ACTS

Distributed Systems

Timed Systems

Distributed Automata

└ Trace theory

Petri nets

└ Event Structures

Σ alphabet $\approx$ action

Q states

$\rightarrow \ \subseteq Q \times \Sigma \times Q$

Labelled Transition Systems: $(Q, \Sigma, \rightarrow, Q_{in}, Q_F)$

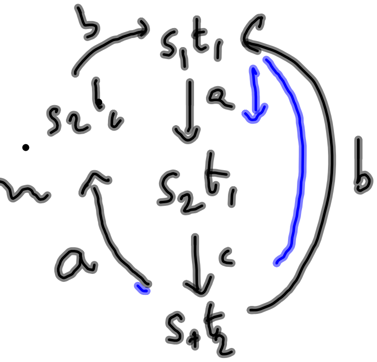
Distributed System

$\boxed{TS_1} \quad \boxed{TS_2} \quad \boxed{TS_3} \quad \ldots \quad \boxed{TS_n}$

$Q_i, \Sigma_i, \ldots$

Coordinate

— synch / communication

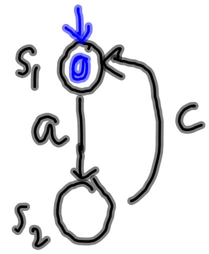


Synch = Common actions

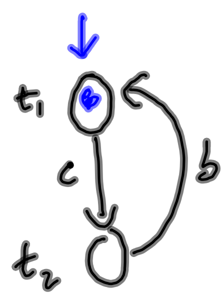
$i \neq j \not\Rightarrow \Sigma_i \cap \Sigma_j = \phi$

$\Sigma_1 = \{a, c\}$ $\Sigma_2 = \{b, c\}$

Common actions together



$(ac)^*$



$(cb)^*$

Direct product

$A_i = (Q_i, \Sigma_i, \rightarrow_i, Q^i_{in}, F_i)$

$\Sigma = \Sigma_1 \cup \Sigma_2 \ldots \cup \Sigma_k$

$a \in \Sigma$, $loc(a) = \{ j \mid a \in \Sigma_j \}$

$Q = Q_1 \times Q_2 \times \ldots \times Q_k$

$\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_k$

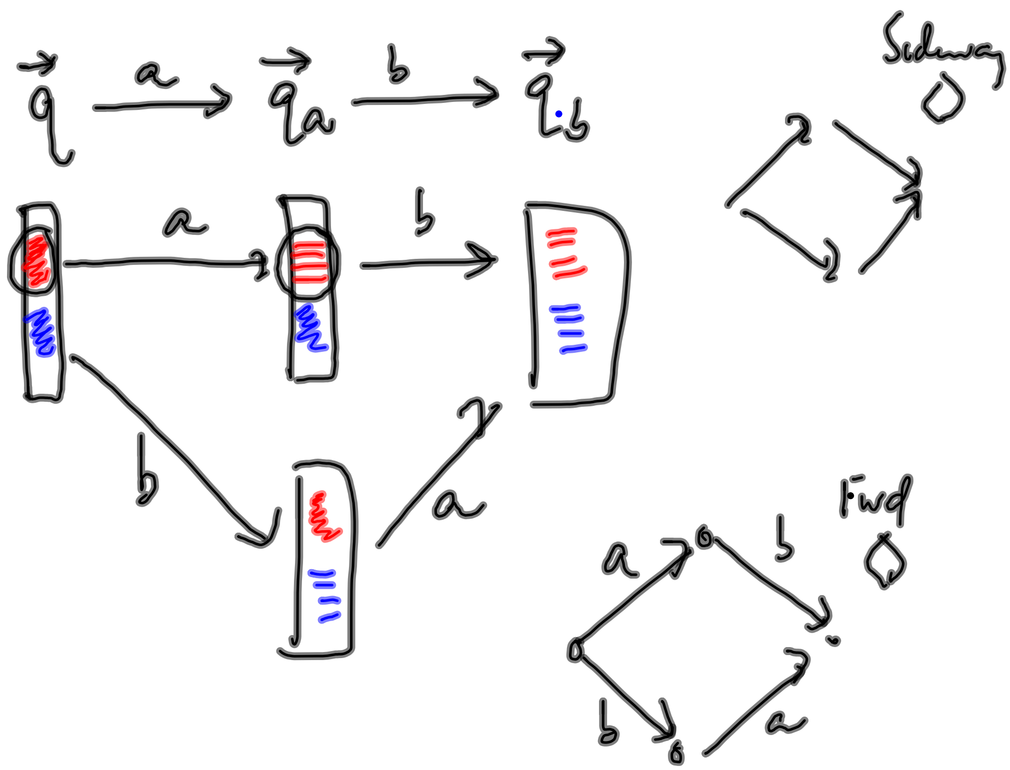
$\langle q_1, \ldots q_k \rangle \xrightarrow{a} \langle q_1', \ldots q_k' \rangle$

$\forall j \in loc(a) \quad q_j \xrightarrow{a}_j q_j'$

$\forall j \notin loc(a) \quad q_j = q_j'$

$Q_{in} = Q^1_{in} \times \ldots \times Q^k_{in}$ $\qquad F = F_1 \times F_2 \times \ldots \times F_k$

Independence: $a\, I\, b$ if $loc(a) \cap loc(b) = \phi$



$\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_k$ $\quad \mathcal{A}_i = (Q_i \ldots, F_i)$

$\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2 \| \cdots \| \mathcal{A}_k$

$L(\mathcal{A}) \subseteq \Sigma^*$

Fix $(\Sigma_1, \ldots, \Sigma_k)$

L is a direct prod / lang if $\exists \mathcal{A} = \mathcal{A}_1 \| \cdots \| \mathcal{A}_k$
s.t. $L = L(\mathcal{A})$

Is every regular lang over Σ a direct product language?

Suppose $a I b$ $\quad uabv \in L(\mathcal{A})$

$$
\begin{tikzcd}
\vec{q}_{in} \arrow[r, "u"] & \vec{q}_u \arrow[r, "a"] \arrow[dr, "b"'] & \vec{q}_{ua} \arrow[r, "b"] & \vec{q}_{uab} \arrow[r, "v"] & \vec{q}_f \\
& & \vec{q}_{ub} \arrow[ur, "a"'] & &
\end{tikzcd}
$$

$\Rightarrow ubav \in L(\mathcal{A})$

Independence-closed regular languages

Synchronized shuffle

$w \in L(A)$

Project w onto Σ_1 – erase letters not in Σ_1

$\hookrightarrow w_1 \in L(A_1)$

* $w \in L(A) \Rightarrow \forall i \;\; w_i = w\restriction_{\Sigma_i} \in L(A_i)$

$\text{shuffle}(L, (\Sigma_1, \ldots, \Sigma_n)) = \{ w \mid \forall i \;\; \exists u \in L \;\; w\restriction_{\Sigma_i} = u\restriction_{\Sigma_i} \}$

<u>not</u> $\exists u \in L \;\; \forall i \;\; w\restriction_{\Sigma_i} = u\restriction_{\Sigma_i}$

$a \, I \, b$ $\quad L = \{ab, aabb\}$

$\Sigma_1 \qquad \Sigma_2$

$abb \in shuffle(L)$ $\qquad abb\restriction_1 = ab\restriction_1$

$\qquad abb\restriction_2 = aabb\restriction_2$

<u>Prop</u> L is a direct prod lang iff $L = shuffle(L)$

<u>Proof</u> $(\Rightarrow)$ $L \subseteq shuffle(L)$ ✓

$shuffle(L) \subseteq L$? $\quad w \in shuffle(L)$ $\quad \forall i \; \exists v \in L$ s.t. $v_i = w_i$

$v \in L \Rightarrow v_i \in L_i \Rightarrow \exists$ run in A_i for v_i

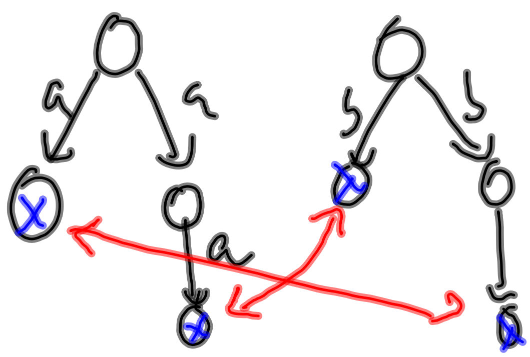
"glue" runs to get run on w

($\Leftarrow$) $L = \text{shuffle}(L)$

$L_i : L\restriction_{\Sigma_i}$ $\exists$ DFA A_i for L_i *

$L(A_1 \| A_2 \| \cdots \| A_k) = L$

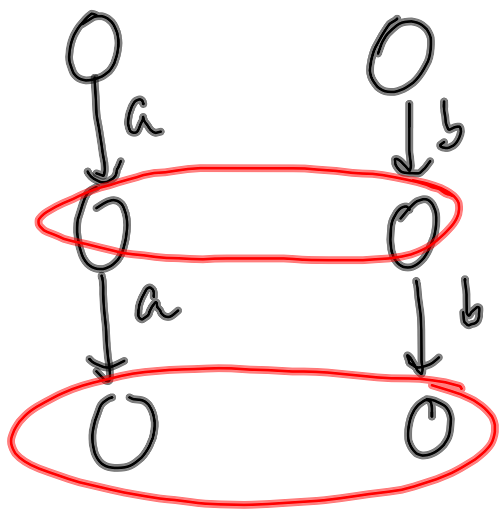
$\therefore$ {ab, ba, aabb, ..., bbaa} is not direct product language.



$\therefore$ Restrict F!

Instead of $F = F_1 \times F_2 \times \ldots \times F_k$

directly specify $F \subseteq Q_1 \times \ldots \times Q_k$ *



Synchronized Product – global final states

Prop L is a synch product lang iff
L is a finite union of direct prod langs.

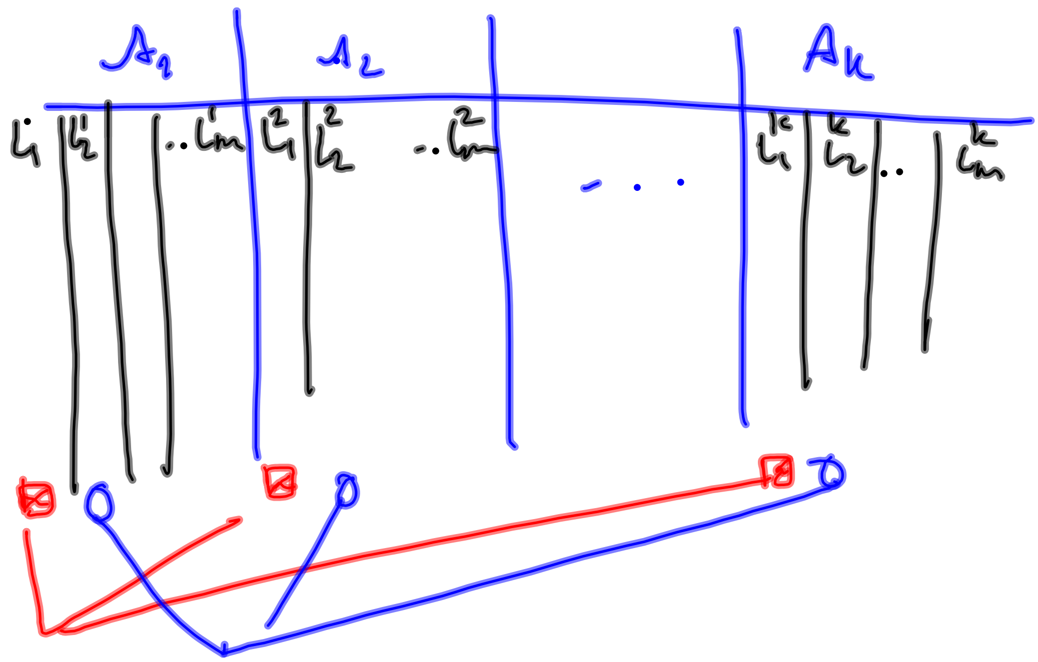
$L = L_1 \cup L_2 \ldots \cup L_m$

Proof: L synch prod $\Rightarrow$ finite union

$\forall f \in F$, create a copy with local acc.
$(f_1, \ldots f_n)$ states $\{f_1\}, \{f_2\}, \ldots, \{f_n\}$

($\Leftarrow$) $L_1, L_2, \ldots, L_m$ are direct product langs

$L_1 \cup L_2 \cup \ldots \cup L_m$ is synch prod ?



Synchronized products $\overset{?}{\neq}$ Direct Products

Choose $F = F_1 \times F_2 \times \cdots \times F_k$

$\{ab, ba\} \cup \{aabb, \ldots, bbaa\}$ $\neq$ direct

Synch products are closed wrt Boolean ops

Union is easy: $L_1 = L_1^1 \cup L_1^2 \cup \cdots \cup L_1^{m_1}$

$L_2 = L_2^1 \cup L_2^2 \cup \cdots \cup L_2^{m_2}$

Closure under complementation

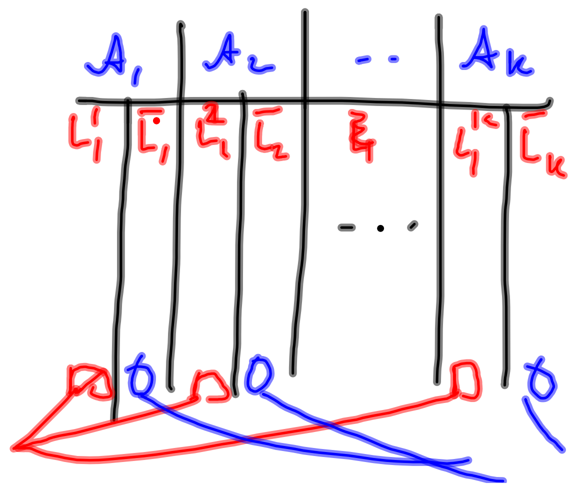
Construct <u>det</u> synch product

Induction on $L = L_1 \cup L_2 \ldots \cup L_m$, decomp into direct product

$L = L_1$ Use direct product proof

$L = L_1 \cup \overbrace{L_2 \ \ldots \ L_m}$

Det Aut q Det Aut $\overline{q}$



$\{a,c\}$ $\{b,c\}$

$$\left[(a|b + aa|bb)\, c\right]^*$$

Suppose $\exists$ synch product. $\exists m$ $L = L_1 \cup L_2 \ldots \cup L_m$

m-block words with ≤ 1 aabbc

$\exists u_i, u_j \in L_p$

abc − 0

aabbc − 1

u_1 0^m

10^{m-1}

010^{m-2}

$\vdots$

v_{m+1} $0^{m-1}1$

$$\left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} m+1$$

✓ $010\ldots0$

$000\ldots10\,0$ ✓

$0\,(aabc)\ldots(abbc)\ldots$

$\downarrow a$

$\downarrow b$

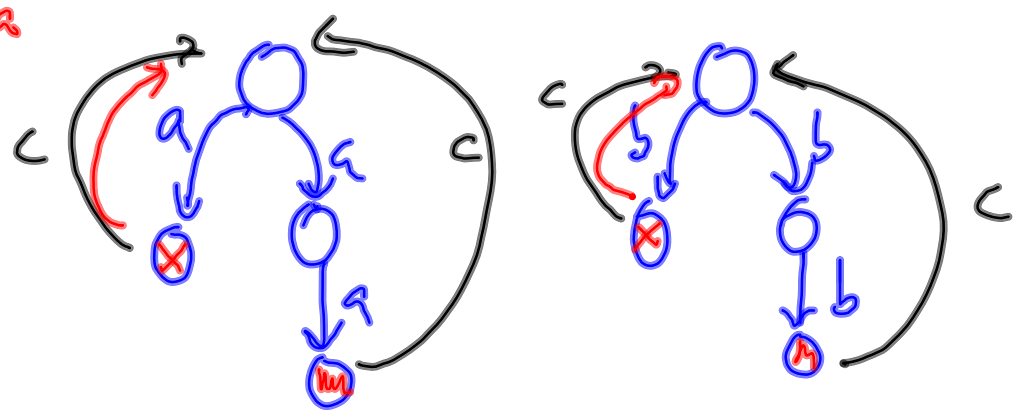
Add "communication" to synchronization

Separate "global" transitions for each $a \in \Sigma$

$\rightarrow_a \subseteq Q_a \times Q_a \qquad Q_a = \prod_{j \in loc(a)} Q_j$

Asynchronous Automaton

Zielonka



Q_i local states, Q_{in}^i

$\forall a: \rightarrow_a \subseteq Q_a \times Q_a$

$F \subseteq Q_1 \times \cdots \times Q_k$

Thm (Zielonka) Every reg. independence closed lang is recognized by an async automaton for any $(\Sigma_1, \ldots, \Sigma_k)$ that matches the independence relation

Independence Alphabet (Mazurkiewicz ~ 1977)

Σ

$I \subseteq \Sigma \times \Sigma$ independence relation

irreflexive, symmetric

$wabv \sim_0 wbav$ if bIa

Define $\sim$ as the transitive closure of $\sim_0$

aIb

$abb \sim_0 bab \sim_0 bba$ $abb \sim bab \sim bba$

- $\sim$ is an equivalence reln

$[w]_I$

$\Uparrow$

equivalence class

trace

Trace language – respects $\sim$

Regular trace lang – regular & $\sim$ closed

Constructing $(\Sigma_1, \ldots, \Sigma_n)$ from (Σ, I)

$a I c$

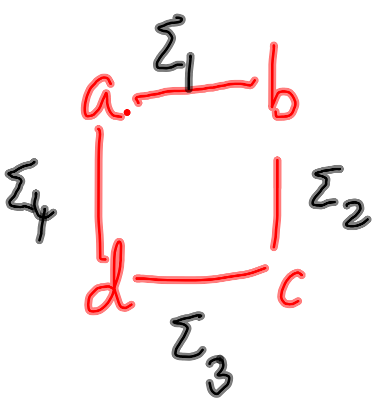
$b I d$

$\{a,b\}, c, d\}$

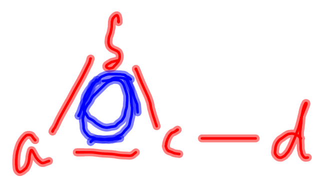
$D = (\Sigma \times \Sigma) - I$

$a D b$ iff $loc(a) \cap loc(b) \neq \emptyset$

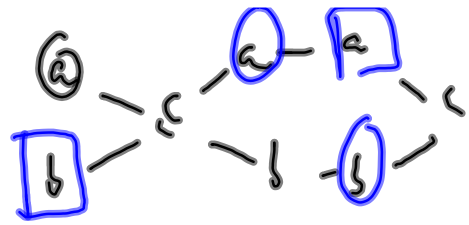
$\exists i \; \{a,b\} \subseteq \Sigma_i$



$\{a,b\}$ $\{b,c\}$ $\{c,d\}$ $\{d,a\}$



$\{a,c\}$ $\{b,c\}$



abcbaabc

trace as labelled partial order $(E, \leq, \lambda)$

$(E, \leq)$ is a p.o. Σ = "events"

$\lambda : E \to \Sigma$

$\lessdot$ $e \lessdot f$ $e \leq f$

$\nexists g \neq e, f$

$e \leq g \leq f$

$e \lessdot f \Rightarrow \lambda(e) \, D \, \lambda(f)$

e, f unordered $\Rightarrow \lambda(e) \, I \, \lambda(f)$