Maurice H. ter Beek

ISTI–CNR, Pisa, Italy

LIACS, Leiden University
15 July 2015
Team Automata: Communication and Compatibility

Maurice H. ter Beek

ISTI–CNR, Pisa, Italy

LIACS, Leiden University
15 July 2015
Outline

1 Origins

2 Team Automata

3 Synchronization strategies

4 (Synchronized) shuffles

5 Compositionality of team automata

6 Communication and compatibility

7 Future work
Some background...

<table>
<thead>
<tr>
<th>Computer Supported Cooperative Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>? how do groups of people work together</td>
</tr>
<tr>
<td>? how can technology (groupware) assist</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ellis @ GROUP’97: Team Automata for Groupware Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual level: define concepts and terminology consistently</td>
</tr>
<tr>
<td>Architectural level: describe and compare groupware systems</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theoretical source of inspiration and foundations</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Lynch &amp; Tuttle @ PODC’87: Input/Output Automata (IOA)</td>
</tr>
<tr>
<td>• ter Beek, Ellis, Kleijn &amp; Rozenberg in CSCW journal (2003): Synchronizations in Team Automata for Groupware Systems</td>
</tr>
</tbody>
</table>
Component Automaton

\[ C = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I) \]

- \( Q \) set of states

- \( \Sigma = \Sigma_{inp} \cup \Sigma_{out} \cup \Sigma_{int} \) alphabet (partition)

- \( \delta \subseteq Q \times \Sigma \times Q \) transition relation

- \( I \subseteq Q \) set of initial states

\( \Sigma_{inp} \) input actions

\( \Sigma_{out} \) output actions

\( \Sigma_{int} \) internal actions cannot be observed

\( \delta_a = \{(q, q') \mid (q, a, q') \in \delta\} \) set of \( a \)-transitions

\( a \in C \) \( q \) if \( \exists q' \in Q : (q, q') \in \delta_a \) \( a \) enabled at \( q \)
A composable system $S = \{C_1, \ldots, C_n\}$ of component automata is a *composable system* if

$$\forall i \in \{1, \ldots, n\}:
\Sigma_i, int \cap \bigcup_{j \in \{1, \ldots, n\}\setminus\{i\}} \Sigma_j = \emptyset$$

$\Rightarrow$ internal actions are private, uniquely associated to one component

**Fix:** $S = \{C_1, \ldots, C_n\}$ is a composable system with

$$\Sigma = \bigcup_{i \in \{1, \ldots, n\}} (\Sigma_{i, inp} \cup \Sigma_{i, out} \cup \Sigma_{i, int})$$

? how to form a team automaton over a composable system
The complete transition space of \( a \in \Sigma \) in \( S \) is

\[
\Delta_a(S) = \{(q, q') \in \prod_{i \in \{1, \ldots, n\}} Q_i \times \prod_{i \in \{1, \ldots, n\}} Q_i \mid \exists j \in \{1, \ldots, n\}: (\text{proj}_j(q), a, \text{proj}_j(q')) \in \delta_j \land \forall i \in \{1, \ldots, n\}: (\text{proj}_i(q), a, \text{proj}_i(q')) \in \delta_i \lor \text{proj}_i(q) = \text{proj}_i(q')\}
\]

Hence, in every team transition:

1. at least one component acts according to its transition relation
2. all other components either join or are idle

\( \Rightarrow \) Moreover, each choice of team transition relations \( \delta_a \subseteq \Delta_a(S) \), \( \forall a \in \Sigma \), defines a specific team automaton
Team Automata

\[ T = \left( \prod_{i \in \{1, \ldots, n\}} Q_i, (\Sigma_{\text{inp}}, \Sigma_{\text{out}}, \Sigma_{\text{int}}), \delta, \prod_{i \in \{1, \ldots, n\}} l_i \right) \]

is a team automaton composed over \( S \) if

- \( \Sigma_{\text{int}} = \bigcup_{i \in \{1, \ldots, n\}} \Sigma_{i,\text{int}} \)
- \( \Sigma_{\text{out}} = \bigcup_{i \in \{1, \ldots, n\}} \Sigma_{i,\text{out}} \)
- \( \Sigma_{\text{inp}} = \left( \bigcup_{i \in \{1, \ldots, n\}} \Sigma_{i,\text{inp}} \right) \setminus \Sigma_{\text{out}} \)
- \( \delta \subseteq \prod_{i \in \{1, \ldots, n\}} Q_i \times \Sigma \times \prod_{i \in \{1, \ldots, n\}} Q_i \) such that \( \forall a \in \Sigma: \delta_a \subseteq \Delta_a(S) \) and
  \( \delta_a = \Delta_a(S) \) if \( a \in \Sigma_{\text{int}} \)

\( \Rightarrow \) every team is a component and can thus be used to iteratively compose team automata
Examples

$C_1$:

![Diagram of $C_1$]

$C_2$:

![Diagram of $C_2$]

Team automata over $\{C_1, C_2\}$ defined by choosing their transitions:

$\mathcal{T}^{\text{free}}$: 

$\mathcal{T}^{\text{ai}}$: 

? who participates in $((q_1, q'_2), b, (q_1, q'_2))$ of $\mathcal{T}^{\text{ai}}$: only $C_1$ or $C_2$ or both

! adopt maximal interpretation: both execute $b$
Computations and behaviour (languages)

\[ \alpha = \left( \begin{array}{c} q_1 \\ q_2 \end{array} \right) a \left( \begin{array}{c} q_1 \\ q'_2 \end{array} \right) a \left( \begin{array}{c} q'_1 \\ q_2 \end{array} \right) \in C_{\mathcal{T}\text{free}} \]

\[ \pi_{C_1}(\alpha) = q_1 a q'_1 \in C_{C_1} \]

\[ \pi_{C_2}(\alpha) = q_2 a q'_2 \in C_{C_2} \]

⇒ if \( \mathcal{T} \) is a team automaton over \( \mathcal{S} \), then \( \forall i \in \{1, \ldots, n\} \):

\[ \pi_{C_i}(C_{\mathcal{T}}) \subseteq C_{C_i} \]

\[ \text{pres}_{\{a,b\}}(\alpha) = aa \in B_{\mathcal{T}\text{free}} \]

\[ aa \notin B_{C_1} \cup B_{C_2} \]

\[ ab \notin B_{\mathcal{T}\text{free}} \cup B_{\mathcal{T}ai} \]

\[ ab \in B_{C_2} \]
The subteam of $\mathcal{T}$ determined by $J \subseteq \{1, \ldots, n\}$ is

$$SUB_J(\mathcal{T}) = \left( \prod_{j \in J} Q_j, (\Sigma_{J,\text{inp}}, \Sigma_{J,\text{out}}, \Sigma_{J,\text{int}}), \delta_J, \prod_{j \in J} I_j \right)$$

- $\Sigma_{J,\text{int}} = \bigcup_{j \in J} \Sigma_{j,\text{int}}$
- $\Sigma_{J,\text{out}} = \bigcup_{j \in J} \Sigma_{j,\text{out}}$
- $\Sigma_{J,\text{inp}} = (\bigcup_{j \in J} \Sigma_{j,\text{inp}}) \setminus \Sigma_{J,\text{out}}$

$\delta_J$ is defined by, $\forall a \in \Sigma_J$,

$$\delta_J \left( a \right) = \left\{ (\text{proj}_J(q), \text{proj}_J(q')) \mid (q, q') \in \delta_a \right\} \cap \Delta_a(\{C_j \mid j \in J\})$$

$\Rightarrow SUB_J(\mathcal{T})$ is a team automaton over $\{C_j \mid j \in J\}$
Teams over teams

Key notions so far

- Component automaton
- Composable system (of component automata)
- Team automaton (over a composable system)
- Subteam (of a team automaton)

Next: compositionality

- Iterated team automata
- Synchronizations in team automata

⇒ after reordering its elements, $\mathcal{T}''$ is a team automaton over $\{C_1, \ldots, C_7\}$
Synchronizations: *free*, *action*- and *state-indispensable*

An action $a$ of a team automaton $\mathcal{T}$ over $S$ is:

- **free** if no $a$-transition of $\mathcal{T}$ contains the simultaneous execution of $a$ by two or more components
- **ai** if all components with $a$ as an action participate in every $a$-transition of $\mathcal{T}$ (blocking can occur)
- **si** if all $a$-transitions of $\mathcal{T}$ involve all components in which $a$ is enabled in the current state (no blocking can occur)
- **no** no restrictions whatsoever
Synchronizations: \textit{free}, \textit{action}- and \textit{state-indispensable}

An action $a$ of a team automaton $\mathcal{T}$ over $S$ is:

\begin{align*}
\text{free} & \quad \delta_a = \{(q, q') \in \Delta_a(S) \mid \#\{i \in \{1, \ldots, n\} \mid a \in \Sigma_i \land (\text{proj}_i(q), \text{proj}_i(q')) \in \delta_{i,a}\} = 1\} \\
\text{ai} & \quad \delta_a = \{(q, q') \in \Delta_a(S) \mid \forall i \in \{1, \ldots, n\}: a \in \Sigma_i \Rightarrow (\text{proj}_i(q), \text{proj}_i(q')) \in \delta_{i,a}\} \\
\text{si} & \quad \delta_a = \{(q, q') \in \Delta_a(S) \mid \forall i \in \{1, \ldots, n\}: (a \in \Sigma_i \land a \in C_i \land \text{proj}_i(q)) \Rightarrow (\text{proj}_i(q), \text{proj}_i(q')) \in \delta_{i,a}\} \\
\text{no} & \quad \delta_a = \Delta_a(S)
\end{align*}

Fix: team automaton $\mathcal{T} = (Q, (\Sigma_{\text{inp}}, \Sigma_{\text{out}}, \Sigma_{\text{int}}), \delta, I)$ over $S$

Recall $\Sigma = \Sigma_{\text{inp}} \cup \Sigma_{\text{out}} \cup \Sigma_{\text{int}}$ and let $a \in \Sigma$
Synchronizations: *free*, *action-* and *state-indispensable*

An action \( a \) of a team automaton \( \mathcal{T} \) over \( S \) is:

\[
\begin{align*}
\text{free } \delta_a &= \{(q, q') \in \Delta_a(S) \mid \\ &\#\{i \in \{1, \ldots, n\} \mid a \in \Sigma_i \land (\text{proj}_i(q), \text{proj}_i(q')) \in \delta_{i,a}\} = 1\} \\
\text{ai } \delta_a &= \{(q, q') \in \Delta_a(S) \mid \forall i \in \{1, \ldots, n\} : \\ &a \in \Sigma_i \Rightarrow (\text{proj}_i(q), \text{proj}_i(q')) \in \delta_{i,a}\} \\
\text{si } \delta_a &= \{(q, q') \in \Delta_a(S) \mid \forall i \in \{1, \ldots, n\} : \\ &(a \in \Sigma_i \land a \in C_i \land \text{proj}_i(q)) \Rightarrow (\text{proj}_i(q), \text{proj}_i(q')) \in \delta_{i,a}\} \\
\text{no } \delta_a &= \Delta_a(S)
\end{align*}
\]

Fix: team automaton \( \mathcal{T} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I) \) over \( S \)

recall \( \Sigma = \Sigma_{inp} \cup \Sigma_{out} \cup \Sigma_{int} \) and let \( a \in \Sigma \)
Fixed team automata

Let $R_a(S) \subseteq \Delta_a(S)$, for all $a \in \Sigma$, and let $R_\Sigma = \{R_a(S) \mid a \in \Sigma\}$

Then $T$ is the $R_\Sigma$-team automaton over $S$ if $\forall a \in \Sigma : \delta_a = R_a(S)$

⇒ maximal interpretation: $((q_1, q'_2), b, (q_1, q'_2))$ is part of the $R^{ai}$-team automaton $T^{ai}$ but not of the $R^{free}$-team automaton $T^{free}$ (omit $\Sigma$)

Synchronization strategies per action, but composition can be based on combinations, e.g. $(R^{ai}_T \cup R^{free}_\Sigma \setminus \Gamma)$-team automaton

Most commonly used synchronization strategy in the literature is ai-based (e.g. IOA)

⇒ peer-to-peer and master-slave types of synchronization cannot be distinguished in IOA
Recall examples

**\( C_1 \):**

\[
\begin{array}{c}
q_1 \\
\end{array}
\xrightarrow{a}
\begin{array}{c}
q_1'
\end{array}
\xrightarrow{b}
\begin{array}{c}
q_1
\end{array}
\]

**\( C_2 \):**

\[
\begin{array}{c}
q_2
\end{array}
\xrightarrow{a}
\begin{array}{c}
q_2'
\end{array}
\xrightarrow{b}
\begin{array}{c}
q_2
\end{array}
\]

**\( \mathcal{T}^{\text{free}} \):**

\[
\begin{array}{c}
\begin{pmatrix} q_1 \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{b}
\begin{array}{c}
\begin{pmatrix} q_1 \\ q_2' \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2' \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1 \\ q_2' \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2 \\ q_1 \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2 \\
q_1 \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{b}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2 \\ q_1 \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{b}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1 \\ q_2 \end{pmatrix}
\end{array}
\]

**\( \mathcal{T}^{\text{ai}} \):**

\[
\begin{array}{c}
\begin{pmatrix} q_1 \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1 \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{b}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{b}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1 \\ q_2 \end{pmatrix}
\end{array}
\xrightarrow{a}
\begin{array}{c}
\begin{pmatrix} q_1' \\ q_2 \end{pmatrix}
\end{array}
\]
Synchronizations: peer-to-peer collaboration

Action $a$ is strong (weak) output peer-to-peer in $\mathcal{T}$ if $a$ is $ai$ ($si$) in $\text{SUB}_{a,\text{out}}(\mathcal{T})$

Action $a$ is strong (weak) input peer-to-peer in $\mathcal{T}$ if $a$ is $ai$ ($si$) in $\text{SUB}_{a,\text{in}}(\mathcal{T})$
Synchronizations: *master-slave* cooperation

Action $a$ is *strong (weak) master-slave* in $\mathcal{T}$ if

1. $\text{SUB}_{a,\text{out}}(\mathcal{T})$ participates in every $a$-transition of $\mathcal{T}$, i.e. slaves may never proceed on their own

2. $\text{SUB}_{a,\text{inp}}(\mathcal{T})$ participates in every $a$-transition of $\mathcal{T}$, i.e. slaves must obey the master (if enabled)
A team automaton over $\mathcal{S}$ is said to satisfy *compositionality* if its behaviour can be described in terms of that of its component automata:

There exists a set-theoretic operation that, if applied to the languages of the component automata in $\mathcal{S}$, yields the language of the team automaton.
A team automaton over $S$ is said to satisfy \textit{compositionality} if its behaviour can be described in terms of that of its component automata: there exists a set-theoretic operation that, if applied to the languages of the component automata in $S$, yields the language of the team automaton.

\begin{mdt}
\textbf{Compositionality of fixed team automata}

which (combinations of) synchronization strategies result in team automata satisfying compositionality
\end{mdt}
Motivation for (synchronized) shuffles

- **Component automaton**: ordinary FSA without final states and its alphabet partitioned into input, output and internal actions
  - prefix-closed languages that may contain *infinite* words

- **Team automaton**: (iterative) composition of component automata that may collaborate by synchronizing on shared actions
  - languages may be *unfair* in the sense that a component may execute ad infinitum, never giving others a turn

- **Compositionality**: language of a team automaton can be defined in terms of the languages of its constituting component automata
  - (synchronized) shuffling of languages must be *associative*
Shuffling: basic definitions

Let $u, v \in \Delta^\infty$. Then

1. $w \in \Delta^\infty$ is a *fair shuffle* of $u$ and $v$ if $w = u_1 v_1 u_2 v_2 \cdots$, where $u_i, v_i \in \Delta^*$, for all $i \geq 1$, are such that $u = u_1 u_2 \cdots$ and $v = v_1 v_2 \cdots$

2. $w \in \Delta^\infty$ is a *shuffle* of $u$ and $v$ if either
   1. $w$ is a fair shuffle of $u$ and $v$, or
   2. $w = u_1 v_1 u_2 v_2 \cdots$, where $u_i, v_i \in \Delta^*$, for all $i \geq 1$, and either $u_1 u_2 \cdots \in \text{pref}(u)$ and $v = v_1 v_2 \cdots \in \Delta^\omega$
      or $u = u_1 u_2 \cdots \in \Delta^\omega$ and $v_1 v_2 \cdots \in \text{pref}(v)$

Notations:

$$u \parallel\!\!\!\parallel v = \{ w \in \Delta^\infty : w \text{ is a fair shuffle of } u \text{ and } v \}$$

$$u \parallel v = \{ w \in \Delta^\infty : w \text{ is a shuffle of } u \text{ and } v \}$$

$$L_1 \parallel\!\!\!\parallel L_2 = \bigcup_{u \in L_1, v \in L_2} u \parallel\!\!\!\parallel v$$

$$L_1 \parallel L_2 = \bigcup_{u \in L_1, v \in L_2} u \parallel v$$
Examples: fair and unfair shuffling

\[ a \parallel b = \{ ab, ba \} \]
\[ a^n \parallel b = \{ a^i b a^j : i, j \geq 0, \ i + j = n \} \]
\[ a^2 \parallel b = \{ a^2 b, aba, ba^2 \} \]
\[ a^n \parallel b = \{ a^i b a^j : i, j \geq 0, \ i + j = n \} \]
\[ a^2 \parallel b = \{ a^2 b, aba, ba^2 \} \]

\[ \Rightarrow \text{ every shuffle of } a^n \text{ and } b \text{ is fair} \]

\[ a^\omega \parallel b = \{ a^i b a^\omega : i \geq 0 \} \]
\[ a^\omega \parallel b = (a^\omega \parallel b) \cup a^\omega \]
\[ a^\omega \parallel b = \{ a^i b a^\omega : i \geq 0 \} \]
\[ a^\omega \parallel b = (a^\omega \parallel b) \cup a^\omega \]

\[ \Rightarrow a^\omega \text{ is an unfair shuffle of } a^\omega \text{ and } b \]

\[ a^\omega \parallel a = a^\omega \parallel a = a^\omega \]

\[ \Rightarrow \text{ infinite words need not result in unfair shuffles} \]
Let $u, v, w \in \Delta^\infty$. Then

1. (fair) shuffling is **commutative**: $u \shuffling v = v \shuffling u$ and $u \shuffling v = v \shuffling u$

2. fair shuffling is **associative**: $u \shuffling (v \shuffling w) = (u \shuffling v) \shuffling w$

3. $\text{pref}(u) \shuffling \text{pref}(v) = \text{pref}(u \shuffling v) = \text{pref}(u \shuffling v) = \text{pref}(u) \shuffling \text{pref}(v)$

4. $\Rightarrow \text{pref}(a^\omega \shuffling b) = \text{pref}(a^\omega \shuffling b) = \{ a^i b a^\omega : i \geq 0 \} \cup a^*$, but recall that $a^\omega \shuffling b \neq a^\omega \shuffling b$

5. Shuffling is **associative**: $u \shuffling (v \shuffling w) = (u \shuffling v) \shuffling w$
Shuffling: basic observations and our result

Let \( u, v, w \in \Delta^\infty \). Then

1. (fair) shuffling is **commutative**: \( u ||| v = v ||| u \) and \( u || v = v || u \)
2. fair shuffling is **associative**: \( u || (v || w) = (u || v) || w \)
3. \( \text{pref}(u) || \text{pref}(v) = \text{pref}(u || v) = \text{pref}(u || v) = \text{pref}(u) || \text{pref}(v) \)

\[ \Rightarrow \text{pref}(a^\omega || b) = \text{pref}(a^\omega || b) = \{ a^i ba^\omega : i \geq 0 \} \cup a^* \text{, but recall that } a^\omega || b \neq a^\omega || b \]

**ter Beek & Kleijn in TCS (2007): Infinite Unfair Shuffles and Associativity**

4. shuffling is **associative**: \( u || (v || w) = (u || v) || w \)
Synchronized shuffling: definitions and examples

Let $u, v \in \Delta^\infty$. Then a word $w \in \Delta^\infty$ is a synchronized shuffle ($S$-shuffle) on $\Gamma$ of $u$ and $v$ if

1. either $u = u_1 x_1 \cdots x_{n-1} u_n$ and $v = v_1 x_1 \cdots x_{n-1} v_n$, with $u_i, v_i \in (\Delta \setminus \Gamma)^*$ and $x_i \in \Gamma$ for $1 \leq i \leq n - 1$, and $u_n, v_n \in (\Delta \setminus \Gamma)^\infty$, in which case $w = w_1 x_1 \cdots x_{n-1} w_n$ with $w_i \in u_i \parallel v_i$ for all $1 \leq i \leq n$ (and $w$ is fair whenever $w_n \in (u_n \parallel v_n)$)

2. or $u = u_1 x_1 u_2 x_2 \cdots$ and $v = v_1 x_1 v_2 x_2 \cdots$, with $u_i, v_i \in (\Delta \setminus \Gamma)^*$ and $x_i \in \Gamma$ for all $i \geq 1$, in which case $w = w_1 x_1 w_2 x_2 \cdots$ with $w_i \in u_i \parallel v_i$ for all $i \geq 1$ (and $w$ is fair)

$$u \parallel \Gamma v = \{ w \in \Delta^\infty : w \text{ is an } S \text{-shuffle on } \Gamma \text{ of } u \text{ and } v \}$$

$$u \parallel|\parallel \Gamma v = \{ w \in \Delta^\infty : w \text{ is a fair } S \text{-shuffle on } \Gamma \text{ of } u \text{ and } v \}$$

$$abc \parallel^\{c\} cd = \{abcd\}, \text{ but } abcd \notin abc \parallel cd \text{ and } abc \parallel^\{b,c\} cd = \emptyset$$
A full S-shuffle of two words is an S-shuffle in which all and only the symbols shared by their alphabets act as synchronizing symbols.

Let \( u \in \Delta_1^\omega \), \( v \in \Delta_2^\omega \), and \( w \in (\Delta_1 \cup \Delta_2)^\omega \). Then \( w \) is a full S-shuffle (fS-shuffle) of \( u \) and \( v \) w.r.t. \( \Delta_1 \) and \( \Delta_2 \) if \( w \) is an S-shuffle on \( \Delta_1 \cap \Delta_2 \) of \( u \) and \( v \) (fair if \( w \) is a fair S-shuffle on \( \Delta_1 \cap \Delta_2 \) of \( u \) and \( v \)).

\[
\begin{align*}
\text{公平} u \Delta_1 \text{|||} \Delta_2 v &= \{ w \in (\Delta_1 \cup \Delta_2)^\omega : w \text{ is a fair fS-shuffle of } u \text{ and } v \text{ w.r.t. } \Delta_1 \text{ and } \Delta_2 \} \\
\text{fS} u \Delta_1 \text{|||} \Delta_2 v &= \{ w \in (\Delta_1 \cup \Delta_2)^\omega : w \text{ is an fS-shuffle of } u \text{ and } v \text{ w.r.t. } \Delta_1 \text{ and } \Delta_2 \}
\end{align*}
\]

If \( \Delta = \{ a, b, c, d \} \), then \( abc \Delta \text{|||} \Delta cd = \emptyset \), but

with \( \Delta' = \{ a, b, c \} \) and \( \Delta'' = \{ c, d \} \), \( abc \Delta' \text{|||} \Delta'' cd = abc \{ c \} \text{|||} \{ c \} cd = \{ abcd \} \)

Also, \( a^\omega \Delta \text{|||} \Delta b = a^\omega \Delta \text{|||} \Delta b = \emptyset \), while

\[
\begin{align*}
a^\omega \{ a \} \text{|||} \{ b \} b &= a^\omega \| b = \{ w \in \{ a, b \}^\omega : w \text{ has one occurrence of } b \} \text{ and } \\
a^\omega \{ a \} \text{|||} \{ b \} b &= a^\omega b = (a^\omega \{ a \} \text{|||} \{ b \} b) \cup \{ a^\omega \}
\end{align*}
\]
Relaxed Synchronized Shuffles

A relaxed $S$-shuffle of two words is an $S$-shuffle in which only some of the symbols shared by their alphabets are synchronizing symbols.

Let $u \in \Delta_1^\infty$, $v \in \Delta_2^\infty$ and $w \in (\Delta_1 \cup \Delta_2)^\infty$. Then $w$ is a relaxed $S$-shuffle (rS-shuffle) on $\Gamma$ of $u$ and $v$ w.r.t. $\Delta_1$ and $\Delta_2$ if $w$ is an $S$-shuffle on $\Gamma \cap \Delta_1 \cap \Delta_2$ of $u$ and $v$ (fair if $w$ is a fair $S$-shuffle on $\Gamma \cap \Delta_1 \cap \Delta_2$ of $u$ and $v$).

$$u_{\Delta_1} \parallel_{\Delta_2}^\Gamma v = \{w \in (\Delta_1 \cup \Delta_2)^\infty : w \text{ is a fair rS-shuffle on } \Gamma \text{ of } u \text{ and } v \text{ w.r.t. } \Delta_1 \text{ and } \Delta_2\}$$

$$u_{\Delta_1} \parallel_{\Delta_2}^\Gamma v = \{w \in (\Delta_1 \cup \Delta_2)^\infty : w \text{ is an rS-shuffle on } \Gamma \text{ of } u \text{ and } v \text{ w.r.t. } \Delta_1 \text{ and } \Delta_2\}$$

$$(a \{a\} \parallel \{b\} \{a,b\} \parallel \{b\}, \{a,b\} = \{ab, ba\} \{a,b\} \parallel \{b\}, \{a,b\} = \{aab, aba\})$$

which differs from

$$(b \{a\} \parallel \{b\} \{a,b\} \parallel \{b\}, \{a,b\} = \{ab\}$$
An *arbitrary S-shuffle* of two words is an S-shuffle in which synchronization on occurrences of the specified synchronizing symbols is only an option.

Let \( u, v \in \Delta^\infty \). Then a word \( w \in \Delta^\infty \) is an *arbitrary S-shuffle* (aS-shuffle) on \( \Gamma \) of \( u \) and \( v \) if

1. either \( u = u_1 x_1 u_2 x_2 \cdots x_{n-1} u_n \) and \( v = v_1 x_1 v_2 x_2 \cdots x_{n-1} v_n \), with \( u_i, v_i \in \Delta^* \) and \( x_i \in \Gamma \), for \( 1 \leq i \leq n-1 \), and \( u_n, v_n \in \Delta^\infty \), in which case \( w = w_1 x_1 w_2 x_2 \cdots x_{n-1} w_n \) with \( w_i \in u_i \parallel v_i \) for all \( 1 \leq i \leq n \) (and \( w \) is *fair* whenever \( w_n \in (u_n \parallel v_n) \))

2. or \( u = u_1 x_1 u_2 x_2 \cdots \) and \( v = v_1 x_1 v_2 x_2 \cdots \), with \( u_i, v_i \in \Delta^* \) and \( x_i \in \Gamma \), for all \( i \geq 1 \), in which case \( w = w_1 x_1 w_2 x_2 \cdots \) with \( w_i \in u_i \parallel v_i \) for all \( i \geq 1 \) (and \( w \) is *fair*).

\[
\begin{align*}
\langle u \rangle \parallel \langle v \rangle & = \{ w \in \Delta^\infty : w \text{ is a fair aS-shuffle on } \Gamma \text{ of } u \text{ and } v \} \\
\langle u \rangle \parallel \langle v \rangle & = \{ w \in \Delta^\infty : w \text{ is an aS-shuffle on } \Gamma \text{ of } u \text{ and } v \} 
\end{align*}
\]

\( cab \in ((ca \parallel \{a\} ab) \parallel \{c\} c) \), whereas all the words in \((ca \parallel \{a\}) \parallel (ab \parallel \{c\} c)\) contain two occurrences of \( c \).
Weak Synchronized Shuffles

A weak S-shuffle of two words is an S-shuffle in which synchronization on occurrences of the specified synchronizing symbols is required, if possible. Let \( u, v \in \Delta^\infty \). Then a word \( w \in \Delta^\infty \) is a weak S-shuffle (wS-shuffle) on \( \Gamma \) of \( u \) and \( v \) if

1. either \( u = u_1x_1u_2x_2 \cdots x_{n-1}u_n \) and \( v = v_1x_1v_2x_2 \cdots x_{n-1}v_n \), with \( u_i, v_i \in \Delta^* \), \( \text{alph}(u_i) \cap \text{alph}(v_i) \cap \Gamma = \emptyset \) and \( x_i \in \Gamma \) for \( 1 \leq i \leq n - 1 \), and \( u_n, v_n \in \Delta^\infty \), in which case \( w = w_1x_1w_2x_2 \cdots x_{n-1}w_n \) with \( w_i \in u_i \parallel v_i \) for all \( 1 \leq i \leq n \) (and \( w \) is fair whenever \( w_n \in (u_n \parallel v_n) \))

2. or \( u = u_1x_1u_2x_2 \cdots \) and \( v = v_1x_1v_2x_2 \cdots \), with \( u_i, v_i \in \Delta^* \), \( \text{alph}(u_i) \cap \text{alph}(v_i) \cap \Gamma = \emptyset \) and \( x_i \in \Gamma \), for all \( i \geq 1 \), in which case \( w = w_1x_1w_2x_2 \cdots \) with \( w_i \in u_i \parallel v_i \) for all \( i \geq 1 \) (and \( w \) is fair)

\[
\begin{align*}
\text{(1)} & \quad \parallel_{\Gamma} u \parallel v = \{ w \in \Delta^\infty : w \text{ is a wS-shuffle on } \Gamma \text{ of } u \text{ and } v \} \\
\text{(2)} & \quad \sim_{\Gamma} u \parallel v = \{ w \in \Delta^\infty : w \text{ is a fair wS-shuffle on } \Gamma \text{ of } u \text{ and } v \}
\end{align*}
\]

\[
\begin{align*}
abc \parallel \{c\} cd &= \{abcd\} \\
\omega \parallel \{a\} a &= \omega \parallel \{a\} a = \omega \parallel a = \omega, \text{ but } \omega \parallel \{a\} a &= \omega \parallel \{a\} a = \emptyset
\end{align*}
\]
Our observations on commutativity

**ter Beek & Kleijn in FI (2009):**
Associateativity of Infinite Synchronized Shuffles and Team Automata

1. (fair) S-shuffling is
2. (fair) fS-shuffling is: $u \Delta_1 \parallel \Delta_2 v = v \Delta_2 \parallel \Delta_1 u$ and $u \Delta_1 \parallel \Delta_2 v = v \Delta_2 \parallel \Delta_1 u$
3. (fair) rS-shuffling is: $u \Delta_1 \parallel \Gamma \Delta_2 v = v \Delta_2 \parallel \Gamma \Delta_1 u$ and $u \Delta_1 \parallel \Gamma \Delta_2 v = v \Delta_2 \parallel \Gamma \Delta_1 u$
4. (fair) aS-shuffling is

---

**ter Beek, Martín-Vide & Mitrana in TCS (2005):** Synchronized Shuffles
5. (fair) wS-shuffling is
Our observations on commutativity

**ter Beek & Kleijn in FI (2009):**

**Associativity of Infinite Synchronized Shuffles and Team Automata**

1. (fair) S-shuffling is
2. (fair) fS-shuffling is: \( u \Delta_1 \parallel \Delta_2 \Delta_2 \Delta_1 = v \Delta_2 \parallel \Delta_1 \Delta_1 \Delta_2 \) and \( u \Delta_1 \parallel \Delta_2 \Delta_2 \Delta_1 = v \Delta_2 \parallel \Delta_1 \Delta_1 \Delta_2 \)
3. (fair) rS-shuffling is: \( u \Delta_1 \parallel \Gamma \Delta_2 \Delta_2 \Delta_1 = v \Delta_2 \parallel \Gamma \Delta_1 \Delta_1 \Delta_2 \) and \( u \Delta_1 \parallel \Gamma \Delta_2 \Delta_2 \Delta_1 = v \Delta_2 \parallel \Gamma \Delta_1 \Delta_1 \Delta_2 \)
4. (fair) aS-shuffling is

**ter Beek, Martín-Vide & Mitrana in TCS (2005): Synchronized Shuffles**

5. (fair) wS-shuffling is
Our results on associativity

ter Beek & Kleijn in FI (2009)

1. (fair) S-shuffling is: \( u \ ||| \Gamma (v \ ||| \Gamma w) = (u \ ||| \Gamma v) \ ||| \Gamma w \) and \( u \ ||| \Gamma (v \ ||| \Gamma w) = (u \ ||| \Gamma v) \ ||| \Gamma w \)

2. fair fS-shuffling is: \( u_{\Delta_1} \ ||| \Delta_{2 \cup \Delta_3} (v_{\Delta_2} \ ||| \Delta_3 w) = (u_{\Delta_1} \ ||| \Delta_2 v)_{\Delta_1 \cup \Delta_2} \ ||| \Delta_3 w \)

3. unfair fS-shuffling is \( \text{not} \), but it is in case of prefix-closed languages:
\[
(\{u\} \cup \text{pref}(u))_{\Delta_1} \ ||| \Delta_{2 \cup \Delta_3} (((\{v\} \cup \text{pref}(v))_{\Delta_2} \ ||| \Delta_3 (\{w\} \cup \text{pref}(w)))) = (((\{u\} \cup \text{pref}(u))_{\Delta_1} \ ||| \Delta_2 (\{v\} \cup \text{pref}(v)))_{\Delta_1 \cup \Delta_2} \ ||| \Delta_3 (\{w\} \cup \text{pref}(w)))
\]

4. fair rS-shuffling is: \( u_{\Delta_1} \ ||| \Gamma (v_{\Delta_2} \ ||| \Gamma w) = (u_{\Delta_1} \ ||| \Gamma v)_{\Delta_1 \cup \Delta_2} \ ||| \Gamma w \)

5. unfair rS-shuffling is \( \text{not} \), but it is in case of prefix-closed languages:
\[
(\{u\} \cup \text{pref}(u))_{\Delta_1} \ ||| \Gamma (\{v\} \cup \text{pref}(v)))_{\Delta_2} \ ||| \Gamma (\{w\} \cup \text{pref}(w))) = (((\{u\} \cup \text{pref}(u))_{\Delta_1} \ ||| \Gamma (\{v\} \cup \text{pref}(v)))_{\Delta_1 \cup \Delta_2} \ ||| \Gamma (\{w\} \cup \text{pref}(w)))
\]

6. (fair) aS-shuffling is: \( u \ \wr \wr \wr \Gamma (v \ \wr \wr \wr \Gamma w) = (u \ \wr \wr \wr \Gamma v) \ \wr \wr \wr \Gamma w \) and \( u \ \wr \wr \wr \Gamma (v \ \wr \wr \wr \Gamma w) = (u \ \wr \wr \wr \Gamma v) \ \wr \wr \wr \Gamma w \)

7. (fair) wS-shuffling is \( \text{not} \)
Our results on associativity

**ter Beek & Kleijn in FI (2009)**

1. (fair) S-shuffling is: $u \mathrel{\mid\mid\mid}_w (v \mathrel{\mid\mid\mid}_w) = (u \mathrel{\mid\mid\mid}_v) \mathrel{\mid\mid\mid}_w$ and $u \mathrel{\mid\mid\mid}_w (v \mathrel{\mid\mid\mid}_w) = (u \mathrel{\mid\mid\mid}_v) \mathrel{\mid\mid\mid}_w$

2. fair fS-shuffling is: $u \mathrel{\Delta_1\mid\mid\Delta_2\cup\Delta_3} (v \mathrel{\Delta_2\mid\mid\Delta_3}_w) = (u \mathrel{\Delta_1\mid\mid\Delta_2}_v) \mathrel{\Delta_1\cup\Delta_2\mid\mid\Delta_3}_w$

3. unfair fS-shuffling is *not*, but it is in case of prefix-closed languages:
   $\mathrel{\langle\{u\} \cup \text{pref}(u)\rangle\Delta_1\mid\mid\Delta_2\cup\Delta_3} (\mathrel{\langle\{v\} \cup \text{pref}(v)\rangle\Delta_2\mid\mid\Delta_3}_w (\mathrel{\{w\} \cup \text{pref}(w)})) = (\mathrel{\langle\{u\} \cup \text{pref}(u)\rangle\Delta_1\mid\mid\Delta_2}_v) \mathrel{\Delta_1\cup\Delta_2\mid\mid\Delta_3}_w (\mathrel{\{w\} \cup \text{pref}(w)}))$

4. fair rS-shuffling is: $u \mathrel{\Delta_1\mid\mid\Delta_2\cup\Delta_3} (v \mathrel{\Delta_2\mid\mid\Delta_3}_w) = (u \mathrel{\Delta_1\mid\mid\Delta_2}_v) \mathrel{\Delta_1\cup\Delta_2\mid\mid\Delta_3}_w$

5. unfair rS-shuffling is *not*, but it is in case of prefix-closed languages:
   $\mathrel{\langle\{u\} \cup \text{pref}(u)\rangle\Delta_1\mid\mid\Delta_2\cup\Delta_3} (\mathrel{\langle\{v\} \cup \text{pref}(v)\rangle\Delta_2\mid\mid\Delta_3}_w (\mathrel{\{w\} \cup \text{pref}(w)})) = (\mathrel{\langle\{u\} \cup \text{pref}(u)\rangle\Delta_1\mid\mid\Delta_2}_v) \mathrel{\Delta_1\cup\Delta_2\mid\mid\Delta_3}_w (\mathrel{\{w\} \cup \text{pref}(w)}))$

6. (fair) aS-shuffling is: $u \mathrel{\mathrel{\Gamma\mid\mid\Gamma\mid\mid\Gamma}_w} (v \mathrel{\mathrel{\Gamma\mid\mid\Gamma\mid\mid\Gamma}_w}) = (u \mathrel{\mathrel{\Gamma\mid\mid\Gamma\mid\mid\Gamma}_v}) \mathrel{\mathrel{\Gamma\mid\mid\Gamma\mid\mid\Gamma}_w}$ and $u \mathrel{\Gamma\mid\mid\Gamma}_w (v \mathrel{\Gamma\mid\mid\Gamma}_w) = (u \mathrel{\Gamma\mid\mid\Gamma}_v) \mathrel{\Gamma\mid\mid\Gamma}_w$

**ter Beek, Martín-Vide & Mitrana in TCS (2005)**

7. (fair) wS-shuffling is *not*
Recall: a team automaton over $S$ is said to satisfy \emph{compositionality} if its behaviour can be described in terms of that of its component automata:

There exists a set-theoretic operation that, if applied to the languages of the component automata in $S$, yields the language of the team automaton

\begin{quote}
\textbf{M.H. ter Beek, Gadducci & Janssens in ENTCS (2007):}
A calculus for team automata

Let $n = 2$ and let $T$ be the $R^{si}$-team automaton over $S$.
Then there exists no set-theoretic operation $\mid$ on languages such that

$$B_{T}^{\Sigma} = B_{C_{1}}^{\Sigma_{1}} \mid B_{C_{2}}^{\Sigma_{2}}$$
\end{quote}
Recall: a team automaton over $S$ is said to satisfy *compositionality* if its behaviour can be described in terms of that of its component automata:

There exists a set-theoretic operation that, if applied to the languages of the component automata in $S$, yields the language of the team automaton

---

Let $n = 2$ and let $\mathcal{T}$ be the $\mathcal{R}^{si}$-team automaton over $S$. Then there exists no set-theoretic operation $|\cdot|$ on languages such that

$$B^\Sigma_{\mathcal{T}} = B^\Sigma_{C_1} | B^\Sigma_{C_2}$$
Our results on compositionality

Let $T$ be the $R^{\text{free}}$-team automaton over $S$. If $S$ is loop-limited, then

$$B^\Sigma_T = \big\|_{i \in \{1, \ldots, n\}} B^\Sigma_{C_i}$$

Let $T$ be the $R^{\text{ai}}$-team automaton over $S$. Then

$$B^\Sigma_T = \big\| \{\Sigma_i \mid i \in \{1, \ldots, n\}\} B^\Sigma_{C_i}$$

Let $T$ be the $(R^{\text{ai}} \cup R^{\text{free}}_{\Sigma \setminus \Gamma})$-team automaton over $S$. If $S$ is $(\Sigma \setminus \Gamma)$-loop-limited, then

$$B^\Sigma_T = \big\|_{\Gamma} \{\Sigma_i \mid i \in \{1, \ldots, n\}\} B^\Sigma_{C_i}$$

Let $T$ be the $R^{\text{no}}$-team automaton over $S$. Then

$$B^\Sigma_T = \big\ll_{\{i \in \{1, \ldots, n\}\}} B^\Sigma_{C_i}$$
Recall examples

\[ C_1: \]

\[ \begin{array}{c}
q_1 \\
q_1 \xrightarrow{a} q_1' \\
q_1' \xrightarrow{b} q_1 \\
\end{array} \]

\[ C_2: \]

\[ \begin{array}{c}
q_2 \\
q_2 \xrightarrow{a} q_2' \\
q_2' \xrightarrow{b} q_2 \\
\end{array} \]

\[ T_{\text{free}}: \]

\[ \begin{array}{c}
(q_1 \ q_2) \\
(q_1 \ q_2) \xrightarrow{a} (q_1 \ q_2) \\
(q_1' \ q_2') \xrightarrow{b} (q_1' \ q_2') \\
(q_1' \ q_2') \xrightarrow{a} (q_1' \ q_2') \\
(q_1 \ q_2) \xrightarrow{a} (q_1 \ q_2) \\
(q_1' \ q_2') \xrightarrow{a} (q_1' \ q_2') \\
(q_1' \ q_2') \xrightarrow{b} (q_1' \ q_2') \\
\end{array} \]

\[ T_{\text{ai}}: \]

\[ \begin{array}{c}
(q_1 \ q_2) \\
(q_1 \ q_2) \xrightarrow{a} (q_1 \ q_2) \\
(q_1' \ q_2') \xrightarrow{b} (q_1' \ q_2') \\
(q_1' \ q_2') \xrightarrow{a} (q_1' \ q_2') \\
\end{array} \]

\[ \Rightarrow B_{T_{\text{ai}}} = \{ \lambda, a \} = \{ b^n, b^n a \mid n \geq 0 \} \Sigma_1 \parallel \Sigma_2 \{ \lambda, ab^n \mid n \geq 0 \} = B_{C_1 \Sigma_1} \parallel \Sigma_2 B_{C_2} \]
Communication and compatibility

Correct-by-construction systems of systems

A concept of compatibility is needed to establish that components within a system (or a system and its environment) always communicate correctly.

Compatibility

- represents an aspect of successful communication behaviour, a necessary ingredient for the correctness of a distributed system
- compatibility failures in a system model may reveal problems in the design of some component(s) to be repaired before implementation

Carmona & Cortadella @ FMCAD’02: Input/Output Compatibility of Reactive Systems

- define compatibility of two components (in a closed environment) that should engage in a dialogue free from message loss and deadlocks (as these may have severe repercussions on reliability, safety and security)
Communication and compatibility

Correct-by-construction systems of systems

A concept of compatibility is needed to establish that components within a system (or a system and its environment) always communicate correctly.

Compatibility

- represents an aspect of successful communication behaviour, a necessary ingredient for the correctness of a distributed system
- compatibility failures in a system model may reveal problems in the design of some component(s) to be repaired before implementation

Carmona & Cortadella @ FMCAD’02: Input/Output Compatibility of Reactive Systems

- define compatibility of two components (in a closed environment) that should engage in a dialogue free from message loss and deadlocks (as these may have severe repercussions on reliability, safety and security)
Compatibility

**Message loss and deadlock**

- message loss occurs when one component sends a message that cannot be received as input by another component
- deadlock occurs when a component is indefinitely waiting for a message that never arrives

**Carmona & Kleijn in TCS (2013): Compatibility in a multi-component environment**

- compatibility generalized to multi-component systems modelled as team automata, within which communication may take place between more than two components at the same time (e.g. broadcasting)
- emphasis on ai-based synchronizations (i.e. $R^{ai}$-team automata)
Compatibility

Message loss and deadlock
- message loss occurs when one component sends a message that cannot be received as input by another component
- deadlock occurs when a component is indefinitely waiting for a message that never arrives

Carmona & Kleijn in TCS (2013): Compatibility in a multi-component environment
- compatibility generalized to multi-component systems modelled as team automata, within which communication may take place between more than two components at the same time (e.g. broadcasting)
- emphasis on ai-based synchronizations (i.e. $\mathcal{R}^{ai}$-team automata)

A research proposal for investigating other composition strategies
Future work

Generalize definition of compatibility in $\mathcal{R}^ai$-team automata

1. non-communicating progress: internal actions do not alter enabledness
2. receptiveness: every output action that is sent, is received as input (i.e. communication never fails, \textit{input-enabledness} in IOA)
3. deadlock-freeness: whenever a component is waiting for input, then the team automaton cannot ‘terminate’

Study influence of team automata being \textit{state-sharing}

\texttt{!} the occurrence of a synchronization (in general) depends also on the current states of the components \textit{not} involved in the synchronization

Compatibility in team automata with master-slave synchronizations

\texttt{?} how is compatibility affected when slaves are added
\texttt{?} in what way does compatibility depend on the collaboration among slaves