

---

---

# Distributed Models

17 March 2010

Lecture 3

Slide credits: R. H. Mak and P. Veltkamp

---

# Topics for Today

---

- Transition systems
  - Executions
- Algorithms
  - Local
  - Distributed
- Asynchronous model
- Synchronous model

Source: Tel 2

# Labeled transition system

---

## Labeled transition system

### 1. Configurations (system state)

- for each node a local (process) state
- virtual concept: when transitions take positive amount of time, then the system may never be in a stable configuration.

### 2. Labeled transitions

- represent the state changes caused by the execution of an event by one of the nodes of the system.
- event is added as a label to the transition
  - Communications
  - computations

# Transition systems

---

A *transition system* is a triple  $(C, \rightarrow, I)$  where

1.  $C$  is a set of *configurations*
2.  $\rightarrow \subseteq C \times C$  is a binary *transition relation*
  1. We write  $\gamma \rightarrow \delta$  instead of  $(\gamma, \delta) \in \rightarrow$
  2. We write  $\rightsquigarrow$  for the reflexive transitive closure of  $\rightarrow$
3.  $I \subseteq C$  is the subset of *initial configurations*

A configuration  $\gamma$  is *terminal*, when there is no  $\delta$  such that  $\gamma \rightarrow \delta$ . A configuration  $\delta$  is *reachable* from  $\gamma$  when  $\gamma \rightsquigarrow \delta$ .

# Executions

---

Let  $S = (C, \rightarrow, I)$  be a transition system. Let  $C^*$  denote the set of finite sequences of configurations. Let  $C^\omega$  denote the set of infinite sequences of configurations.

A sequence  $s \in C^* \cup C^\omega$  is *maximal*, when it is infinite, or when it is finite and ends in a configuration  $\gamma$  for which there is no configuration  $\delta$  such that  $\gamma \rightarrow \delta$ .

An *execution*  $E$  of  $S$  is a maximal sequence  $(\gamma_0, \gamma_1, \gamma_2, \dots)$  such that  $\gamma_0 \in I$  and  $\gamma_i \rightarrow \gamma_{i+1}$ , for all  $0 \leq i$ .

# So far

---

- Transition systems
  - Executions
- Algorithms
  - Local
  - Distributed
- Asynchronous model
- Synchronous model

# Distributed algorithm

---

---

1. A distributed algorithm for a collection  $\mathbb{P} = \{p_1, \dots, p_n\}$  of processes is a collection  $\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_n\}$  of algorithms, where  $\mathcal{A}_i$  is the local algorithm for  $p_i$ .
2. Let  $\mathcal{M}$  be the set of messages used by algorithm  $\mathcal{A}$ . At any time during the execution of  $\mathcal{A}$  zero or more of messages are in transit, *i.e.* , reside somewhere in the communication network.
3. Since the same message can be sent to several receivers, the state of the communication network is given by a multi-set  $M \in \mathbb{M}(\mathcal{M})$  of messages that are in transit (sent, but not yet received).

# Local Algorithm

---

The local algorithm  $\mathcal{A}$  of a process  $p$  is a quintuple

$$(\mathcal{Z}, \mathcal{I}, \vdash^i, \vdash^s, \vdash^r)$$

where

- $\mathcal{Z}$  is a set of local states
- $\mathcal{I} \subseteq \mathcal{Z}$  is the set of initial states
- $\vdash^i \subseteq \mathcal{Z} \times \mathcal{Z}$  is a relation that defines the internal state transitions
- $\vdash^s \subseteq \mathcal{Z} \times \mathcal{M} \times \mathcal{Z}$  is a relation that defines the state transitions associated with the sending of a message
- $\vdash^r \subseteq \mathcal{Z} \times \mathcal{M} \times \mathcal{Z}$  is a relation that defines the state transitions associated with the receiving of a message

Define  $c \vdash d \iff (c, d) \in \vdash^i \vee \exists m \in \mathcal{M} ((c, m, d) \in \vdash^s \cup \vdash^r)$

# Asynchronous Communication Model

---

---

Let  $\mathbb{P} = \{p_1, \dots, p_N\}$  be a set of processes. Let  $\mathcal{A}_{p_i} = (\mathcal{Z}_{p_i}, \mathcal{I}_{p_i}, \vdash_{p_i}^i, \vdash_{p_i}^s, \vdash_{p_i}^r)$  be the local algorithm of process  $p_i$ , with common message set  $\mathcal{M}$ . Then the asynchronous communication model of the distributed algorithm  $\mathcal{A} = \{\mathcal{A}_{p_1}, \dots, \mathcal{A}_{p_N}\}$  is the transition system  $(\mathcal{C}, \rightarrow, \mathcal{I})$  where

1.  $\mathcal{C} = \mathcal{Z}_{p_1} \times \dots \times \mathcal{Z}_{p_N} \times \mathbb{M}(\mathcal{M}) \wedge \mathcal{I} = \mathcal{I}_{p_1} \times \dots \times \mathcal{I}_{p_N} \times \{\{\}\}$
2.  $\rightarrow$  is the smallest relation such that
  - $(c_{p_i}, c'_{p_i}) \in \vdash_{p_i}^i \implies (\dots, c_{p_i}, \dots, M) \rightarrow (\dots, c'_{p_i}, \dots, M)$
  - $(c_{p_i}, m, c'_{p_i}) \in \vdash_{p_i}^s \implies (\dots, c_{p_i}, \dots, M) \rightarrow (\dots, c'_{p_i}, \dots, M + [m])$
  - $(c_{p_i}, m, c'_{p_i}) \in \vdash_{p_i}^r \implies (\dots, c_{p_i}, \dots, M) \rightarrow (\dots, c'_{p_i}, \dots, M - [m])$

# Synchronous Communication Model

---

---

Let  $\mathbb{P} = \{p_1, \dots, p_N\}$  be a set of processes. Let  $\mathcal{A}_{p_i} = (\mathcal{Z}_{p_i}, \mathcal{I}_{p_i}, \vdash_{p_i}^i, \vdash_{p_i}^s, \vdash_{p_i}^r)$  be the local algorithm of process  $p_i$ , with common message set  $\mathcal{M}$ . Then the synchronous communication model of the distributed algorithm  $\mathcal{A} = \{\mathcal{A}_{p_1}, \dots, \mathcal{A}_{p_N}\}$  is the transition system  $(\mathcal{C}, \rightarrow, \mathcal{I})$  where

1.  $\mathcal{C} = \mathcal{Z}_{p_1} \times \dots \times \mathcal{Z}_{p_N} \wedge \mathcal{I} = \mathcal{I}_{p_1} \times \dots \times \mathcal{I}_{p_N}$

2.  $\rightarrow$  is the smallest relation such that

- $(c_{p_i}, c'_{p_i}) \in \vdash_{p_i}^i \implies (c_{p_1}, \dots, c_{p_i}, \dots, c_{p_N}) \rightarrow (c_{p_1}, \dots, c'_{p_i}, \dots, c_{p_N})$
- $(c_{p_i}, m, c'_{p_i}) \in \vdash_{p_i}^s \wedge (c_{p_j}, m, c'_{p_j}) \in \vdash_{p_j}^r \implies (\dots, c_{p_i}, \dots, c_{p_j}, \dots) \rightarrow (\dots, c'_{p_i}, \dots, c'_{p_j}, \dots)$