Models of Computation: Automata and Processes

Paul van Tilburg

(joint work with Jos Baeten, Bas Luttik and Pieter Cuijpers)

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

YR-CONCUR 2009,
September 5, 2009

Introduction

Automata & Formal Language theory

- Parsing, compilers
- Computability, complexity

Automata & Formal Language theory

- Parsing, compilers
- Computability, complexity

- *Back in the days:* different model and real-world computers
- Fixed input string
- Input separated from output
- Batch process
- Abstracts from interaction

Introduction

Automata & Formal Language theory

- Parsing, compilers
- Computability, complexity

- *Back in the days:* different model and real-world computers
- Fixed input string
- Input separated from output
- Batch process
- Abstracts from interaction

- *Nowadays:* one click as input
- Computers are reactive systems
- Interaction much more important

Introduction (2)

Process theory

- Split off, separate development
- Focuses on interaction
- Deals with concurrent setting

Integration

- Attempt reveals differences and similarities
- Use *analogies* to make the integration explicit
- Increase understanding of both theories

Introduction (2)

Process theory

- Split off, separate development
- Focuses on interaction
- Deals with concurrent setting

Integration

- Attempt reveals differences and similarities
- Use *analogies* to make the integration explicit
- Increase understanding of both theories

- *Practical side:* merge in undergraduate curriculum course

Technische Universiteit Eindhoven University of Technology

Overview

Correspondence

- Finite automata, regular languages and processes [FSEN 2009]
- Pushdown automata, processes and context-free languages
  - Pushdown automaton as regular process communicating with a stack [CONCUR 2008]
- Basic parallel processes
  - Parallel pushdown automaton as a regular process communicating with a bag [EXPRESS 2008]
- Computable processes
  - Turing machine as a regular process communicating with two stacks [FSEN 2009]

Overview

Correspondence

- Finite automata, regular languages and processes [FSEN 2009]
- Pushdown automata, processes and context-free languages
  - Pushdown automaton as regular process communicating with a stack [CONCUR 2008]
- Basic parallel processes
  - Parallel pushdown automaton as a regular process communicating with a bag [EXPRESS 2008]
- Computable processes
  - Turing machine as a regular process communicating with two stacks [FSEN 2009]

Other questions

- Relative expressivity
- Decidability

Overview

Correspondence

- Finite automata, regular languages and processes [FSEN 2009]
- Pushdown automata, processes and context-free languages
  - Pushdown automaton as regular process communicating with a stack [CONCUR 2008]
- Basic parallel processes
  - Parallel pushdown automaton as a regular process communicating with a bag [EXPRESS 2008]
- Computable processes
  - Turing machine as a regular process communicating with two stacks [FSEN 2009]

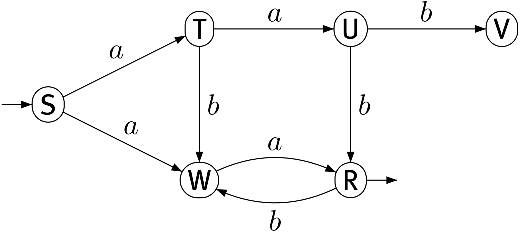
Other questions

- Relative expressivity
- Decidability

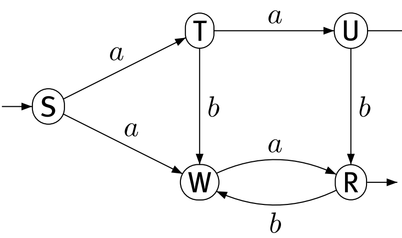
Finite Automaton

- Corresponds to regular language
- No memory!
- Two equivalences: language equivalence and isomorphism

Grammars and Recursive Specifications



Grammars and Recursive Specifications



$$
\begin{aligned}
S &= aT + aW \\
T &= aU + bW \\
U &= bV + bR \\
V &= \mathbf{0} \\
W &= aR \\
R &= bW + \mathbf{1}
\end{aligned}
$$

- From finite automaton to recursive specification

From Recursive Specification to Automaton

$$\frac{}{\mathbf{1}\downarrow} \qquad \frac{}{ax \xrightarrow{a} x}$$

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

$$\frac{t \xrightarrow{a} x \quad P = t}{P \xrightarrow{a} x} \qquad \frac{t\downarrow \quad P = t}{P\downarrow}$$

- Structural Operational Semantics [Plotkin, 1981]

Similarities with Process Algebra

- Finite automaton = finite labelled transition system
- Grammar = recursive specification over $\mathbf{0}, \mathbf{1}, a\_, +, \cdot$ (TSP_τ)
- Regular expression = closed term over $\mathbf{0}, \mathbf{1}, a\_, +, \cdot, {}^*$

Technische Universiteit Eindhoven University of Technology

Similarities with Process Algebra

- Finite automaton = finite labelled transition system
- Grammar = recursive specification over $\mathbf{0}, \mathbf{1}, a\_, +, \cdot$ (TSP_τ)
- Regular expression = closed term over $\mathbf{0}, \mathbf{1}, a\_, +, \cdot, ^*$

Theory of Sequential Processes

- $\mathbf{0}$ inaction, unsuccessful termination, deadlock
- $\mathbf{1}$ empty process, skip, successful termination
- $a\_$ action prefix
- $+$ alternative composition, choice
- $\cdot$ sequential composition

[Baeten, Basten, Reniers, *Process Algebra*, Cambridge UP, 2009]

Bisimulation

- In process theory a difference equivalent is used
- Expose interaction and preserve choices

Definition

We call the largest symmetric relation R such that

- if $p \xrightarrow{a} p'$ then there exists q' such that $q \xrightarrow{a} q'$ and $p' \; R \; q'$
- if $q \xrightarrow{a} q'$ then there exists p' such that $p \xrightarrow{a} p'$ and $p' \; R \; q'$
- if $p\downarrow$ implies $q\downarrow$ and vice versa

the *bisimulation* relation

Notes

- If $(p, q) \in R$, then p and q are *bisimilar* (notation: $p \leftrightarroweq q$)
- Prefer branching bisimulation

Definition

A *regular language* is a language equivalence class of a finite (non-deterministic) automaton

TU/e Technische Universiteit Eindhoven University of Technology

Regular Language & Process

Definition

A *regular language* is a language equivalence class of a finite (non-deterministic) automaton

Definition

A *regular process* is a bisimulation equivalence class of a finite, non-deterministic automaton

Technische Universiteit Eindhoven University of Technology

Regular Language & Process

Definition

A *regular language* is a language equivalence class of a finite (non-deterministic) automaton

Definition

A *regular process* is a bisimulation equivalence class of a finite, non-deterministic automaton

- A regular process is given by a recursive specification over the signature $\mathbf{0}, \mathbf{1}, a\_, +$ $\qquad$ (BSP_τ)

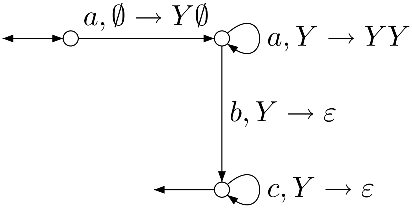
Definition

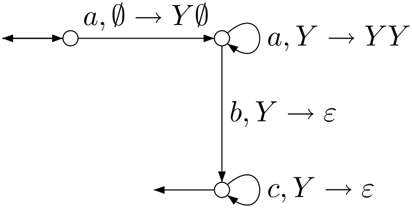
A *regular language* is a language equivalence class of a finite (non-deterministic) automaton

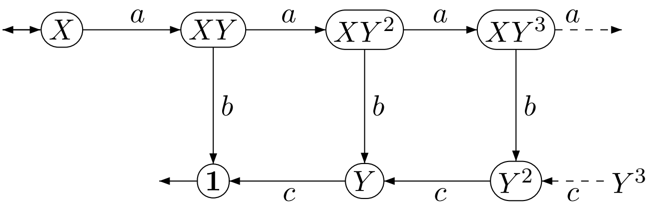
Definition

A *regular process* is a bisimulation equivalence class of a finite, non-deterministic automaton

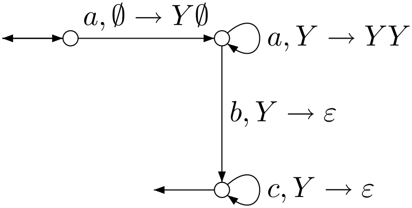
- A regular process is given by a recursive specification over the signature $\mathbf{0}, \mathbf{1}, a\_, +$ $\quad$ (BSP_τ)
- Processes given by deterministic automata, and by regular expressions, form a subclass
  [Baeten, Corradini, Grabmayer, *JACM* 2007]


$a, \emptyset \to Y\emptyset$
$a, Y \to YY$
$b, Y \to \varepsilon$
$c, Y \to \varepsilon$

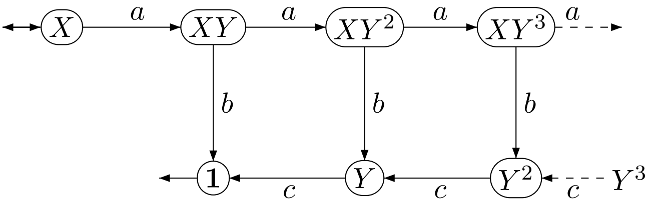




Technische Universiteit Eindhoven University of Technology

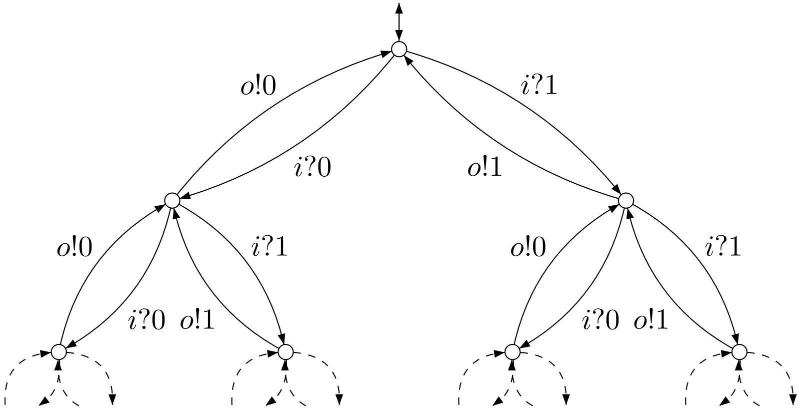


$$X = aX \cdot Y + b\mathbf{1}$$
$$Y = c\mathbf{1}$$

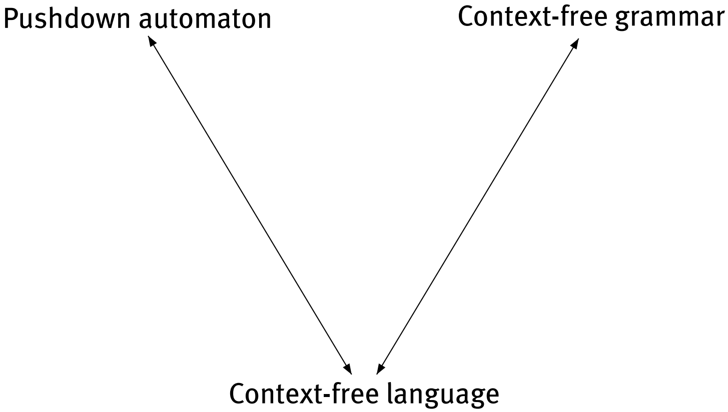


The Stack

$$S = \mathbf{1} + \sum_{d \in \mathcal{D}} i?d.S \cdot o!d.S$$



Pushdown Correspondence



Pushdown Correspondence

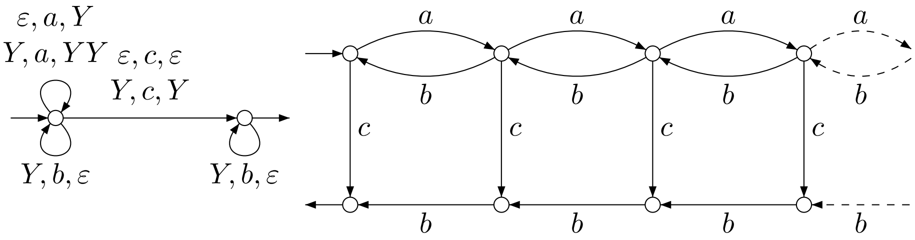
Pushdown automaton ↕ Pushdown process

"Context-free" specification ↕ Pushdown process

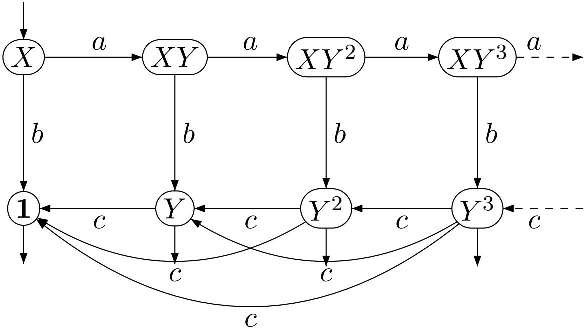
Pushdown process $\underline{\leftrightarrow}_b$ Pushdown process

Problem with Pop-choice

- Context-free languages correspond to language accepted by PDAs
- Not the case with bisimulation! [Moller, 1996]
- *Fix:* do not allow for *pop-choice* (to ensure existence specification)

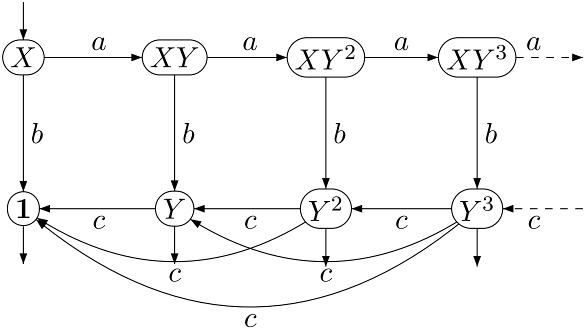


Problem with 1-summands



$$X = aX \cdot Y + b\mathbf{1}$$
$$Y = \mathbf{1} + c\mathbf{1}$$

Problem with 1-summands



$$X = aX \cdot Y + b\mathbf{1}$$
$$Y = \mathbf{1} + c\mathbf{1}$$

- Recursive specifications over TSP_τ can lead to unbounded branching
- *Fix:* transparency-restricted Greibach normal form

Pushdown Correspondence Theorem

Theorem

A process is a pop choice-free pushdown process iff it is definable by a transparency-restricted recursive specification [FSEN, 2009]

Technische Universiteit Eindhoven University of Technology

Pushdown Correspondence Theorem

Theorem

A process is a pop choice-free pushdown process iff it is definable by a transparency-restricted recursive specification [FSEN, 2009]

Notes

- Decidability of bisimulation shown for this class!
- Is it the right correspondence?

Conclusion

- Integration of automata theory and process theory is beneficial for both theories
- Correspondence finite automata, regular languages and processes
- Correspondence pushdown automata, context-free language, pushdown processes

- This integrated theory can be a first-year course in any academic bachelor program in computer science (or related subjects)

Thank you!

Questions?