Foundations of Informatics: a Bridging Course

Week 3: Formal Languages and Semantics
Part C: Processes and Concurrency

Thomas Noll

Software Modeling and Verification Group (MOVES)

RWTH AACHEN

noll@cs.rwth-aachen.de

http://cosec.bit.uni-bonn.de/students/teaching/10us/10us-bridgingcourse/
http://www-i2.informatik.rwth-aachen.de/i2/b-it10/

b-it, Bonn, Winter semester 2010/11

Outline of Part C

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

Outline of Part C

Reminder

Product construction for DFA $\mathfrak{A}_1, \mathfrak{A}_2$:

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

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

$$F := F_1 \times F_2$$

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

Reminder

Product construction for DFA $\mathfrak{A}_1, \mathfrak{A}_2$:

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

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

$$F := F_1 \times F_2$$

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

Interpretation: fully synchronized parallel execution of two automata

Reminder

Product construction for DFA $\mathfrak{A}_1, \mathfrak{A}_2$:

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

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

$$F := F_1 \times F_2$$

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

Interpretation: fully synchronized parallel execution of two automata

Generalization:

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

Synchronized Product of Automata I

Definition C.1

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

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

where

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

Synchronized Product of Automata II

Example C.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