show this by using the fact that a necessary condition for a synchronized automaton to be $\{a\}$-enabling, for an action a, is that in each of its states (at least) one of its constituting automata enables a. However, it is not guaranteed that each of the synchronized automaton's (sub)automata enables a in each of its states.

Example 4.6.18. (Example 4.2.1 continued) Clearly $\mathcal{T}$ is action reduced and state reduced (and thus transition reduced). It is moreover enabling. However, we immediately see that $\mathcal{A}_1$ and $\mathcal{A}_3$ are not. It is also easy to see that $SUB_{\{3\}}$, which is essentially a copy of $\mathcal{A}_3$, is not enabling. □

Note that this example allows us to conclude that also the Θ-enabling of a Θ-action-reduced (Θ-transition-reduced, state-reduced) synchronized automaton in general is not inherited by its (sub)automata.

The enabling of a synchronized automaton *is* inherited by each of its (sub)automata in case every action of the synchronized automaton is *ai*.

Theorem 4.6.19. *Let $\mathcal{T}$ be Θ-enabling. Then*

(1) if $\delta_a \subseteq \mathcal{R}_a^{ai}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_J$, then SUB_J is Θ-enabling, and

(2) if $\delta_a \subseteq \mathcal{R}_a^{ai}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_j$, then $\mathcal{A}_j$ is Θ-enabling.

Proof. (1) Let $a \in \Theta \cap \Sigma_J$ and let $\delta_a \subseteq \mathcal{R}_a^{ai}(\mathcal{S})$. Let $p \in Q_J$. Now let $q \in Q$ be such that $\mathrm{proj}_J(q) = p$. Since $\mathcal{T}$ is Θ-enabling we know that a en$_\mathcal{T}$ q. Hence there exists a $q' \in Q$ such that $(q, q') \in \delta_a$. Moreover, $\mathrm{proj}_J^{[2]}(q, q') \in (\delta_J)_a$ because $a \in \Sigma_J$ and $\delta_a \subseteq \mathcal{R}_a^{ai}(\mathcal{S})$. Consequently, a en$_{SUB_J}$ p. Hence SUB_J is Θ-enabling.

(2) Analogous. □

It is worthwhile to notice that the requirement of every action being *ai* as condition in this theorem cannot be replaced by requiring each action to be *free* or *si* without invalidating the statement. In the following example we show this by demonstrating that the enabling of $\mathcal{T}$ in general is not inherited by each of its (sub)automata in case $\mathcal{T}$ is the *maximal-free* synchronized automaton nor in case $\mathcal{T}$ is the *maximal-si* synchronized automaton — and hence neither in case $\mathcal{T}$ is a synchronized automaton in which every action is *free* nor in case $\mathcal{T}$ is a synchronized automaton in which every action is *si*.

Example 4.6.20. Let $\mathcal{A}_1 = (\{q_1, q_1'\}, \{a\}, \{(q_1, a, q_1'), (q_1', a, q_1)\}, \{q_1\})$ and let $\mathcal{A}_2 = (\{q_2, q_2'\}, \{a\}, \{(q_2, a, q_2')\}, \{q_2\})$. These automata are depicted in Figure 4.14.

Fig. 4.14. Automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

In Figure 4.15(a) we have depicted the $\mathcal{R}^{free}$-synchronized automaton $\mathcal{T}^{free} = (Q, \{a\}, \delta^{free}, \{(q_1, q_2)\})$, with $Q = \{(q_1, q_2), (q_1, q_2'), (q_1', q_2), (q_1', q_2')\}$ and δ^{free} as depicted, over $\{\mathcal{A}_1, \mathcal{A}_2\}$.

It is easy to see that $\mathcal{T}^{free}$ is enabling. Now note that $SUB_{\{2\}}(\mathcal{T}^{free})$ is essentially a copy of $\mathcal{A}_2$. Clearly neither $SUB_{\{2\}}(\mathcal{T}^{free})$ nor $\mathcal{A}_2$ is enabling.

Consequently, in Figure 4.15(b) we have depicted the $\mathcal{R}^{si}$-synchronized automaton $\mathcal{T}^{si} = (Q, \{a\}, \delta^{si}, \{(q_1, q_2)\})$, with δ^{si} as depicted, over $\{\mathcal{A}_1, \mathcal{A}_2\}$.

It is again easy to see that $\mathcal{T}^{si}$ is enabling. Clearly also $SUB_{\{2\}}(\mathcal{T}^{si})$ is essentially a copy of $\mathcal{A}_2$. Since $\mathcal{A}_2$ is not enabling, neither is $SUB_{\{2\}}(\mathcal{T}^{si})$.

Finally, we note that the $\mathcal{R}^{ai}$-synchronized automaton $\mathcal{T}^{ai} = (Q, \{a\}, \{((q_1, q_2), a, (q_1', q_2')), ((q_1', q_2), a, (q_1, q_2'))\}, \{(q_1, q_2)\})$ over $\{\mathcal{A}_1, \mathcal{A}_2\}$ is not enabling. □

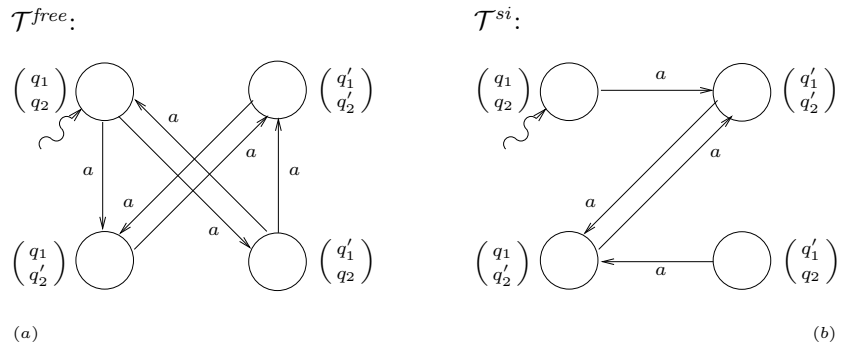

Fig. 4.15. Synchronized automata $\mathcal{T}^{free}$ and $\mathcal{T}^{si}$.

Determinism

We now conclude this subsection by turning to the inheritance of determinism. We begin by showing that when a synchronized automaton $\mathcal{T}$ over $\mathcal{S}$ is Θ-deterministic, then this in general does not imply that each of its (sub)automata is Θ-deterministic. In case of inheritance from a synchronized automaton to its constituting automata, this can be concluded directly from Example 4.6.5. In case of inheritance from a synchronized automaton to its subautomata, this can be concluded from the following example. This example uses the fact that the states of $\mathcal{A}_j$ can be used to distinguish states of a synchronized automaton $\mathcal{T}$ that without the j-th component cannot be distinguished.

Example 4.6.21. Consider automata $\mathcal{A}_1 = (\{p, p'\}, \{a\}, \{(p, a, p'), (p', a, p)\}, \{p\})$, $\mathcal{A}_2 = (\{q, q'\}, \{a\}, \{(q, a, q'), (q', a, q)\}, \{q\})$, and $\mathcal{A}_3 = (\{r, r'\}, \{a\}, \{(r, a, r'), (r', a, r)\}, \{r\})$, as depicted in Figure 4.16.

In Figure 4.17(a) we have depicted the synchronized automaton $\mathcal{T} = (Q, \{a\}, \delta, \{(p, q, r)\})$, with $Q = \{(p, q, r), (p', q, r), (p, q', r), (p', q', r), (p, q, r'), (p', q, r'), (p, q', r'), (p', q', r')\}$ and δ as depicted, over $\{\mathcal{A}_i \mid i \in [3]\}$.

It is easy to see that $\mathcal{T}$ is action reduced and state reduced (and thus transition reduced). Furthermore, $\mathcal{T}$ clearly is deterministic.

Consequently, in Figure 4.17(b) we have depicted its subautomaton $SUB_{\{1,2\}} = (\{(p, q), (p, q'), (p', q), (p', q')\}, \{a\}, \delta_{\{1,2\}}, \{(p, q)\})$, with $\delta_{\{1,2\}}$ as depicted.

Clearly $SUB_{\{1,2\}}$ is not deterministic as, e.g., $((p', q), a, (p, q)) \in \delta_{\{1,2\}}$ and $((p', q), a, (p, q')) \in \delta_{\{1,2\}}$. □

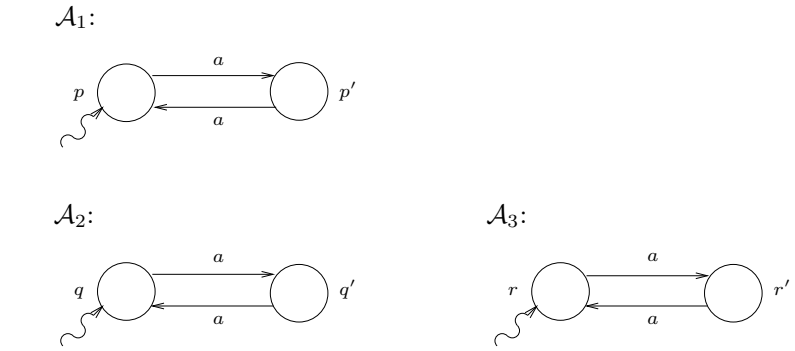

Fig. 4.16. Automata $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$.

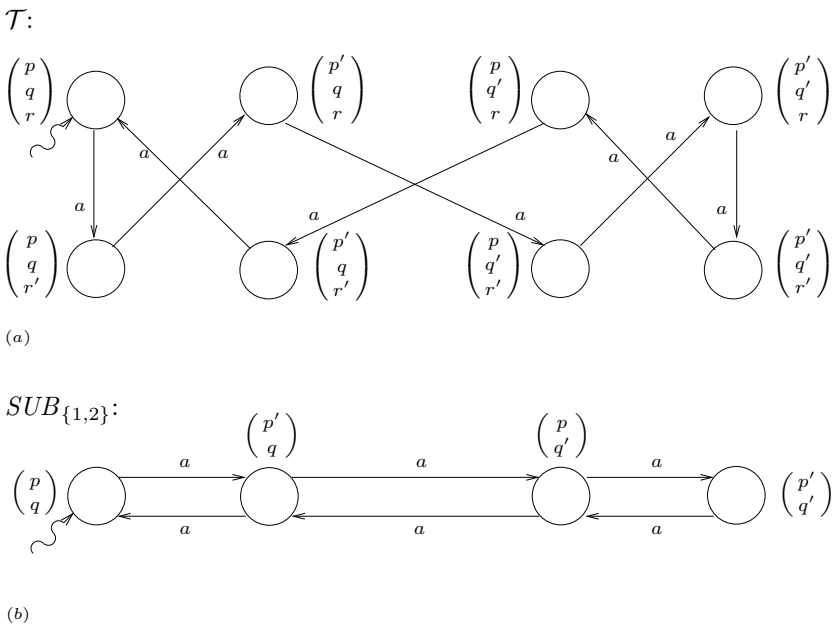

Fig. 4.17. Synchronized automaton $\mathcal{T}$ and its subautomaton $SUB_{\{1,2\}}$.

The determinism of a *maximal-free* (*maximal-ai*, *maximal-si*) synchronized automaton *is* inherited by each of its (sub)automata in case each of the latter's transitions is present in the synchronized automaton.

Theorem 4.6.22. *Let $\mathcal{T}$ be Θ-deterministic and let $syn \in \{no, free, ai, si\}$. Then*

(1) if $\delta_a = \mathcal{R}_a^{syn}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_J$, then SUB_J is Θ-deterministic, and

(2) if $\delta_a = \mathcal{R}_a^{syn}(\mathcal{S})$ *and each* a*-transition of* $\mathcal{A}_j$ *is present in* $\mathcal{T}$*, for all* $a \in \Theta \cap \Sigma_j$*, then* $\mathcal{A}_j$ *is* Θ*-deterministic.*

Proof. (1) Let $a \in \Theta \cap \Sigma_J$ and let $\delta_a = \mathcal{R}_a^{syn}(\mathcal{S})$. Since $\mathcal{T}$ is Θ-deterministic we know that $I = \{q_0\}$, for some $q_0 \in Q$. Hence, trivially, $I_J = \{\mathrm{proj}_J(q_0)\}$. It thus remains to prove that for all $q \in Q_J$, there exists at most one $q' \in Q_J$ such that $(q, a, q') \in \delta_J$.

Now assume that there exists a $p \in Q_J$ such that $(p, a, p') \in \delta_J$ and $(p, a, p'') \in \delta_J$, with $p' \neq p''$. Then Theorem 4.6.2 implies that there exist a $(q, a, q') \in \delta$ such that $(\mathrm{proj}_J(q), a, \mathrm{proj}_J(q')) = (p, a, p')$ and an $(r, a, r') \in \delta$ such that $(\mathrm{proj}_J(r), a, \mathrm{proj}_J(r')) = (p, a, p'')$. Moreover, since $q' \neq r'$ and $\mathcal{T}$ is Θ-deterministic, we know that $q \neq r$. Consequently, the fact that $\delta_a = \mathcal{R}_a^{syn}(\mathcal{S})$ implies that we can replace the components from J in (q, q') by those in (r, r') and still have a transition in $\mathcal{R}_a^{syn}(\mathcal{S})$. Hence there exists a $q'' \in Q$ such that $(q, a, q'') \in \delta$ with $\mathrm{proj}_{\mathcal{I}\setminus J}(q'') = \mathrm{proj}_{\mathcal{I}\setminus J}(q')$ and $\mathrm{proj}_J(q'') = p'' = \mathrm{proj}_J(r')$. Since $p' \neq p''$ this means that $\mathcal{T}$ is not Θ-deterministic, a contradiction. Hence SUB_J is Θ-deterministic.

(2) Analogous. □

Together with Theorems 4.6.3, 4.6.4, 4.6.6, and 4.6.10(2) this implies the following result.

Corollary 4.6.23. *Let* $\mathcal{T}$ *be* Θ*-deterministic and let* $syn \in \{si, no\}$*. Then*

(1) if $\delta_a = \mathcal{R}_a^{syn}(\mathcal{S})$*, for all* $a \in \Theta \cap \Sigma_j$*, then* $\mathcal{A}_j$ *is* Θ*-deterministic,*

(2) if $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S}) \neq \varnothing$*, for all* $a \in \Theta \cap \Sigma_j$*, then* $\mathcal{A}_j$ *is* Θ*-deterministic, and*

(3) if $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$*, for all* $a \in \Theta \cap \Sigma_j$*, and* $\mathcal{S}$ *is* Θ*-*j*-loop limited, then* $\mathcal{A}_j$ *is* Θ*-deterministic.* □

4.6.2 Bottom-Up Inheritance of Properties

Dual to the above investigations we now change focus and study sufficient conditions under which the automata-theoretic properties of Section 3.2 are preserved from automata to synchronized automata.

We recall from Section 4.3 that $\mathcal{T}$ is a synchronized automaton over $\mathcal{S}'$ — upto a reordering — whenever $\mathcal{S}' = \{SUB_{\mathcal{I}_j} \mid \{\mathcal{I}_j \mid j \in \mathcal{J}\}$ forms a partition of $\mathcal{I}\}$. Hence it suffices to investigate the conditions under which a property that holds for (elements of) a set of automata is preserved by a synchronized automaton over that set of automata. Therefore, we extend Definition 4.6.7 by defining when a set of automata is Θ-action reduced (Θ-transition reduced, state reduced, Θ-deterministic).

Definition 4.6.24. *$\mathcal{S}$ is Θ-action reduced (Θ-transition reduced, state reduced, Θ-deterministic) if for all $i \in \mathcal{I}$, $\mathcal{A}_i$ is Θ-action reduced (Θ-transition reduced, state reduced, Θ-deterministic).* □

If $\mathcal{S}$ is Σ-action reduced (Σ-transition reduced, Σ-deterministic) we may also simply say that $\mathcal{S}$ is action reduced (transition reduced, deterministic).

In the following example we show that the fact that $\mathcal{S}$ is Θ-action reduced (Θ-transition reduced, state reduced) in general does not imply that $\mathcal{T}$ is Θ-action reduced (Θ-transition reduced, state reduced). We moreover show that in case $\mathcal{S}$ is Θ-enabling (Θ-deterministic), then this in general does not imply that $\mathcal{T}$ is Θ-enabling (Θ-deterministic). To show this we use the fact that the transition relation of a synchronized automaton over a set of automata is chosen from the complete transition space. Hence we simply consider a set of automata that satisfies a certain property (i.e. each of its constituting automata satisfies this particular property) and consequently we choose the transition relation of a synchronized automaton over it in such a way that the property fails to hold for that particular synchronized automaton.

Example 4.6.25. Let automata $\mathcal{A}_1 = (\{q_1, q_1'\}, \{a\}, \{(q_1, a, q_1'), (q_1', a, q_1)\}, \{q_1\})$ and $\mathcal{A}_2 = (\{q_2, q_2'\}, \{a, b\}, \{(q_2, b, q_2), (q_2, a, q_2'), (q_2', b, q_2')\}, \{q_2\})$ be as depicted in Figure 4.18.

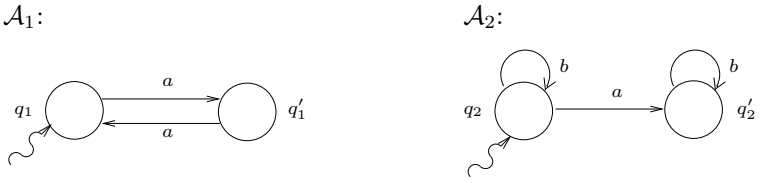

Fig. 4.18. Automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

It is easy to see that both $\mathcal{A}_1$ and $\mathcal{A}_2$ are action reduced, state reduced (and thus transition reduced), and deterministic. Moreover, $\mathcal{A}_1$ is enabling and $\mathcal{A}_2$ is $\{b\}$-enabling.

Now consider the synchronized automaton $\mathcal{T} = (Q, \{a, b\}, \delta, \{(q_1, q_2)\})$, where $Q = \{(q_1, q_2), (q_1, q_2'), (q_1', q_2), (q_1', q_2')\}$ and $\delta = \{((q_1, q_2), a, (q_1', q_2')), ((q_1, q_2), a, (q_1, q_2')), ((q_1', q_2), a, (q_1, q_2'))\}$, over $\{\mathcal{A}_1, \mathcal{A}_2\}$. It is depicted in Figure 4.19(a).

Since b is not active in $\mathcal{T}$ it is clear that $\mathcal{T}$ is not action reduced. Furthermore, $\mathcal{T}$ is not transition reduced (and thus neither state reduced) since $((q_1', q_2), a, (q_1, q_2'))$ is not useful in $\mathcal{T}$. By removing both this transition and

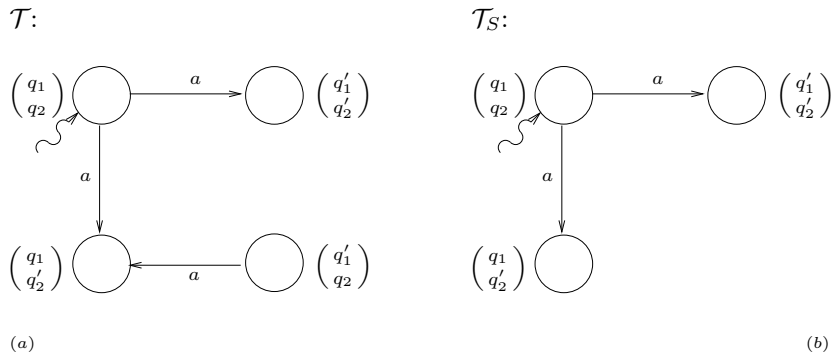

Fig. 4.19. Synchronized automaton $\mathcal{T}$ and its state-reduced version $\mathcal{T}_S$.

the resulting isolated state (q_1', q_2) we obtain the state-reduced version $\mathcal{T}_S$ of $\mathcal{T}$, which is depicted in Figure 4.19(b).

Clearly neither $\mathcal{T}$ nor $\mathcal{T}_S$ is enabling since, e.g., b is not even active in either of these synchronized automata. It is also easy to see that neither $\mathcal{T}$ nor $\mathcal{T}_S$ is deterministic since both synchronized automata contain the transition $((q_1, q_2), a, (q_1, q_2'))$ as well as the transition $((q_1, q_2), a, (q_1', q_2'))$. □

Note that this example thus also suffices to conclude that the Θ-enabling (Θ-determinism) of a set of automata is not inherited by a Θ-action-reduced (Θ-transition-reduced, state-reduced) synchronized automaton over that set of automata.

Summarizing, we conclude that the automata-theoretic properties of Section 3.2 in general are not preserved from a set of automata $\mathcal{S}$ to a synchronized automaton $\mathcal{T}$ over $\mathcal{S}$. We nevertheless show next that — under certain conditions — some of these properties *are* preserved from $\mathcal{S}$ to $\mathcal{T}$.

Reduced Versions

As before we begin by considering action reducedness, transition reducedness, and state reducedness. Note that these properties are based on the notion of reachability of states. We know from Lemma 4.6.11(2) that whenever a state q is reachable in $\mathcal{T}$, then for all $i \in \mathcal{I}$, $\mathrm{proj}_i(q)$ is reachable in $\mathcal{A}_i$. Here we study the inheritance from automata to synchronized automata. Given a state q of a synchronized automaton $\mathcal{T}$ over $\mathcal{S}$ comprising solely reachable states of the automata from $\mathcal{S}$, it is not necessarily the case that q is reachable in $\mathcal{T}$. This is because it may be the case that the transition relation of $\mathcal{T}$ allows no synchronous execution of actions from its constituting automata that would lead to q. In the following example we show that even when we consider

the *maximal-free* (*maximal-ai*, *maximal-si*) synchronized automaton over $\mathcal{S}$, then this may still be the case.

Example 4.6.26. (Examples 4.6.5 and 4.6.20 continued) Note that all the states of all the automata of Examples 4.6.5 and 4.6.20, depicted in Figures 4.11 and 4.14, are reachable. Hence all these automata are state reduced.

Since in Example 4.6.5 both the *maximal-free* synchronized automaton $\mathcal{T}_{1,2}^{free}$ and the *maximal-ai* synchronized automaton $\mathcal{T}_{2,3}^{ai}$ have an empty transition relation, it is however clear that (p, q') and (q', r) are not reachable in $\mathcal{T}_{1,2}^{free}$ and $\mathcal{T}_{2,3}^{ai}$, respectively. Finally, in the *maximal-si* synchronized automaton $\mathcal{T}^{si}$ of Example 4.6.20 — depicted in Figure 4.15(b) — it is clear that (q_1', q_2) is not reachable. Hence neither of these three maximal synchronized automata is state reduced. □

This example thus not only presents counterexamples for the preservation of reachability of states of automata from $\mathcal{S}$ to the *maximal-free* (*maximal-ai*, *maximal-si*) synchronized automaton over $\mathcal{S}$, but it also demonstrates that state reducedness of automata from $\mathcal{S}$ in general is not preserved by the *maximal-free* (*maximal-ai*, *maximal-si*) synchronized automaton over $\mathcal{S}$.

We now show that we can use the notion of loop limitedness to prove the reachability of any state q of the *maximal-free* synchronized automaton $\mathcal{T}$ over $\mathcal{S}$ that comprises solely reachable states of the automata from $\mathcal{S}$. To this aim, we extend Definition 4.6.9 by defining when $\mathcal{S}$ is *Θ-loop-limited*.

Definition 4.6.27. *$\mathcal{S}$ is* Θ-loop limited *if for all $i \in \mathcal{I}$, $\mathcal{S}$ is Θ-i-loop limited.* □

If $\mathcal{S}$ is Σ-loop limited, then we may also simply say that $\mathcal{S}$ is loop limited. Observe that whenever there exists a $k \in \mathcal{I}$ such that $\Theta \subseteq \Sigma_k \setminus (\bigcup_{i \in \mathcal{I} \setminus \{k\}} \Sigma_i)$, then $\mathcal{S}$ is Θ-loop limited. Whenever $\mathcal{S}$ is loop limited and $\mathcal{T}$ is the *maximal-free* synchronized automaton over $\mathcal{S}$, then Theorem 4.6.10 implies that all transitions of the automata from $\mathcal{S}$ are omnipresent in $\mathcal{T}$. Moreover, in all synchronizations of $\mathcal{T}$ only one automaton participates. If in addition $\mathcal{S}$ is finite, then we can thus simply reach q by executing one by one the sequences of transitions responsible for the reachability of those states constituting q.

Lemma 4.6.28. *Let $q \in Q$ be such that for all $i \in \mathcal{I}$, $proj_i(q)$ is reachable in $\mathcal{A}_i$. Then*

> *if $\mathcal{S}$ is finite and loop limited and $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$, for all $a \in \Sigma$, then q is reachable in $\mathcal{T}$.*

Proof. Let $\mathcal{S}$ be finite and loop limited and let $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$, for all $a \in \Sigma$. Now let $\#\mathcal{I} = n$, for some $n \geq 1$, and assume without loss of generality that $\mathcal{I} = [n]$. For all $i \in [n]$, we can fix a computation $\alpha_i = q_{i_0} a_{i_1} q_{i_1} a_{i_2} q_{i_2} \cdots a_{i_{m_i}} q_{i_{m_i}} \in \mathbf{C}_{\mathcal{A}_i}$ such that $m_i \geq 0$, $q_{i_0} \in I_i$, $a_{i_k} \in \Sigma_i$ and $q_{i_k} \in Q_i$, for all $k \in [m_i]$, and $q_{i_{m_i}} = \mathrm{proj}_i(q) \in Q_i$. Consequently, we define β inductively by a sequence $\beta_0, \beta_1, \ldots, \beta_n$ such that $\beta_n = \beta$ as follows.

$\beta_0 = q_0$ is defined by $\mathrm{proj}_i(q_0) = q_{i_0}$, for all $i \in [n]$. Hence $q_0 \in \prod_{i \in [n]} I_i = I$ and $\beta_0 \in \mathbf{C}_{\mathcal{T}}$. Moreover, $\pi_{\mathcal{A}_i}(\beta_0) = q_{i_0}$, for all $i \in [n]$.

$\beta_1 = \beta_0 a_{1_1} q_1 a_{1_2} q_2 \cdots a_{1_{m_1}} q_{m_1}$ is defined, for all $k \in [m_1]$, by $\mathrm{proj}_1(q_k) = q_{1_k}$ and $\mathrm{proj}_i(q_k) = \mathrm{proj}_i(q_0) = q_{i_0}$ if $1 < i \leq n$. Since $(q_{1_{k-1}}, a_{1_k}, q_{1_k}) \in \delta_1$, for all $k \in [m_1]$, $\mathcal{S}$ is loop limited, and $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$, for all $a \in \Sigma$, it follows that $\beta_1 \in \mathbf{C}_{\mathcal{T}}$, $\pi_{\mathcal{A}_1}(\beta_1) = \alpha_1 \in \mathbf{C}_{\mathcal{A}_1}$, and $\pi_{\mathcal{A}_i}(\beta_1) = q_{i_0}$, for all $1 < i \leq n$.

Now let $0 \leq \ell \leq n$ and assume that $\beta_0, \beta_1, \ldots, \beta_{\ell-1}$ are defined in such a way that $\beta_{\ell-1} \in \mathbf{C}_{\mathcal{T}}$, $\pi_{\mathcal{A}_i}(\beta_{\ell-1}) = \alpha_i \in \mathbf{C}_{\mathcal{A}_i}$, for all $i \in [\ell - 1]$, and $\pi_{\mathcal{A}_i}(\beta_{\ell-1}) = q_{i_0}$, for all $\ell \leq i \leq n$.

$\beta_\ell = \beta_{\ell-1} a_{\ell_1} p_1 a_{\ell_2} p_2 \cdots a_{\ell_{m_\ell}} p_{m_\ell}$ is defined, for all $k \in [m_\ell]$, by $\mathrm{proj}_\ell(p_k) = q_{\ell_k}$, $\mathrm{proj}_i(p_k) = q_{i_{m_i}}$ if $i \in [\ell - 1]$, and $\mathrm{proj}_i(p_k) = \mathrm{proj}_i(q_0) = q_{i_0}$ if $\ell < i \leq n$. Since $(q_{\ell_{k-1}}, a_{\ell_k}, q_{\ell_k}) \in \delta_\ell$, for all $k \in [m_\ell]$, $\mathcal{S}$ is loop limited, and $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$, for all $a \in \Sigma$, it follows that $\beta_\ell \in \mathbf{C}_{\mathcal{T}}$, $\pi_{\mathcal{A}_i}(\beta_\ell) = \alpha_i \in \mathbf{C}_{\mathcal{A}_i}$, for all $i \in [\ell]$, and $\pi_{\mathcal{A}_i}(\beta_\ell) = q_{i_0}$, for all $\ell < i \leq n$.

$\beta_n = \beta = q_0 b_1 q_1 b_2 q_2 \cdots b_z q_z$ is thus defined in such a way that $\beta \in \mathbf{C}_{\mathcal{T}}$ and, for all $i \in [n]$, $\pi_{\mathcal{A}_i}(\beta) = \alpha_i \in \mathbf{C}_{\mathcal{A}_i}$ and $\mathrm{proj}_i(q_z) = q_{i_{m_i}} = \mathrm{proj}_i(q)$. Hence q is reachable in $\mathcal{T}$. □

An immediate consequence of Lemma 4.6.28 is that whenever $\mathcal{S}$ is a finite, loop-limited, and state-reduced set of automata, then the *maximal-free* synchronized automaton over $\mathcal{S}$ is state reduced (and thus transition reduced).

Theorem 4.6.29. *Let $\mathcal{S}$ be state reduced. Then*

> *if $\mathcal{S}$ is finite and loop limited and $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$, for all $a \in \Sigma$, then $\mathcal{T}$ is state reduced as well as transition reduced.* □

It is worthwhile to notice that the requirement of every action being *maximal-free* as condition in this theorem cannot be replaced by requiring each action to be *maximal-ai* or *maximal-si* without invalidating the statement. In the following example we show this by demonstrating that the fact that $\mathcal{S}$ is state reduced (and thus transition reduced) in general does not imply that either the *maximal-ai* synchronized automaton over $\mathcal{S}$ or the *maximal-si* synchronized automaton over $\mathcal{S}$ is state reduced — nor does it imply that either of these synchronized automata is transition reduced.

Example 4.6.30. (Example 4.6.20 continued) Clearly $\mathcal{A}_1$ and $\mathcal{A}_2$ form a state-reduced (and thus transition-reduced), finite, and loop-limited set of automata. We have seen, however, that the *maximal-si* synchronized automaton $\mathcal{T}^{si}$ and the *maximal-ai* synchronized automaton $\mathcal{T}^{ai}$ both contain the transition $((q_1', q_2), a, (q_1, q_2'))$ while (q_1', q_2) is not reachable in either of these synchronized automata. Hence neither $\mathcal{T}^{si}$ nor $\mathcal{T}^{ai}$ is transition reduced (and thus neither state reduced). □

Finally, we investigate the conditions under which action reducedness is preserved from $\mathcal{S}$ to a synchronized automaton over $\mathcal{S}$. It turns out that already one action-reduced automaton $\mathcal{A}_k$ in $\mathcal{S}$ guarantees that $\mathcal{T}$ is action reduced, provided that each transition of $\mathcal{A}_k$ is omnipresent in $\mathcal{T}$.

Theorem 4.6.31. *Let $\mathcal{A}_j$ be Θ-action reduced. Then*

if each transition of $\mathcal{A}_j$ is omnipresent in $\mathcal{T}$ and $I \neq \varnothing$, then $\mathcal{T}$ is $\Theta \cap \Sigma_j$-action reduced.

Proof. Let each transition of $\mathcal{A}_j$ be omnipresent in $\mathcal{T}$ and let $I \neq \varnothing$. If $\Theta \cap \Sigma_j = \varnothing$, then there is nothing to prove. We thus assume that $a \in \Theta \cap \Sigma_j$. Then the fact that $\mathcal{A}_j$ is Θ-action reduced implies that there exists a useful transition $(p, a, p') \in \delta_j$ and a computation $p_0 a_1 p_1 a_2 p_2 \cdots a_m p_m a p' \in \mathbf{C}_{\mathcal{A}_j}$ such that $p_m = p$. Now let $q_0 \in I$ be such that $\mathrm{proj}_j(q_0) = p_0$. Then the fact that each transition of $\mathcal{A}_j$ is omnipresent in $\mathcal{T}$ implies that there exists a $(q_0, a_1, q_1) \in \delta$ such that $\mathrm{proj}_j(q_1) = p_1$. By repeating this argument we thus obtain that for all $k \in [m]$, there exists a $(q_{k-1}, a_k, q_k) \in \delta$ such that $\mathrm{proj}_j(q_{k-1}) = p_{k-1}$ and $\mathrm{proj}_j(q_k) = p_k$. This means that there exists a computation $\alpha = q_0 a_1 q_1 a_2 q_2 \cdots a_m q_m \in \mathbf{C}_{\mathcal{T}}$ such that $\pi_{\mathcal{A}_j}(\alpha) = p_0 a_1 p_1 a_2 p_2 \cdots a_m p_m$. Since $\mathrm{proj}_j(q_m) = p_m = p$ and (p, a, p') is omnipresent in $\mathcal{T}$, there must exist a computation $\alpha a q_{m+1} \in \mathbf{C}_{\mathcal{T}}$ such that $\mathrm{proj}_j(q_{m+1}) = p'$. Hence a is active in $\mathcal{T}$ and $\mathcal{T}$ is thus $\Theta \cap \Sigma_j$-action reduced. □

Together with Theorems 4.6.3, 4.6.4, 4.6.8, and 4.6.10(2) this implies the following result.

Corollary 4.6.32. *Let $\mathcal{A}_j$ be Θ-action reduced, let $I \neq \varnothing$, and let $syn \in \{si, no\}$. Then*

(1) if $\delta_a = \mathcal{R}_a^{syn}(\mathcal{S})$, for all $a \in \Sigma$, then $\mathcal{T}$ is $\Theta \cap \Sigma_j$-action reduced,

(2) if $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S})$, for all $a \in \Sigma$, and $\mathcal{S}$ is Θ-enabling, then $\mathcal{T}$ is $\Theta \cap \Sigma_j$-action reduced, and

(3) if $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$, for all $a \in \Sigma$, and $\mathcal{S}$ is Θ-j-loop limited, then $\mathcal{T}$ is $\Theta \cap \Sigma_j$-action reduced. □

Enabling

We now turn to an investigation of the conditions under which enabling is preserved from $\mathcal{S}$ to a synchronized automaton over $\mathcal{S}$. It turns out that already one enabling automaton $\mathcal{A}_k$ in $\mathcal{S}$ guarantees that $\mathcal{T}$ is enabling, provided that each transition of $\mathcal{A}_k$ is omnipresent in $\mathcal{T}$.

Theorem 4.6.33. *Let $\mathcal{A}_j$ be Θ-enabling. Then*

if each a-transition of $\mathcal{A}_j$, for all $a \in \Theta$, is omnipresent in $\mathcal{T}$, then $\mathcal{T}$ is $\Theta \cap \Sigma_j$-enabling.

Proof. Let each a-transition of $\mathcal{A}_j$, for all $a \in \Theta$, be omnipresent in $\mathcal{T}$. If $\Theta \cap \Sigma_j = \varnothing$, then there is nothing to prove. We thus assume that $a \in \Theta \cap \Sigma_j$. Now let $q \in Q$. Since $a \in \Sigma_j$ and $\mathcal{A}_j$ is Θ-enabling we know that $a \ \mathrm{en}_{\mathcal{A}_j} \ \mathrm{proj}_j(q)$. The fact that each a-transition of $\mathcal{A}_j$, for all $a \in \Theta$, is omnipresent in $\mathcal{T}$ consequently implies that $a \ \mathrm{en}_{\mathcal{T}} \ q$. Hence $\mathcal{T}$ is $\Theta \cap \Sigma_j$-enabling. □

Together with Theorems 4.6.3, 4.6.4, 4.6.8, and 4.6.10(2) this implies the following result.

Corollary 4.6.34. *Let $\mathcal{A}_j$ be Θ-enabling and let $syn \in \{si, no\}$. Then*

(1) if $\delta_a = \mathcal{R}_a^{syn}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_j$, then $\mathcal{T}$ is $\Theta \cap \Sigma_j$-enabling,

(2) if $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_j$, and $\mathcal{S}$ is Θ-enabling, then $\mathcal{T}$ is $\Theta \cap \Sigma_j$-enabling, and

(3) if $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma_j$, and $\mathcal{S}$ is Θ-j-loop limited, then $\mathcal{T}$ is $\Theta \cap \Sigma_j$-enabling. □

Determinism

Finally, we turn to an investigation of the conditions under which determinism is preserved from $\mathcal{S}$ to a synchronized automaton over $\mathcal{S}$. It turns out that whenever $\mathcal{S}$ is deterministic, then so is $\mathcal{T}$ provided that all its actions are *maximal-ai* or *maximal-si*.

Theorem 4.6.35. *Let $\mathcal{S}$ be Θ-deterministic and let $syn \in \{ai, si\}$. Then*

if $\delta_a \subseteq \mathcal{R}_a^{syn}(\mathcal{S})$, for all $a \in \Theta \cap \Sigma$, then $\mathcal{T}$ is Θ-deterministic.

Proof. Let $a \in \Theta \cap \Sigma$ and let $\delta_a \subseteq \mathcal{R}_a^{syn}(\mathcal{S})$. Now assume there exists a $q \in Q$ such that $(q, q') \in \delta_a$ and $(q, q'') \in \delta_a$, with $q' \neq q''$. Then there must exist an $i \in \mathcal{I}$ such that $\text{proj}_i(q') \neq \text{proj}_i(q'')$. Now we have two possibilities.
If $\text{proj}_i{}^{[2]}(q, q') \in \delta_{i,a}$ and $\text{proj}_i{}^{[2]}(q, q'') \in \delta_{i,a}$, then $\mathcal{A}_i$ is not $\{a\}$-deterministic, a contradiction.
If $\text{proj}_i{}^{[2]}(q, q') \in \delta_{i,a}$ and $\text{proj}_i{}^{[2]}(q, q'') \notin \delta_{i,a}$ or — vice versa — $\text{proj}_i{}^{[2]}(q, q') \notin \delta_{i,a}$ and $\text{proj}_i{}^{[2]}(q, q'') \in \delta_{i,a}$, then $(q, q'') \notin \mathcal{R}_a^{syn}(\mathcal{S})$ or — respectively — $(q, q') \notin \mathcal{R}_a^{syn}(\mathcal{S})$, a contradiction in either way.
Hence $q' = q''$ and $\mathcal{T}$ is thus Θ-deterministic. □

We note that this theorem does not cover the case of *maximal-free* synchronized automata. In fact, if $\mathcal{S}$ is Θ-deterministic, then this in general does not imply that also the *maximal-free* synchronized automaton over $\mathcal{S}$ is Θ-deterministic. This can be concluded from Example 4.6.20, where it is easy to see that $\{\mathcal{A}_1, \mathcal{A}_2\}$ is loop limited and deterministic, whereas $\mathcal{T}^{free}$ is not deterministic. This implies that neither the Θ-determinism of the $\mathcal{R}^{no}$-team automaton over $\mathcal{S}$ is implied by the Θ-determinism of $\mathcal{S}$.

4.6.3 Conclusion

This section forms a detailed, although limited, account of our initial investigation of the top-down inheritance — from synchronized automata to their (sub)automata — and the bottom-up preservation — from automata to synchronized automata — of the automata-theoretic properties from Section 3.2. The obtained results lean heavily on the presence and omnipresence of transitions of (sub)automata in synchronizations of synchronized automata over these (sub)automata. These two auxiliary notions have been treated in an intermezzo preceding our investigation.

We have focused on *maximal-free*, *maximal-ai*, and *maximal-si* synchronized automata. To a lesser degree we have moreover considered synchronized automata in which either every action is *free*, or every action is *ai*, or every action is *si*. Results on the $\mathcal{R}^{no}$-synchronized automaton over $\mathcal{S}$ have been mentioned only when they required almost no effort. Finally, the only additional conditions that have been considered in our search for sufficient conditions under which the automata-theoretic properties from Section 3.2 are inherited top-down or preserved bottom-up, are the loop limitedness and enabling of $\mathcal{S}$. Consequently, for many of these properties it remains to narrow down which combinations of specific conditions and types of (synchronized) automata guarantee their top-down inheritance and their bottom-up preservation. Furthermore, once other types of synchronization have been introduced, inheritance and preservation can be considered in the context of a broader class of synchronized automata (cf. Chapter 5).

4.7 Inheritance of Synchronizations

In the previous section we investigated the effect that the types of synchronization introduced in Sections 4.4 and 4.5 have on the inheritance of the automata-theoretic properties from Section 3.2. In this section we investigate the conditions under which these types of synchronization are themselves inherited top-down — from synchronized automata to subautomata — and preserved bottom-up — from subautomata to synchronized automata.

Note that we deal with synchronizations *between* automata constituting a synchronized automaton. There is thus no need to study whether synchronizations are inherited by automata from synchronized automata — and vice versa — since in any automaton — and in any synchronized automaton over a single automaton — all its actions trivially are *free*, *ai*, and *si*.

We begin by studying the inheritance of the types of synchronization introduced in Section 4.4. The property of an action a being *free* (*ai*, *si*) in a synchronized automaton is inherited by all its subautomata having a as one of their actions.

Lemma 4.7.1. *(1)* $\Sigma_J \cap \mathit{Free}(\mathcal{T}) \subseteq \mathit{Free}(\mathit{SUB}_J)$,

(2) $\Sigma_J \cap \mathit{AI}(\mathcal{T}) \subseteq \mathit{AI}(\mathit{SUB}_J)$, *and*

(3) $\Sigma_J \cap \mathit{SI}(\mathcal{T}) \subseteq \mathit{SI}(\mathit{SUB}_J)$.

Proof. (1) Let $a \in \Sigma_J \cap \mathit{Free}(\mathcal{T})$. Now assume that $a \notin \mathit{Free}(\mathit{SUB}_J)$. This means there must exist a transition $(p, a, p') \in \delta_J$ such that $\#\{i \in J \mid \mathrm{proj}_i^{[2]}(p, p') \in \delta_{i,a}\} > 1$. Then Theorem 4.6.2 implies that there exists a $(q, q') \in \delta_a$ such that $\mathrm{proj}_J^{[2]}(q, q') = (p, p')$, and thus $\#\{i \in \mathcal{I} \mid \mathrm{proj}_i^{[2]}(q, q') \in \delta_{i,a}\} > 1$. This contradicts the fact that a is *free* in $\mathcal{T}$. Hence $a \in \mathit{Free}(\mathit{SUB}_J)$.

(2,3) Analogous. □

Note that the proof of Lemma 4.7.1 relies heavily on the observation that in a subautomaton of a synchronized automaton no (new) transitions — i.e. other than those obtained as projections of existing transitions of the synchronized automaton — are introduced. Hence if there exists a transition in SUB_J violating the *free* (*ai*, *si*) requirement for a, then Theorem 4.6.2 implies that this transition is present in $\mathcal{T}$, i.e. there exists an "extension" of this transition in $\mathcal{T}$ which also violates the *free* (*ai*, *si*) requirement for a.

The converses of the statements of Lemma 4.7.1 in general do not hold. The reason for this resides in the fact that an action a that is not *free* in a synchronized automaton $\mathcal{T}$, *is free* in a subautomaton of $\mathcal{T}$ provided the

restriction to a subset of the automata leads to dropping those automata from $\mathcal{T}$ that caused a not to be *free* in $\mathcal{T}$. The same reasoning can be applied in case a is *ai* or *si*. In the following example we demonstrate this.

Example 4.7.2. (Example 4.4.8 continued) We have seen that in synchronized automaton $\mathcal{T}^1$ action a is neither *free*, nor *ai*, nor *si*. However, in subautomaton $SUB_{\{2\}}(\mathcal{T}^1)$ — which is essentially a copy of $\mathcal{A}_2$ — action a trivially is *free*, *ai*, and *si*. □

We now demonstrate that the converses of the statements of Lemma 4.7.1 do hold if always only one automaton participates in the execution of an action, as is the case for internal actions. More general, whenever an action only belongs to automata which are included in a subautomaton, then the properties of being *free* (*ai*, *si*) are preserved from that subautomaton to the synchronized automaton as a whole.

Lemma 4.7.3. *Let* $\Sigma_J \cap (\bigcup_{i \in \mathcal{I} \setminus J} \Sigma_i) = \varnothing$. *Then*

(1) $Free(SUB_J) \subseteq \Sigma_J \cap Free(\mathcal{T})$,

(2) $AI(SUB_J) \subseteq \Sigma_J \cap AI(\mathcal{T})$, *and*

(3) $SI(SUB_J) \subseteq \Sigma_J \cap SI(\mathcal{T})$.

Proof. (1) Let $a \in Free(SUB_J)$. Hence $a \in \Sigma_J$. Now assume that $a \notin Free(\mathcal{T})$. Then there exists a transition $(q, q') \in \delta_a$ that violates the requirement for a to be *free* in $\mathcal{T}$. However, since $\Sigma_J \cap (\bigcup_{i \in \mathcal{I} \setminus J} \Sigma_i) = \varnothing$, we have that for all $i \in \mathcal{I} \setminus J$, $a \notin \Sigma_i$. We conclude that the violation of the requirement for a to be *free* in $\mathcal{T}$ thus occurs in SUB_J, i.e. $\mathrm{proj}_J{}^{[2]}(q, q')$ violates the requirement for a to be *free* in SUB_J, a contradiction. Hence $a \in \Sigma_J \cap Free(\mathcal{T})$.

(2,3) Analogous. □

Together with Lemma 4.7.1, this lemma implies the following result.

Theorem 4.7.4. *Let* $\Sigma_J \cap (\bigcup_{i \in \mathcal{I} \setminus J} \Sigma_i) = \varnothing$. *Then*

(1) $\Sigma_J \cap Free(\mathcal{T}) = Free(SUB_J)$,

(2) $\Sigma_J \cap AI(\mathcal{T}) = AI(SUB_J)$, *and*

(3) $\Sigma_J \cap SI(\mathcal{T}) = SI(SUB_J)$. □

Finally, we conclude this section with a result on the inheritance of the maximal types of synchronization introduced in Section 4.5. We show that under certain conditions, the property of an action a being *maximal-free* (*maximal-ai*, *maximal-si*) in a synchronized automaton is inherited by each subautomaton of that synchronized automaton having a as one of its actions.

Theorem 4.7.5. *Let $a \in \Sigma_J$. Then*

(1) if $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$ and $\mathcal{S}$ is $\{a\}$-loop limited, then $(\delta_J)_a = \mathcal{R}_a^{free}(\{\mathcal{A}_j \mid j \in J\})$,

(2) if $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S}) \neq \varnothing$, then $(\delta_J)_a = \mathcal{R}_a^{ai}(\{\mathcal{A}_j \mid j \in J\})$, and

(3) if $\delta_a = \mathcal{R}_a^{si}(\mathcal{S})$, then $(\delta_J)_a = \mathcal{R}_a^{si}(\{\mathcal{A}_j \mid j \in J\})$.

Proof. (1) Let $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$ and let $\mathcal{S}$ be $\{a\}$-loop limited. Then according to Lemma 4.7.1(1) we only need to prove that $\mathcal{R}_a^{free}(\{\mathcal{A}_j \mid j \in J\}) \subseteq (\delta_J)_a$. Now let $(q, q') \in \mathcal{R}_a^{free}(\{\mathcal{A}_j \mid j \in J\})$. Then there exists a $k \in J$ such that $\mathrm{proj}_k{}^{[2]}(q, q') = (p, p') \in \delta_{k,a}$ and for all $i \in J \setminus \{k\}$, $\mathrm{proj}_i(q') = \mathrm{proj}_i(q)$. Since $\mathcal{S}$ is $\{a\}$-loop limited it follows from Theorem 4.6.10(2) that (p, a, p') is omnipresent in $\mathcal{T}$. Together with the fact that $\delta_a = \mathcal{R}_a^{free}(\mathcal{S})$ this implies that there must exist an $(r, r') \in \delta_a$ such that $\mathrm{proj}_J{}^{[2]}(r, r') = (q, q')$ and thus $(q, q') = \mathrm{proj}_J{}^{[2]}(r, r') \in (\delta_J)_a$.

(2) Let $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S}) \neq \varnothing$. Then by Lemma 4.7.1(2) we only need to prove that $\mathcal{R}_a^{ai}(\{\mathcal{A}_j \mid j \in J\}) \subseteq (\delta_J)_a$. Now let $(q, q') \in \mathcal{R}_a^{ai}(\{\mathcal{A}_j \mid j \in J\})$. Then there exists a $K \subseteq J$ such that for all $k \in K$, $a \in \Sigma_k$, $\mathrm{proj}_k{}^{[2]}(q, q') \in \delta_{k,a}$, and for all $i \in J \setminus K$, $\mathrm{proj}_i{}^{[2]}(q, q') \notin \delta_{i,a}$ and $a \notin \Sigma_i$. Since $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S}) \neq \varnothing$, it follows from Theorem 4.6.6 that for all $k \in K$, $(\mathrm{proj}_k(q), a, \mathrm{proj}_k(q'))$ is present in $\mathcal{T}$. Together with the fact that $\delta_a = \mathcal{R}_a^{ai}(\mathcal{S})$ this implies that there must exist an $(r, r') \in \delta_a$ such that $\mathrm{proj}_J{}^{[2]}(r, r') = (q, q')$ and thus $(q, q') = \mathrm{proj}_J{}^{[2]}(r, r') \in (\delta_J)_a$.

(3) Let $\delta_a = \mathcal{R}_a^{si}(\mathcal{S})$. Then according to Lemma 4.7.1(3) we only need to prove that $\mathcal{R}_a^{si}(\{\mathcal{A}_j \mid j \in J\}) \subseteq (\delta_J)_a$. Now let $(q, q') \in \mathcal{R}_a^{si}(\{\mathcal{A}_j \mid j \in J\})$. Then there exists a $K \subseteq J$ such that for all $k \in K$, $\mathrm{proj}_k{}^{[2]}(q, q') \in \delta_{k,a}$ and for all $i \in J \setminus K$, $\mathrm{proj}_i{}^{[2]}(q, q') \notin \delta_{i,a}$ and a is not enabled in $\mathcal{A}_i$ at $\mathrm{proj}_i(q)$. From Theorem 4.6.3 it now follows that for all $k \in K$, $(\mathrm{proj}_k(q), a, \mathrm{proj}_k(q'))$ is omnipresent in $\mathcal{T}$. Together with the fact that $\delta_a = \mathcal{R}_a^{si}(\mathcal{S})$ this implies that there must exist an $(r, r') \in \delta_a$ such that $\mathrm{proj}_J{}^{[2]}(r, r') = (q, q')$ and thus $(q, q') = \mathrm{proj}_J{}^{[2]}(r, r') \in (\delta_J)_a$. □