

Process Algebra CSP – A Technique to Model Concurrent Programs

Hui Shi

January 15, 2002



Contents

- CSP-Processes
- Operational Semantics
 - Transition systems and state machines
 - Bisimulation
 - Firing rules for CSP
 - Model-Checker FDR
- Denotational Semantics
- Specification and Refinement

Semantic approaches to CSP

Operational semantics interprets CSP processes as *transition diagrams*, with visible and invisible actions for moving between various program states.

Denotational semantics maps CSP into some abstract models: *traces*, *failures* and *failures/divergences*.

Algebraic semantics is defined by a set of *algebraic laws*, from which process equivalence between CSP processes can be derived.

Operational Semantics

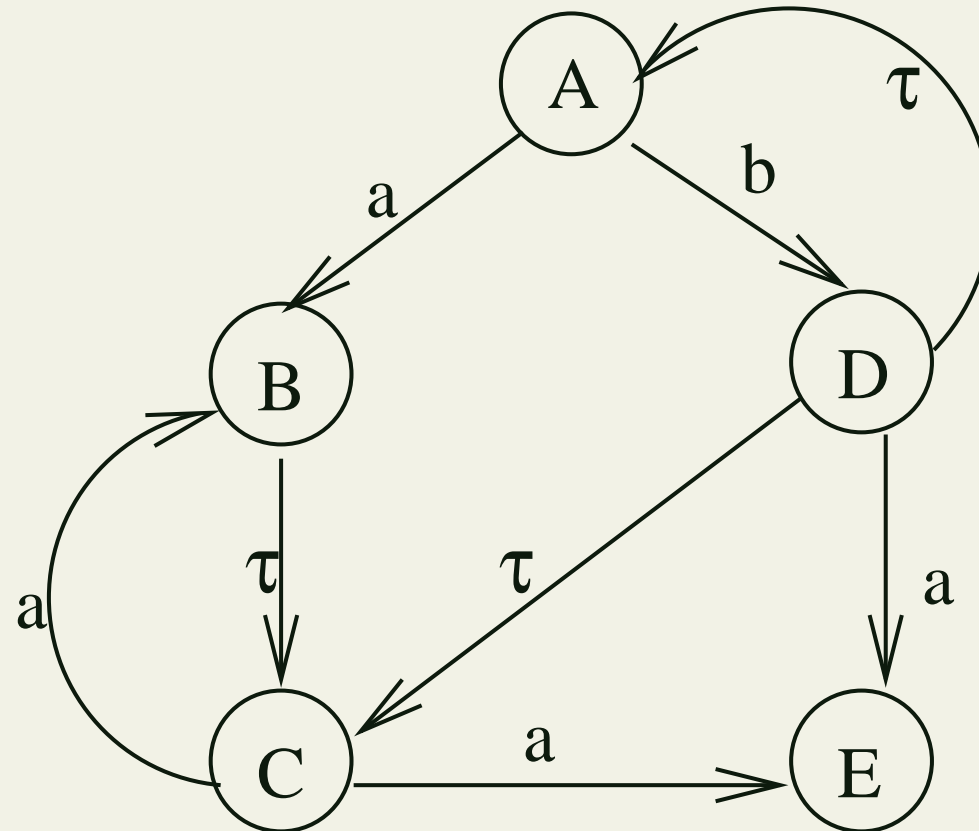
Transition systems and state machines

- Labelled Transition System (LTD)

- a set of nodes as process states
- a starting node n_0
- for each $a \in \Sigma^{\checkmark, \tau}$, a relation \xrightarrow{a} between nodes, where

$$\Sigma^{\checkmark, \tau} = \Sigma \cup \{\checkmark, \tau\}$$

Σ is the alphabet of all communications of a process



Example 1 (A labelled transition system)

- Strong bisimulation

Definition 1 (Strong Bisimulation)

If S is an LTS, the relation R on the set of nodes S' of S is said to be a **strong bisimulation** if, and only if, both the following hold:

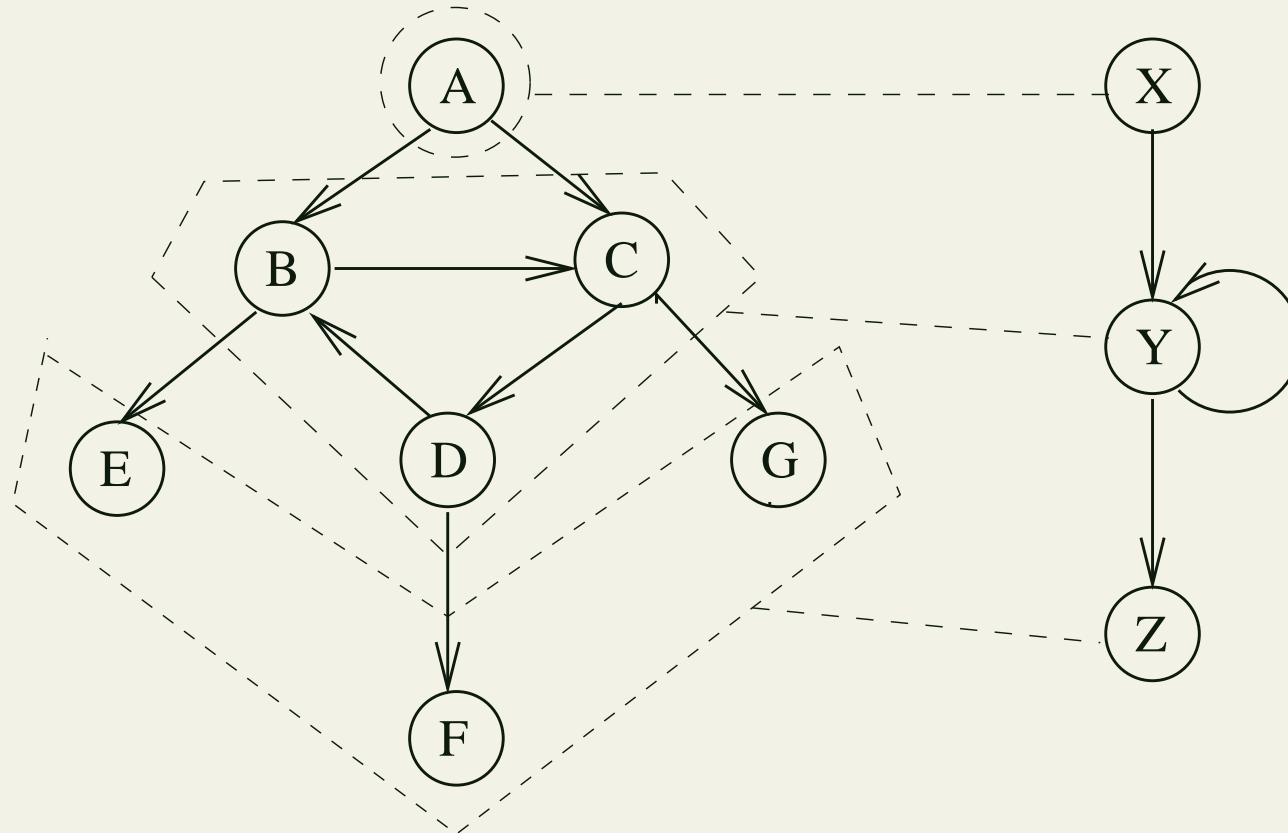
$$\forall n_1, n_2, m_1 \in S'. \forall x \in \Sigma^{\checkmark, \tau}.$$

$$n_1 R n_2 \wedge n_1 \xrightarrow{x} m_1 \Rightarrow \exists m_2 \in S'. n_2 \xrightarrow{x} m_2 \wedge m_1 R m_2$$

$$\forall n_1, n_2, m_2 \in S'. \forall x \in \Sigma^{\checkmark, \tau}.$$

$$n_1 R n_2 \wedge n_2 \xrightarrow{x} m_2 \Rightarrow \exists m_1 \in S'. n_1 \xrightarrow{x} m_1 \wedge m_1 R m_2$$

Two nodes in S' are said to be **bisimilar** if there is any bisimulation which relates them.



Example 2 (Bisimulation equivalence)

- Firing rules for CSP
 - Fundamental operators

$$\frac{}{Skip \xrightarrow{\surd} \Omega} \quad \Omega \text{ denotes any terminated process}$$

$$\frac{}{e \rightarrow P \xrightarrow{a} subs(a, e, P)} \quad (a \in comms(e))$$

where,

$comms(e)$ is the set of communication described by e
 $subs(a, e, P)$ is the result of substituting the part of a
 for each identifier in P bound by e

$$\frac{}{P \sqcap Q \xrightarrow{\tau} P} \quad \frac{}{P \sqcap Q \xrightarrow{\tau} Q}$$

$$\overline{\mu p.P \xrightarrow{\tau} P[\mu p.P/p]}$$

$$\frac{P \xrightarrow{\tau} P'}{P \sqcap Q \xrightarrow{\tau} P' \sqcap Q}$$

$$\frac{Q \xrightarrow{\tau} Q'}{P \sqcap Q \xrightarrow{\tau} P \sqcap Q'}$$

$$\frac{P \xrightarrow{a} P'}{P \sqcap Q \xrightarrow{a} P'} \quad (a \neq \tau)$$

$$\frac{Q \xrightarrow{a} Q'}{P \sqcap Q \xrightarrow{a} Q'} \quad (a \neq \tau)$$

Example 3 ($a \rightarrow P \sqcap b \rightarrow Q$ and $a \rightarrow P \sqcap b \rightarrow Q$)

○ Parallel operators

$$\frac{P \xrightarrow{\tau} P'}{P \parallel_X Q \xrightarrow{\tau} P' \parallel_X Q}$$

$$\frac{Q \xrightarrow{\tau} Q'}{P \parallel_X Q \xrightarrow{\tau} P \parallel_X Q'}$$

$$\frac{P \xrightarrow{a} P'}{P \parallel_X Q \xrightarrow{a} P' \parallel_X Q} \quad (a \in \Sigma \setminus X)$$

$$\frac{Q \xrightarrow{a} Q'}{P \parallel_X Q \xrightarrow{a} P \parallel_X Q'} \quad (a \in \Sigma \setminus X)$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q'}{P \parallel_X Q \xrightarrow{a} P' \parallel_X Q'} \quad (a \in X)$$

$$\frac{P \xrightarrow{\checkmark} P'}{P \parallel_X Q \xrightarrow{\tau} \Omega \parallel_X Q}$$

$$\frac{Q \xrightarrow{\checkmark} Q'}{P \parallel_X Q \xrightarrow{\tau} P \parallel_X \Omega}$$

$$\frac{}{\Omega \parallel_X \Omega \xrightarrow{\checkmark} \Omega}$$

Exercise: Derive the transitions of the process

$$\text{SVAR}(0) \parallel_{\{read, write\}} (\text{USER}(1) \parallel \parallel \text{USER}(2))$$

- Hiding and renaming

$$\frac{P \xrightarrow{x} P'}{P \setminus B \xrightarrow{x} P' \setminus B} \quad (x \notin B \cup \{\checkmark\})$$

$$\frac{P \xrightarrow{\checkmark} P'}{P \setminus B \xrightarrow{\checkmark} \Omega}$$

$$\frac{P \xrightarrow{a} P'}{P \setminus B \xrightarrow{\tau} P' \setminus B} \quad (a \in B)$$

$$\frac{P \xrightarrow{\tau} P'}{P \parallel R \parallel \xrightarrow{\tau} P' \parallel R \parallel}$$

$$\frac{P \xrightarrow{\checkmark} P'}{P \parallel R \parallel \xrightarrow{\checkmark} \Omega}$$

$$\frac{P \xrightarrow{a} P'}{P \parallel R \parallel \xrightarrow{b} P' \parallel R \parallel} \quad (a R b)$$

- Sequential composition

$$\frac{P \xrightarrow{x} P'}{P; Q \xrightarrow{x} P'; Q} \quad (x \neq \checkmark)$$

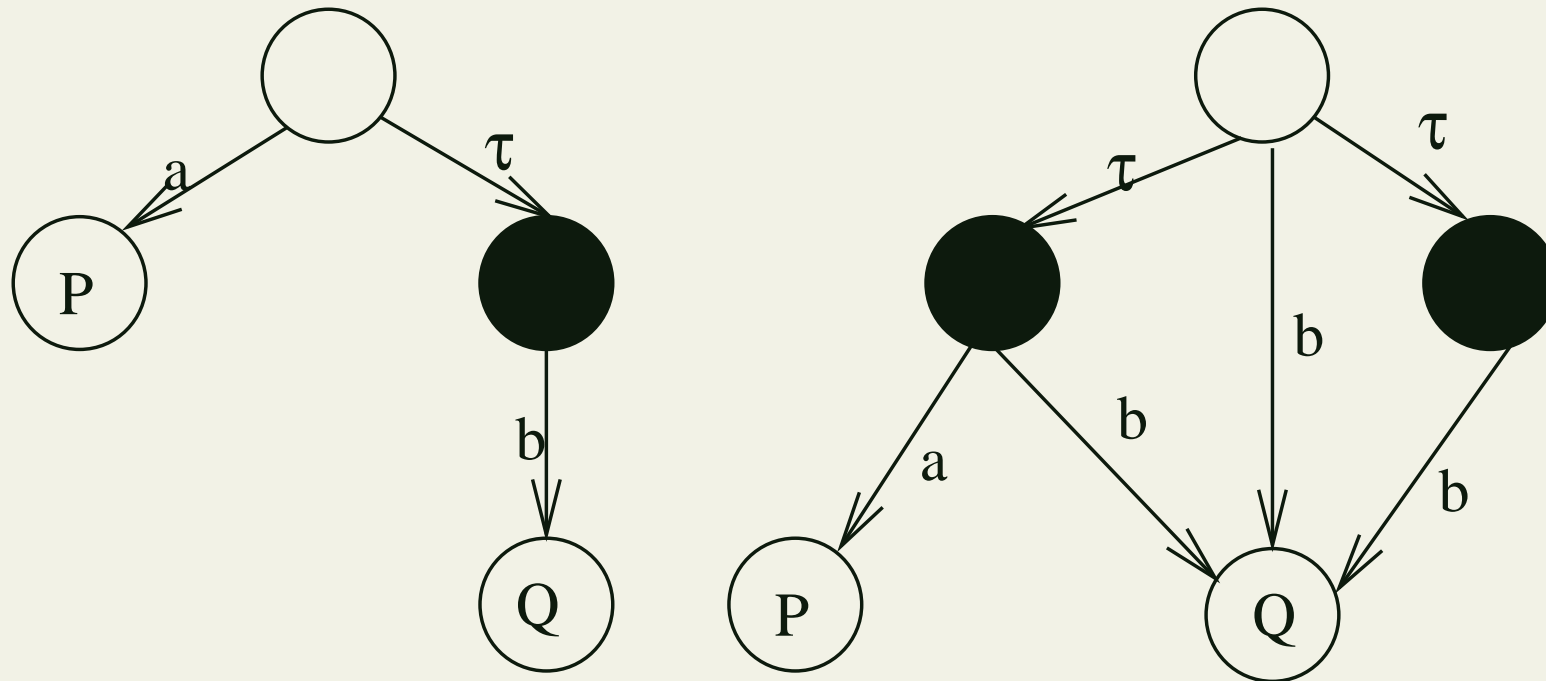
$$\frac{\exists P'. P \xrightarrow{\checkmark} P'}{P; Q \xrightarrow{\tau} Q}$$

- Time-out

$$\frac{P \xrightarrow{\tau} P'}{P \triangleright Q \xrightarrow{\tau} P' \triangleright Q}$$

$$\frac{P \xrightarrow{a} P'}{P \triangleright Q \xrightarrow{a} P'} \quad (a \neq \tau)$$

$$\frac{}{P \triangleright Q \xrightarrow{\tau} Q}$$



Example 4 ($P' \triangleright Q'$ and $(P' \sqcap Stop) \sqsubseteq Q'$)

$$P' = a \rightarrow P$$

$$Q' = b \rightarrow Q$$

Exercise: Derive the transitions of the following process:

$$\text{COPY} \gg \text{COPY} = \\ (\text{COPY} \parallel \text{right}/\text{mid} \parallel \parallel_{\{\text{mid}\}} \text{COPY} \parallel \text{left}/\text{mid} \parallel) \setminus \{\text{mid}\}$$

where

$$\text{COPY} = \mu p. \text{left}?x \rightarrow \text{right}.x \rightarrow p$$

The Model-Checker FDR

- The basic concept of FDR

FDR (Failures-Divergence-Refinement) is a model-checking tool for labelled transition systems with foundations in CSP based mainly on explicit model-checking techniques.

Explicit model-checking techniques: the check proceeds by a recursive induction which fully expands the reachable state-space of the two systems and visits each pair of supposedly corresponding state in turn.

- The basic operation of FDR
 - Calculate the operational semantics using firing rules.
 - Normalise the specification process into a LTS in which all states are semantically distinct based mainly on the strong bisimulation.
 - Check process properties in terms of refinement relations.

Example 5 (Model-checking)

- A concurrent system is *deadlocked* if no component can make any progress, generally because each is waiting for communication with others.
- A concurrent system can *livelock*, when a network communicates infinitely internally without any component communicating externally.

Some examples

- Five dining philosophers (deadlock check)
- Producer-consumer system with hiding (livelock check)
- Buffer (refinement check)