Part III

Processes and Concurrency
Outline

1. Motivation
2. Communicating Automata
3. Petri Nets
4. Outlook
Motivation

- So far: only sequential models of computation
- Now: Consider systems of processes with concurrent behaviour
Motivation

- So far: only sequential models of computation
- Now: Consider systems of processes with concurrent behaviour
- Applications:
  - Programming languages with concurrency (e.g., Java’s threads)
  - Operating systems
  - Embedded systems with interacting hardware and software components
  - Web services
Motivation

- So far: only **sequential** models of computation
- Now: Consider systems of **processes** with **concurrent** behaviour

**Applications:**
- Programming languages with concurrency (e.g., Java’s threads)
- Operating systems
- Embedded systems with interacting hardware and software components
- Web services

**Goals:**
- Better understanding of behaviour
- Formal verification of desirable properties (e.g., absence of deadlocks)
- Systematic construction of implementations from (abstract) specifications
1 Motivation
2 Communicating Automata
3 Petri Nets
4 Outlook
Product construction for DFA $\mathcal{A}_1, \mathcal{A}_2$:

$$\mathcal{A} := \langle Q_1 \times Q_2, \Sigma, \delta, (q_0^1, q_0^2), F \rangle$$

is defined by

$$\delta((q_1, q_2), a) := (\delta_1(q_1, a), \delta_2(q_1, a))$$

for every $a \in \Sigma$

and

$$F := F_1 \times F_2$$

recognizes $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ (similar construction for $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$)
Product construction for DFA $\mathcal{A}_1, \mathcal{A}_2$:

$$\mathcal{A} := \langle Q_1 \times Q_2, \Sigma, \delta, (q^1_0, q^2_0), F \rangle$$

is defined by

$$\delta((q_1, q_2), a) := (\delta_1(q_1, a), \delta_2(q_1, a))$$

for every $a \in \Sigma$

and

$$F := F_1 \times F_2$$

recognizes $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ (similar construction for $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$)

**Interpretation:** fully synchronized coupling of two automata
Product construction for DFA $\mathcal{A}_1, \mathcal{A}_2$:

$$\mathcal{A} := \langle Q_1 \times Q_2, \Sigma, \delta, (q_0^1, q_0^2), F \rangle$$

is defined by

$$\delta((q_1, q_2), a) := (\delta_1(q_1, a), \delta_2(q_1, a)) \text{ for every } a \in \Sigma$$

and

$$F := F_1 \times F_2$$

recognizes $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ (similar construction for $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$)

**Interpretation:** fully synchronized coupling of two automata

**Generalization:**

- arbitrary number of automata
- NFA rather than DFA
- no full synchronization, i.e., not every action relevant for every automaton
**Definition III.1**

Let $\mathcal{A}_i = \langle Q_i, \Sigma_i, \Delta_i, q_0^i, F_i \rangle$ be NFA for $1 \leq i \leq n$. The **synchronized product** of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ is the NFA

$$\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n := \langle Q, \Sigma, \Delta, q_0, F \rangle$$

where

- $Q := Q_1 \times \ldots \times Q_n$
- $\Sigma := \Sigma_1 \cup \ldots \cup \Sigma_n$
- $((q_1, \ldots, q_n), a, (q'_1, \ldots, q'_n)) \in \Delta \iff \begin{cases} (q_i, a, q'_i) \in \Delta_i & \text{if } a \in \Sigma_i \\ q'_i = q_i & \text{otherwise} \end{cases}$
- $q_0 := (q_0^1, \ldots, q_0^n)$
- $F := F_1 \times \ldots \times F_n$
Example III.2

Dining Philosophers Problem:

- $n$ philosophers sitting around a table
- a fork between every two of them
- philosophers are thinking, hungry or eating
- need both neighbouring forks to eat
- component automata + product: on the board
Outline

1. Motivation
2. Communicating Automata
3. Petri Nets
4. Outlook
Petri Nets

Definition III.3

A Petri Net is a quadruple

\[ N = \langle P, T, F, m_0 \rangle \]

where

- \( P \) is a non-empty and finite set of places
- \( T \) is a non-empty and finite set of transitions
- \( F \subseteq P \times T \cup T \times P \) is a flow relation
- \( m_0 \) is the initial marking

A marking of \( N \) is a function

\[ m : P \rightarrow \mathbb{N} \]

which assigns a number of tokens to every place. If \( p = \{p_1, \ldots, p_n\} \) we write \( m = (m_1, \ldots, m_n) \) where \( m_i = m(p_i) \) for every \( 1 \leq i \leq n \).
Graphical Representation of Petri Nets

- places as \( \bigcirc \)
- transitions as \( \mid \)
- tokens as \( \bullet \)
- flow relation by arrows

Example III.4
Mutual exclusion protocol (on the board)
Definition III.5
Let $N = \langle P, T, F, m_0 \rangle$ be a Petri Net.

- The **preset** of $t \in T$ is the set
  \[ \bullet t := \{ p \in P \mid (p, t) \in F \}. \]

- The **postset** of $t \in T$ is the set
  \[ t\bullet := \{ p \in P \mid (t, p) \in F \}. \]

- Similarly for places and for sets of transitions or places
- $t \in T$ is **enabled** in $m$ if $m(p) > 0$ for every $p \in \bullet t$. 
Definition III.6 (continued)

- The **firing relation** is defined by:

\[ m \triangleright_t m' \iff t \text{ enabled in } m, \quad m'(p) = \begin{cases} 
  m(p) - 1 & \text{if } p \in \bullet t \setminus t\bullet \\
  m(p) + 1 & \text{if } p \in t\bullet \setminus \bullet t \\
  m(p) & \text{otherwise}
\end{cases} \]

(we then also write \( m \triangleright m' \))

- A marking \( m \neq (0, \ldots, 0) \) is called a **deadlock** if there exists no \( m' \) such that \( m \triangleright m' \).

- A marking \( m' \) is called **reachable** from \( m \) if \( m \triangleright^* m' \).

- \( N \) is called **\( k \)-safe** if for every marking \( m \) reachable from \( m_0 \) and every \( p \in P \), \( m(p) \leq k \).

- \( N \) is called **unsafe** if no such \( k \) exists.
Example III.7

(on the board)

1. Firing of a transition
2. A deadlock
3. A 1-safe Petri Net
4. An unsafe Petri Net
5. A more complicated example
Definition III.8

The **safeness problem** for Petri Nets is specified as follows.

**Input:** Petri Net $N = \langle P, T, F, m_0 \rangle$

**Question:** is $N$ k-safe for some $k \in \mathbb{N}$?
The safeness problem for Petri Nets is specified as follows.

**Input:** Petri Net $N = \langle P, T, F, m_0 \rangle$

**Question:** is $N$ $k$-safe for some $k \in \mathbb{N}$?

**Applications:**
- $N$ safe $\implies$ bounded use of resources (e.g., buffer memory)
- $N$ $k$-safe $\implies$ $N$ representable by finite automaton (at most $(k + 1)^{|P|}$ states reachable)
The Safeness Problem II

Theorem III.9 (Karp, Miller 1968)

The safeness problem for Petri Nets is decidable.
The Safeness Problem II

Theorem III.9 (Karp, Miller 1968)

The safeness problem for Petri Nets is decidable.

Proof.

(idea)

- start with $m_0$
- enumerate all marking reachable from $m_0$
- if $m_0 \triangleright^* m \triangleright^* m'$ where $m' > m$, then $N$ is unsafe
- only finitely many combinations to consider
The reachability problem for Petri Nets is specified as follows.

**Input:** Petri Net $N = \langle P, T, F, m_0 \rangle$, set $M$ of markings

**Question:** does $m_0 \triangleright^* M$ (i.e., $m_0 \triangleright^* m$ for some $m \in M$) hold?
The reachability problem for Petri Nets is specified as follows.

**Input:** Petri Net $N = \langle P, T, F, m_0 \rangle$, set $M$ of markings

**Question:** does $m_0 \triangleright^* M$ (i.e., $m_0 \triangleright^* m$ for some $m \in M$) hold?

**Application:**
- $M :=$ set of “bad” states (e.g., deadlock markings)
- $N$ correct $\iff M$ unreachable
Theorem III.11

The reachability problem for Petri Nets is decidable for finite reachability sets $M$ (even for unbounded nets).

Proof.

omitted
Example III.12

Petri Net representation of Dining Philosophers
($n = 2$; non-atomic picking; on the board)
1 Motivation

2 Communicating Automata

3 Petri Nets

4 Outlook
Outlook

- Communicating automata with FIFO channels
- Petri Nets with weights and capacities
- Petri Nets as language acceptors
- Matrix representation of Petri Nets
- Message Sequence Charts
- Process algebras