Process Algebrae
Formal Methods
Lecture 7

Farn Wang
Department of Electrical Engineering
National Taiwan University

What is Process Algebra?

- An algebraic approach to the study of concurrent communicating processes.
- The term "process algebra" was coined in 1982 by Bergstra & Klop and was used to denote an area of science since 1984.
- A process algebra was a structure in the sense of universal algebra that satisfied a particular set of axioms.

The main algebraic approaches to concurrency

- Communicating Sequential Processes (CSP)
  - Hoare (1969, 1978)
- Calculus of Communicating Systems (CCS)
  - Milner (1980)
- Algebra of Communicating Processes (ACP)
  - Bergstra & Klop (1984)

Preliminaries: Equational Specification

- Definition: Equational Specification (Σ, E). Here E is a set of equations of the form $t_1=t_2$ where t_1 and t_2 are terms (axioms) and Σ is the signature.

Example: Natural Number(E_1,Σ_1)

- E_1: axioms

  a(x,0)=x
  a(x,s(y))=s(a(x,y))
  m(x,0)=0
  m(x,s(y))=a(m(x,y),x)

- Σ_1: signature
  - Constant symbol: 0
  - variables: x,y
  - Function symbol: s(successor), a(addition) and m(multiplication)

Term

- Definition:
  - variables x,y… are terms
  - constant symbols 0… are terms
  - if F is a function symbol of arity n, and $t_1,\ldots,t_n$ are terms, then $F(t_1,\ldots,t_n)$ is a term
- open term: a term that contains a variable
- closed term: a term without variables

Σ-algebra

- If A is a Σ-algebra, then the equation $t_1=t_2$ over (Σ,E) has a meaning in A, when we interpret the constant and function symbols in t_1, t_2 by the corresponding constants and function in A
- Abbreviation: A $\models$ E
  - If the Σ-algebra A satisfies all equation $t_1 = t_2$ of E (A is a model of E)

Basic Process Algebra (Σ_{BPA},E_{BPA})

- E_{BPA}(Syntax)
  - x+y = y+x — A1
  - (x+y)+z=x+(y+z) — A2
  - x+x=x — A3
  - (x+y)z=xz+yz — A4
  - (xy)z=x(yz) — A5
  - x+δ — A6
- The operator · is often omitted, thus xy means x · y
- Brackets are also omitted
- · binds stronger than +, thus xy+z means (xy)+z

Basic Process Algebra (Σ_{BPA}, E_{BPA})

- Semantics
  - A1(the commutativity of +)
  - A2(the associativity of +)
  - A3(the idempotency of +)
  - A4(the right distributivity of · over +)
  - A5(the associativity of ·)
  - A6(termination or deadlock)

Basic Process Algebra (Σ_{BPA}, E_{BPA})

- If M $\models (\Sigma_{BPA}, E_{BPA})$, then the elements of its domain are called **processes**
- Example processes
  - δ
  - $x \cdot y \cdot x \cdot \delta$
  - $x \cdot \delta + y \cdot \delta$
  - $x \cdot (y \cdot z \cdot \delta + z \cdot \delta)$

Reasoning in BPA

$$
\begin{aligned}
& (x \cdot y + x \cdot w) \cdot z + x \cdot y \cdot z \\
&= x \cdot y \cdot z + x \cdot w \cdot z + x \cdot y \cdot z && \text{(A4)} \\
&= x \cdot w \cdot z + x \cdot y \cdot z + x \cdot y \cdot z && \text{(A1)} \\
&= x \cdot w \cdot z + (x \cdot y \cdot z + x \cdot y \cdot z) && \text{(A2)} \\
&= x \cdot w \cdot z + x \cdot y \cdot z && \text{(A3)}
\end{aligned}
$$

Example: Coffee Machine

- Consider a simple coffee machine. Coffee costs 40 cents and tea costs 30 cents. Excess money should be returned.
- Define set of actions: {c10, C, T, d, r10}
- First coffee machine:

  $M_1 = c10 \cdot c10 \cdot c10 \cdot c10 \cdot C \cdot d \cdot \delta + c10 \cdot c10 \cdot c10 \cdot T \cdot d \cdot \delta$
- Another coffee machine:

  $M_2 = C \cdot c10 \cdot c10 \cdot c10 \cdot c10 \cdot d \cdot \delta + T \cdot c10 \cdot c10 \cdot c10 \cdot d \cdot \delta$

Example: Coffee Machine (cont.)

- A better coffee machine

  $M_3 = c10 \cdot c10 \cdot c10 \cdot (T \cdot d \cdot \delta + c10 \cdot (T \cdot d \cdot r10\ \delta + C \cdot d \cdot \delta))$

- What does it do?
  - insert three 10 cent coins (c10)
  - choose between tea (T) or another 10 cent coin
  - choose between tea or coffee (C)
  - dispense drink (d)
  - return excess money

From processes to transition systems

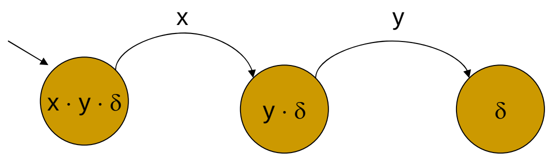
- $x \xrightarrow{a} x'$ means x can execute a and become x'
- Derivation rules for transitions

$$\frac{}{a \cdot x \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$$

- Constructing process graphs:
  - Take process terms as states
  - Add an edge from x to x' with label a if $x \xrightarrow{a} x'$ can be derived from the transition rules.

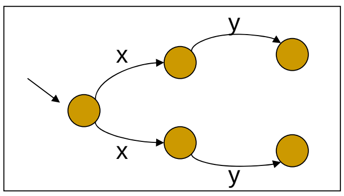
Example: Deriving transitions

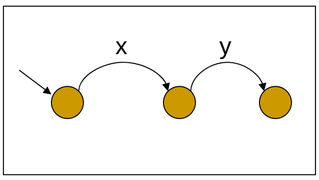
- Constructing a process graph for $x \cdot y \cdot \delta$
  - Derive transition $x \cdot y \cdot \delta \xrightarrow{x} y \cdot \delta$
  - Derive transition $y \cdot \delta \xrightarrow{y} \delta$
  - Define states: $x \cdot y \cdot \delta$, $y \cdot \delta$ and δ



Equality on process graphs

- Equality on process graphs should match equality on processes as defined by the axioms

So 

and 

Are equal, because $x \cdot y \cdot \delta + x \cdot y \cdot \delta = x \cdot y \cdot \delta$ (A3)

Bisimulation between G_1 and G_2

- Let $N= N_1 \cup N_2$
- A relation $R : N_1 \times N_2$ is a bisumulation if
  If (m,n) in R then
  1. If $m \xrightarrow{a} m'$ then $\exists n':n \xrightarrow{a} n'$
     and (m',n') in R
  2. If $n \xrightarrow{a} n'$ then $\exists m':m \xrightarrow{a} m'$
     and (m',n') in R.

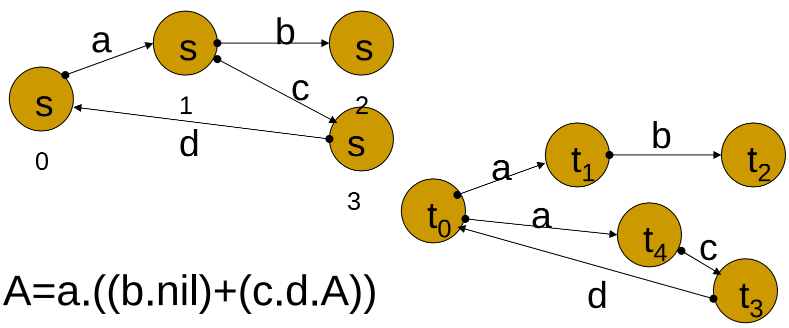
Algorithm for bisimulation:

- Partition N into blocks $B_1 \cup B_2 \cup \ldots \cup B_n = N$.
- Initially: one block, containing all of N.
- Repeat until no change:
  Choose a block B_i and a letter a.
  If some of the transitions of B_i move to some block B_j and some not, partition B_i accordingly.
- At the end: Structures bisimilar if initial states of two structures are in same blocks.

Correctness of algorithm

- Invariant: if (m,n) in R then m and n remain in the same block throughout the algorithm.
- Termination: can split only a finite number of times.

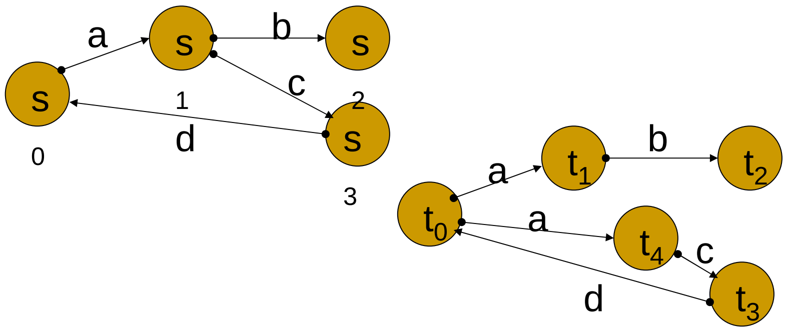
Example:



A=a.((b.nil)+(c.d.A))

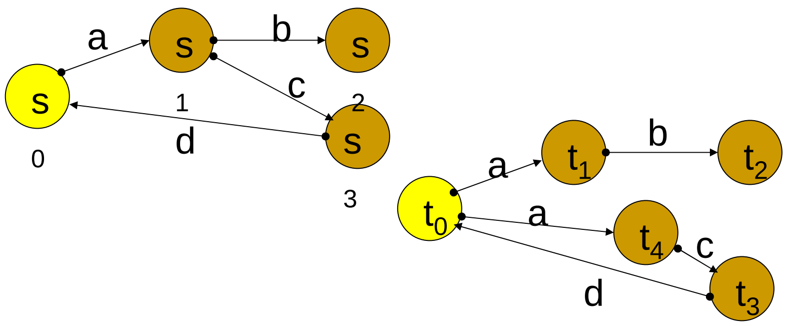
B=(a.(b.nil))+(a.c.d.B)

Example:



$\{s_0,s_1,s_2,s_3,t_0,t_1,t_2,t_3,t_4\}$

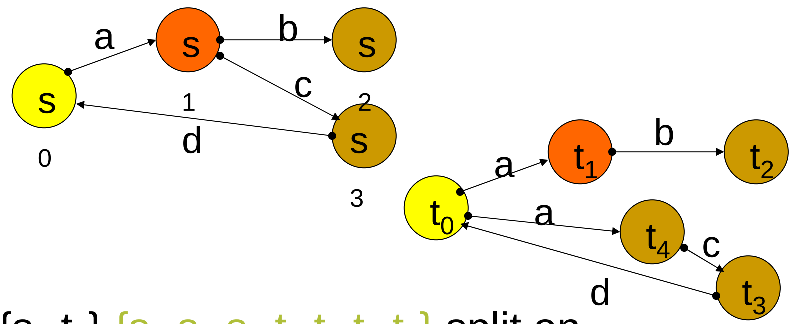
Example:



$\{s_0,s_1,s_2,s_3,t_0,t_1,t_2,t_3,t_4\}$ split on a

$\{s_0,t_0\},\{s_1,s_2,s_3,t_1,t_2,t_3,t_4\}$

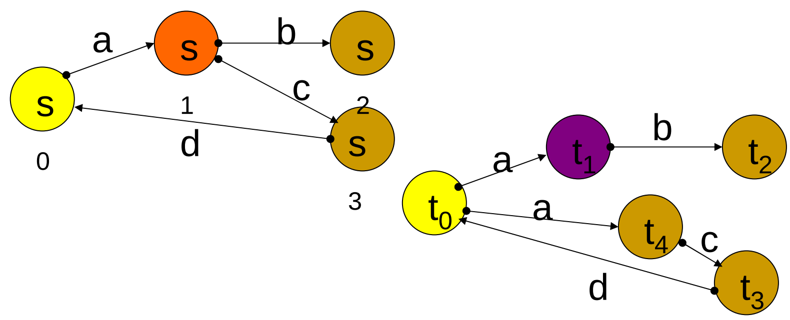
Example:



$\{s_0,t_0\},\{s_1,s_2,s_3,t_1,t_2,t_3,t_4\}$ split on *b*

$\{s_0,t_0\},\{s_1,t_1\},\{s_0,s_2,s_3,t_2,t_3,t_4\}$

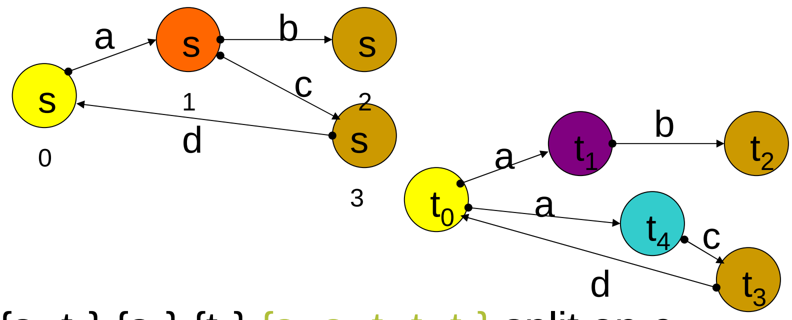
Example:



$\{s_0,t_0\},\{s_1,t_1\},\{s_2,s_3,t_2,t_3,t_4\}$ split on *c*

$\{s_0,t_0\},\{s_1\},\{t_1\},\{s_2,s_3,t_2,t_3,t_4\}$

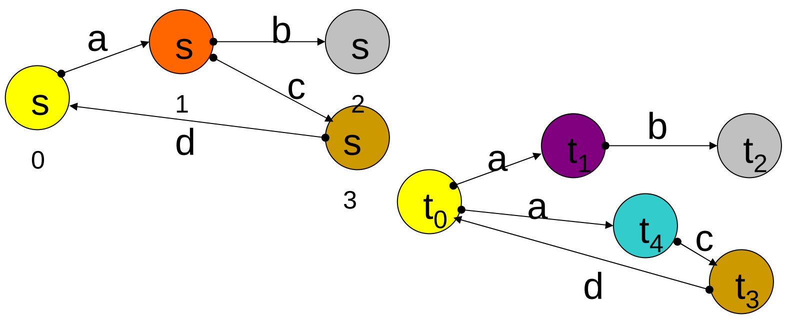
Example:



$\{s_0,t_0\},\{s_1\},\{t_1\},\{s_2,s_3,t_2,t_3,t_4\}$ split on *c*

$\{s_0,t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_2,s_3,t_2,t_3\}$

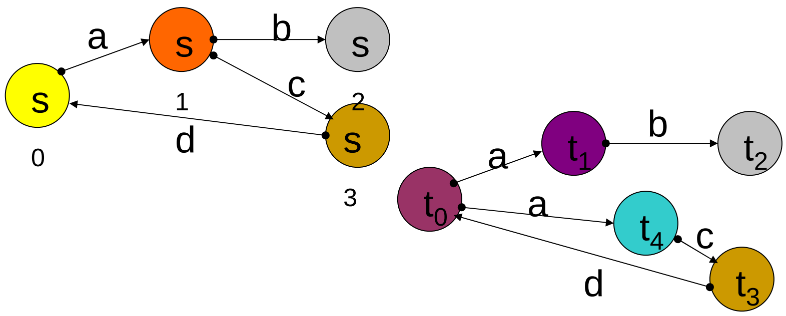
Example:



$\{s_0,t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_2,s_3,t_2,t_3\}$ split on *d*

$\{s_0,t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_3, t_3\},\{s_2,t_2\}$

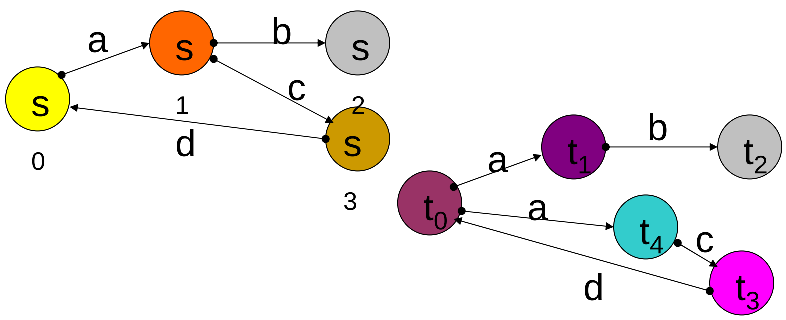
Example:



$\{s_0,t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_2,t_2\},\{s_3,t_3\}$ split on *a*

$\{s_0\},\{t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_3, t_3\},\{s_2,t_2\}$

Example:



$\{s_0\},\{t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_2,s_3,t_2,t_3\}$ split on *d*

$\{s_0\},\{t_0\},\{s_1\},\{t_1\},\{t_4\},\{s_3\},\{t_3\},\{s_2,t_2\}$

Extending BPA

- To make BPA useful for real applications, it has been extended with:
  - Successful termination: $x \cdot \delta$ vs $x \cdot \varepsilon$
  - Sequential processes: $(x \cdot \varepsilon + y \cdot \varepsilon) \cdot (y \cdot \delta + x \cdot \delta)$
  - Recursion: $M = x \cdot (y \cdot M + x \cdot \varepsilon)$ or $M = x \cdot (y \cdot M + x \cdot M)$
  - Parameterization: $M(n) = x \cdot (y \cdot M(n) + x \cdot M(n + 1))$
- The resulting theory is called Theory of Sequential Processes (TSP)

Coffee Machine Example

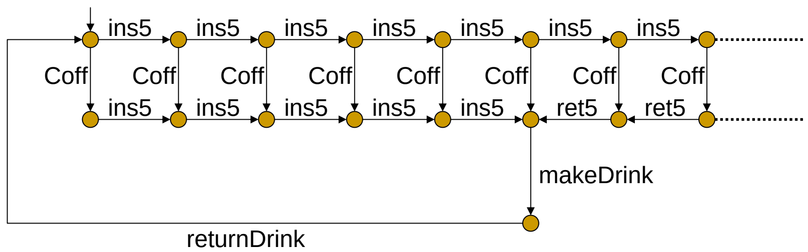
- A coffee machine that makes coffee and chocolate. It can breakdown unexpectedly. Coffee costs 25 cent, chocolate costs 20 cent. The user can insert coins (5 cent, 10 cent, and 20 cent) and make choices in any order. If a choice is made and there is enough money inserted, the drink is offered and change is returned.

How to describe this system?

Coffee Machine Example

- Simplify:
  - Ignore user
  - Ignore coins of 10 and 20 cent
  - Ignore chocolate
  - No breakdown
- Actions:
  - ins5: machine accepts 5 cent coin
  - Coff: machine selects coffee
  - makeDrink: machine prepares selected drink
  - returnDrink: machine offers drink
  - ret5: machine returns 5 cent coin

Coffee Machine Example



- Problems:
  - Sloppy notation for infinite systems
  - No structuring mechanism

Coffee Machine Example

- Machine represented by $M_{m,c}$:
  - $m \in \{5 \cdot n \mid n$ is a natural number$\}$ denote money (cent)
  - $c \in \{nil, Coff\}$ denotes choice of drink

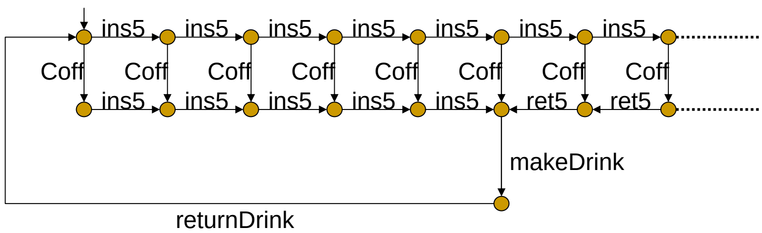
$$M_{0,nil} = ins5 \cdot M_{5,nil} + Coff \cdot M_{0,Coff}$$
$$M_{5,nil} = ins5 \cdot M_{10,nil} + Coff \cdot M_{5,Coff}$$
$$\vdots$$
$$M_{25,Coff} = makeDrink \cdot returnDrink \cdot M_{0,nil}$$
$$M_{30,Coff} = ret5 \cdot M_{25,Coff}$$
$$M_{35,Coff} = ret5 \cdot M_{25,Coff}$$
$$\vdots$$

Coffee Machine Example

- General from:

$$M_{m,c} = \begin{cases} ins5 \cdot M_{m+5,c} + Coff \cdot M_{m,Coff} & \text{if } c \neq Coff \\ ins5 \cdot M_{m+5,c} & \text{if } m<25 \text{ and } c=Coff \\ makeDrink \cdot returnDrink \cdot M_{0,nil} & \text{if } m=25 \text{ and } c=Coff \\ ret5 \cdot M_{m-5,c} & \text{if } m>25 \text{ and } c=Coff \end{cases}$$

- Relation with transition-system:



Coffee Machine Example

- Observations
  - Even the simplified coffee machine is quite complex
  - Structuring mechanism needed
  - Two mathematical domains:
    - transition-systems (possibly infinite)
    - Equations on terms (possibly recursive)
  - Results in one domain should hold in the other as well
  - Transition-systems are more intuitive
  - Equations are better suited for formal reasoning

Extending TSP

- To make TSP more useful for real applications, it has been extended with:
  - Parallelism: $M \parallel M \parallel (x \cdot y \cdot \varepsilon + y \cdot z \cdot \varepsilon)$
  - Communication: $s(3) \cdot x \parallel (r(1) + \ldots + r(10)) \cdot b$
  - Abstraction: $\tau_{\{i_1,i_2\}}\{a \cdot i_1 \cdot i_2 \cdot y \cdot \varepsilon\}$
- We call the resulting theory Algebra of Communicating Processes (ACP)

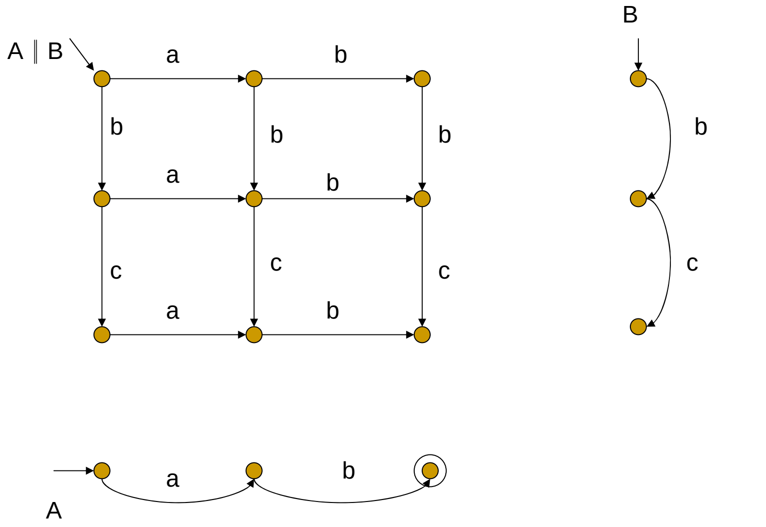
Parallelism: Interleaving Concurrency

- If A and B are processes which cannot communicate, then the parallel composition A $\parallel$ B executes A and B arbitrarily interleaved.
- For example: $A = a \cdot b \cdot \varepsilon$ and $B = b \cdot c \cdot \delta$, then A $\parallel$ B can execute a,b,b,c, or b,a,c,b, or a,b,c,b, etc.

  The $\parallel$ operator can be eliminated:

  $A \parallel B = a \cdot (b \cdot b \cdot c \cdot \delta + b \cdot (b \cdot c \cdot \delta + c \cdot b \cdot \delta)) + b \cdot (a \cdot (b \cdot c \cdot \delta + c \cdot b \cdot \delta) + c \cdot a \cdot b \cdot \delta)$

Process graph of A $\parallel$ B



Parallelism: Interleaving Concurrency and Communication

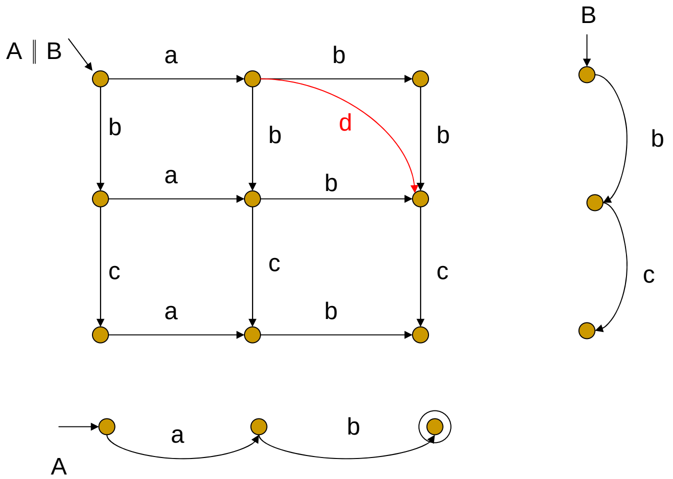
- A and B as before, but now they communicate on action b. The result of a communication is an action d:

  $$\gamma(b,b)=d$$

  Again, the ∥ operator can be eliminated:

  A ∥ B=a·(b·b·c·δ+b·(b·c·δ+c·b·δ)+d·c·δ)+b·(a·(b·c·δ+c·b·δ)+c·a·b·δ)

Process graph of A ∥ B with communication



Enforcing Communication

- By *encapsulating* (disabling) certain actions, communication can be enforced.
- New process operator: $\partial_H()$, with $H \subseteq A$
- Process $\partial_H(x)$ is like x, but cannot execute action $a \in H$

  For example,

  $$\partial_{\{b\}}(A \parallel B) = a \cdot d \cdot c \cdot \delta$$

  The b actions of $A \parallel B$ are encapsulated.

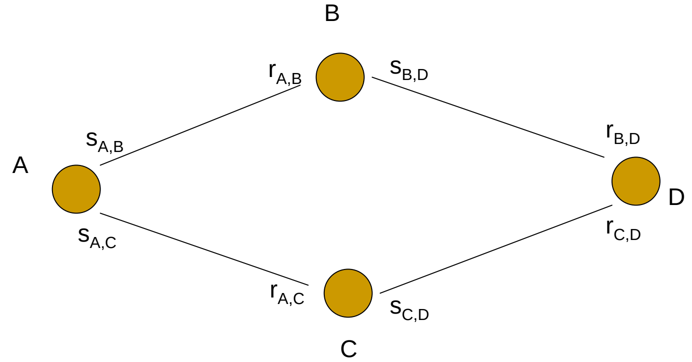
Building Concurrent Systems

- Specify separate components
- Specify the communication actions between these components
- Construct parallel compositions of components
- I Encapsulate certain actions to enforce communication

Example

- 4 sequential components: A, B, C and D. Communication between A and B; A and C; B and D; and C and D.
- Send actions: $s_{X,Y}$ and receive actions $r_{X,Y}$ where X, Y range over A, B, C, D.
- Communication actions: $\gamma(s_{X,Y}, r_{X,Y}) = c_{X,Y}$
- Complete system:
  $S = \partial_H(A \parallel B \parallel C \parallel D)$ where $H=\{s_{X,Y}\} \cup \{r_{X,Y}\}$

A ║ B ║ C ║ D



Abstraction

- Operator: $\tau_H()$, with $H \subseteq A$
- Purpose: hide internal action of components
  - First, all $a \in H$ are renamed into τ
  - Then some τ's are removed using axioms for τ

For example:

$$\tau_{\{b,c\}}(a \cdot b \cdot (c \cdot a \cdot \varepsilon + c \cdot \varepsilon))$$
$$= a \cdot \tau \cdot (\tau \cdot a \cdot \varepsilon + \tau \cdot \varepsilon)$$
$$= a \cdot (\tau \cdot a + \tau \cdot \varepsilon)$$

Only τ's that don't make a choice can be removed!