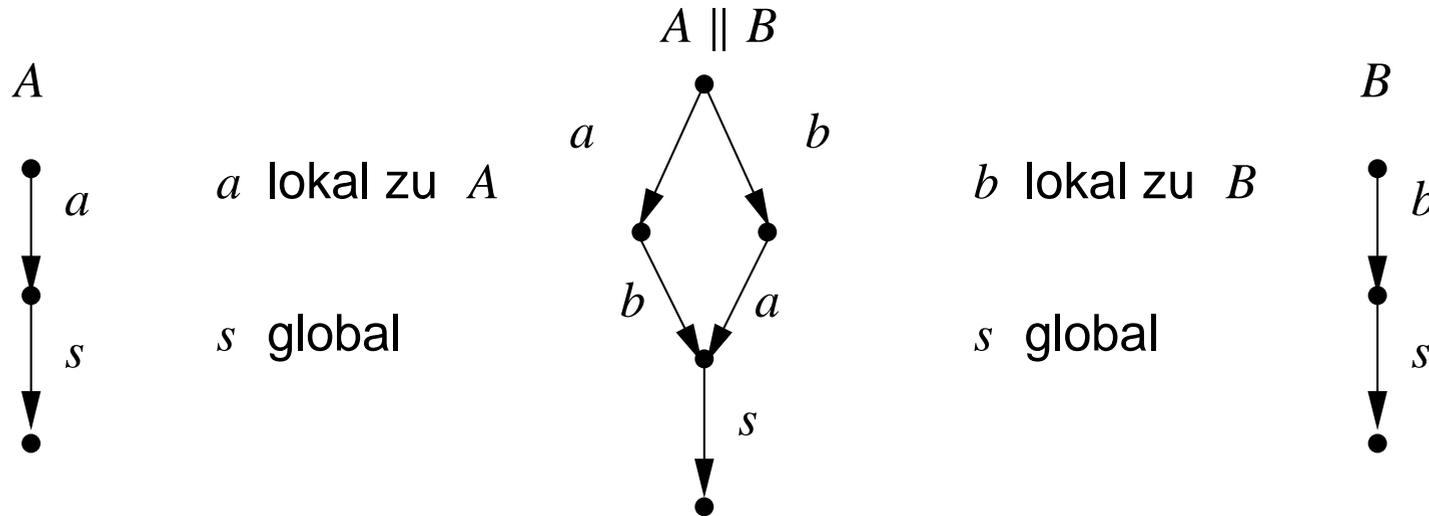


- Synchrone Komposition
 - Gleichschritt aller Komponenten zu einem globalen Takt
 - entspricht der Konstruktion des Produkt-Automaten
 - typische Modellierung von Sequentieller **Hardware**
(obwohl auf Transistor-Ebene eigentlich alles asynchron)
- Asynchrone Komposition
 - Komponenten laufen im Wesentlichen unabhängig nach eigenem Takt
 - typische Modellierung für Kommunikationsprotokolle und Verteilte **Software**
 - Synchronisation: Systemcalls/Interrupts/Signale/Nachrichten/Kanäle/RPC
 - geeignete Vereinfachung: **Interleaving**



- jeder Prozess kommt **abwechselnd** dran mit seinen **lokalen** Aktionen
- auf **globale** Aktionen wird per **Rendezvous** synchronisiert
- Standard-Parallel-Komposition in Prozess-Algebra

Definition Zu zwei LTS A_1 und A_2 besteht die parallele Komposition $A = A_1 \parallel A_2$ aus folgenden Bestandteilen:

$S = S_1 \times S_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$, $I = I_1 \times I_2$, T wird wie folgt definiert:

$s \xrightarrow{a} s'$ in A_1 und $t' = t$ falls $a \in \Sigma \setminus \Sigma_2$

$(s, t) \xrightarrow{a} (s', t')$ in A gdw. $t \xrightarrow{a} t'$ in A_2 und $s' = s$ falls $a \in \Sigma \setminus \Sigma_1$

$s \xrightarrow{a} s'$ in A_1 und $t \xrightarrow{a} t'$ in A_2 falls $a \in \Sigma_1 \cap \Sigma_2$

Interleaving mit Synchronisation auf gemeinsamen Aktionen

Definition In $A_1 \parallel A_2$ ist ein Symbol a **lokal** für A_i gdw. $a \in \Sigma_i$ und $a \notin \Sigma_j$ für alle $i \neq j$. Die Menge der lokalen Symbole für A_i wird mit Λ_i bezeichnet.

Definition Ein Symbol heißt **lokal** falls es lokal für ein A_i ist. Allen lokalen Symbole sind in $\Lambda = \bigcup \Lambda_i$ zusammengefasst.

Definition Übergang $(s_1, s_2) \xrightarrow{a} (s'_1, s'_2)$ in $A_1 \parallel A_2$ ist lokal (für A_i), falls a lokal (für A_i).

Definition **Globale** Symbole bzw. Übergänge sind solche, die nicht lokal sind. Die Menge der globalen Symbole zu A_i wird mit Γ_i bezeichnet und deren Zusammenfassung als $\Gamma = \bigcup \Gamma_i$.

Falls $i = 1$ so sei $\sigma(i) = 2$ und umgekehrt.

Fakt $(s_1, s_2) \xrightarrow{a} (s'_1, s'_2)$ in A gdw. $s_i \xrightarrow{a} s'_i$ in A_i und $s'_{\sigma(i)} = s_{\sigma(i)}$ falls a lokal für A_i
 $s_j \xrightarrow{a} s'_j$ in A_j für alle $j = 1, 2$ falls a global

Fakt Die asynchrone parallele Komposition \parallel ist assoziativ.

(die Schreibweise $A_1 \parallel A_2 \parallel \dots \parallel A_n$ ist also wohldefiniert)

Fakt ... und kommutativ modulo Bisimulation: $A_1 \parallel A_2 \approx A_2 \parallel A_1$

Fakt Für die Übergangsrelation von $A_1 \parallel A_2 \parallel \dots \parallel A_n$ gilt:

Sei $\Psi(a) \subseteq \{1, \dots, n\}$ die Menge der Indizes i mit $a \in \Sigma_i$ und $\overline{\Psi(a)}$ deren Komplement.

$(s_1, \dots, s_n) \xrightarrow{a} (s'_1, \dots, s'_n)$ gdw. $s_i \xrightarrow{a} s'_i$ für alle $i \in \Psi(a) \neq \emptyset$ und $s'_j = s_j$ für alle $j \in \overline{\Psi(a)}$.

Definition Zu zwei LTS A_1 und A_2 besteht die *voll asynchrone* parallele Komposition $A = A_1 ||| A_2$ aus folgenden Bestandteilen:

$S = S_1 \times S_2$, $\Sigma = P(\Sigma_1 \cup \Sigma_2)$, $I = I_1 \times I_2$, T wird wie folgt definiert:

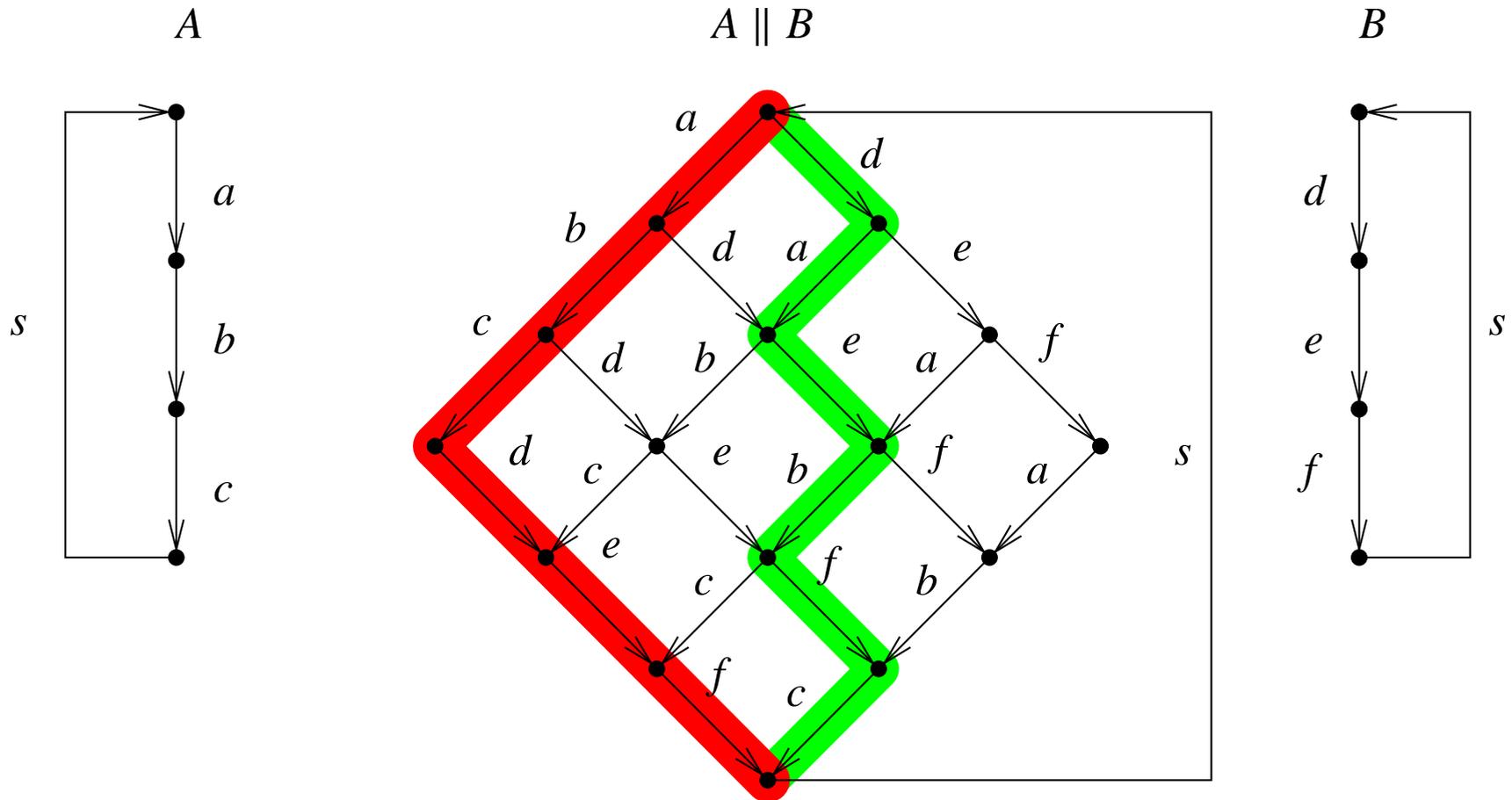
$s \xrightarrow{a} s'$ in A_1 und $t' = t$ falls $M = \{a\} \subseteq \Sigma_1 \setminus \Sigma_2$

$t \xrightarrow{b} t'$ in A_2 und $s' = s$ falls $M = \{b\} \subseteq \Sigma_2 \setminus \Sigma_1$

$(s, t) \xrightarrow{M} (s', t')$ in A gdw. $s \xrightarrow{a} s'$ in A_1 und $t \xrightarrow{a} t'$ in A_2 falls $M = \{a\} \subseteq \Sigma_1 \cap \Sigma_2$

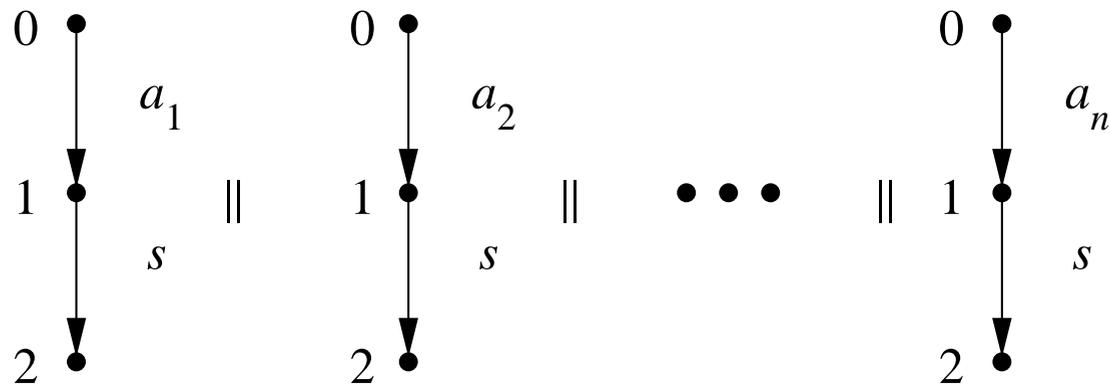
$s \xrightarrow{a} s'$ in A_1 und $t \xrightarrow{b} t'$ in A_2 falls $a \in \Sigma_1 \setminus \Sigma_2$
und $b \in \Sigma_2 \setminus \Sigma_1$
und $M = \{a, b\}$

- Erweiterung der Vollen Asynchronen Komposition auf n LTS:
 - $\Sigma = P(\Sigma_1 \cup \dots \cup \Sigma_n)$, $T = \dots$
(Möglichkeit der gleichzeitigen Synchronisation auf mehrere globale Symbole)
 - insgesamt also exponentielles Aufblasen des Alphabetes!
- Interleaving ist eine Vereinfachung
 - **Fakt** gleiche Menge erreichbarer Zustände
 - Länge von Pfaden zwischen zwei Zuständen kann sich unterscheiden, aber ...
 - ... Modellierung nicht exakt bez. relativer Geschwindigkeit von Komponenten



Idee Verfolge nur einen der 8 möglichen Pfade, z.B. den roten oder den grünen

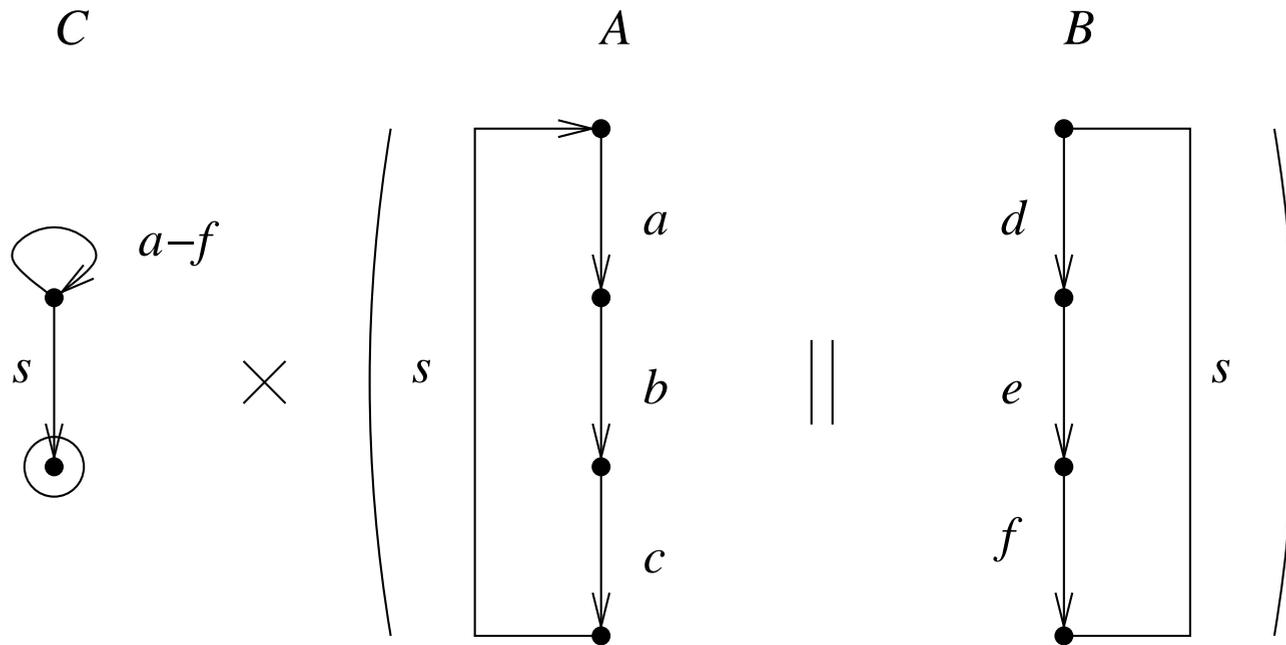
$$\Sigma_i = \{a_i, s\}, \quad \Sigma = \{a_1, \dots, a_n, s\}, \quad \Lambda_i = \{a_i\}, \quad \Gamma = \Gamma_i = \{s\}$$



Anzahl Zustände: $|S| = |\{0, 1, 2\}^n| = 3^n.$

Anzahl erreichbarer Zustände: $|\{0, 1\}^n \cup \{2\}^n| = 2^n + 1$

Anzahl notwendiger Zustände: $|(1^*0^* \cap \{0, 1\}^n) \cup \{2\}^n| = (n + 1) + 1$

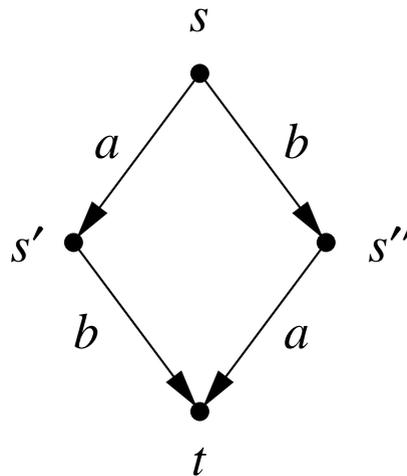


$$C \times (A \parallel B)$$

- Checker muss “invariant” gegenüber weggelassenen Übergängen sein
 - Auslassen von Übergängen darf Erreichen von Finalzuständen nicht verhindern
 - Reduktion ist abhängig vom Checker!
- Maximal viele Übergänge wegzulassen geht nicht:
 - nur möglich wenn man schon die Erreichbarkeit von Finalzuständen kennt
 - damit wäre kein Gewinn möglich!
- Ziel ist möglichst einfaches Kriterium zum Weglassen von Übergängen
 - welches im besten Fall statisch überprüft werden kann
 - oder effizient dynamisch während der Suche

Definition Ein Zustand $s = (s_1, \dots, s_n)$ in $A_1 \parallel \dots \parallel A_n$ heißt **lokal** zu A_i gdw. alle Übergänge in A_i ausgehend von s_i lokal für A_i sind und ein solcher existiert.

Definition Ein Symbol a kommutiert mit dem Symbol b im Zustand s gdw. es für alle s', s'' mit $s \xrightarrow{a} s'$ und $s \xrightarrow{b} s''$ einen Zustand t gibt, mit $s' \xrightarrow{b} t$ und $s'' \xrightarrow{a} t$.



Fakt Sei s lokal zu A_i , dann kommutieren alle Λ_i mit allen andere $\Sigma \setminus \Lambda_i$.

Partial Order Reduction = Partielle Ordnungs-Reduktion

Definition Eine *Expansion* eines Zustandes ist eine Teil-Menge seiner Nachfolger.

BFS/DFS iteriert nur noch über die Expansion von `current` in der inneren Schleife

```
partial_order_recursive_dfs_aux (Stack stack, State current)
{
    ...

    forall next in expansion (current)
        partial_order_recursive_dfs_aux (stack, next);

    ...
}
```

Definition Eine *partielle* Expansion ist eine **echte** Teil-Menge der Nachfolger.

Definition Die *Lokale Partielle Ordnungs-Reduktion* wählt als möglicherweise partielle Expansion eines Zustandes s eine Teil-Menge seiner Nachfolger wie folgt aus:

- genau die lokalen Übergänge von A_i , falls s lokal zu A_i ist
(in diesem Fall hat man eine partielle Expansion)
- bei mehreren solchen A_i wird ein beliebiges i gewählt
- falls kein solches A_i existiert, besteht die Expansion aus allen Nachfolgern

auch “stutter equivalent”

Definition zwei Traces w und w' sind **lokal äquivalent**, geschrieben $w \approx_l w'$, gdw. wenn sie nach Herausstreichen aller lokalen Symbole identisch sind.

Fakt die lokale Äquivalenz \approx_l ist tatsächlich eine Äquivalenzrelation

Beweis

- Reflexivität, Symmetrie folgen unmittelbar
- Schreibweise $w|_{\Sigma'}$ für den Trace w eingeschränkt auf Symbole aus Σ'
- Transitivität: aus $w_1 \approx_l w_2$, $w_2 \approx_l w_3$ folgt $w_1|_{\Gamma} = w_2|_{\Gamma} = w_3|_{\Gamma}$ und somit $w_1 \approx_l w_3$

Definition Checker C ignoriert lokale Symbole gdw.

$$s \xrightarrow{a} = \{s\} \quad \text{in } C \text{ für alle lokalen } a \in \Lambda$$

(auch “ a ist unsichtbar für C ”)

Fakt C ignoriere lokale Symbole und $w \approx_l w'$, dann $w \in L(C) \Leftrightarrow w' \in L(C)$

Beweis Sei $w \in L(C)$. Zustandssequenz, die w akzeptiert, akzeptiert auch w' und umgekehrt.

Suche mit Partieller Ordnungs-Reduktion muss also nur für jede Äquivalenzklasse von \approx_l mindestens einen Repräsentanten ablaufen.

Problem

- eine partielle Expansion “verzögert” Ausführung Übergänge anderer A_j
- diese könnten für immer verzögert werden

Lösung

- Zyklus mit nur partiellen Expansionen muss an einer Stelle voll expandiert werden
- lässt sich leicht in Tiefensuche (DFS) einbauen:
 - jeder Zyklus wird durch “Backedge” zu einem Knoten auf Suchstack geschlossen
 - bei einer “Backedge” wird `current` einfach voll expandiert
- Approximation bei BFS: volle Expansion bei Kanten zu Knoten früherer Generation

- statt Synchronisation durch Aktionen:
 - Synchronisation durch gemeinsame globale Variable
 - und/oder Synchronisation durch Monitore/Semaphoren
 - und/oder Synchronisation durch Nachrichten/Kanäle
- Partielle Ordnungs-Reduktion lässt sich mit folgenden Konzepten anwenden:
 - Unabhängigkeit bzw. Kommutieren von Befehlen
 - * Read/Write und Write/Read abhängig, Read/Read unabhängig
 - * analog für Nachrichten (Read = Receive, Write = Send)
 - Unsichtbarkeit von lokalen Befehlen

- Message Passing
 - Kommunikation durch Nachrichtenaustausch über Kanäle/Puffer/FIFO
 - beachte: endliche Modelle haben immer Kanäle endlicher Kapazität
- Unabhängige oder lokale Operationen (2 Prozesse, 1 Kanal):
 - Lesen von einem **nicht vollem** Kanal
 - Schreiben in einen **nicht leeren** Kanal
- Abhängige oder globale Operationen (2 Prozesse, 1 Kanal):
 - Lesen von einem **vollem** Kanal
 - Schreiben in einen **leeren** Kanal