Models of Computation: Automata and Processes

Paul van Tilburg

Formal Methods Group
Department of Mathematics and Computer Science
Eindhoven University of Technology

IPA Fall Days, 2007
Motivation – "Beyond Turing"

- Automata theory: simple models of computation
  - Understanding the *principles* of computing
  - Analysis of *computability*, complexity
- Process theory: origins in automata theory
  - "No interaction with environment"
  - Focus: notion of *interaction* and parallel behaviour
Motivation – "Beyond Turing"

- Automata theory: simple models of computation
  - Understanding the *principles* of computing
  - Analysis of *computability*, complexity
- Process theory: origins in automata theory
  - “No interaction with environment”
  - Focus: notion of *interaction* and parallel behaviour

1. Goal: *integration* of automata and process theory
   - Attempt reveals differences and similarities
   - Use *analogies* to make the integration explicit

2. Goal: Add process theory to the undergraduate curriculum
Automata accept languages as correct or wanted behaviour:

The above automata accept the same language, they are *language equivalent*:
- a *coin* followed by *coffee*
- a *coin* followed by *tea*
Automata accept languages as correct or wanted behaviour:

The above automata accept the same language, they are *language equivalent*:

- a **coin** followed by **coffee**
- a **coin** followed by **tea**

Process theory differentiates using the *bisimulation equivalence*
Regular Expressions and Process Terms

- Regular expressions describe languages:
  
  \[ \text{coin} \cdot \text{coffee} + \text{coin} \cdot \text{tea}, \quad \text{coin} \cdot (\text{coffee} + \text{tea}) \]

- Regular expressions can describe all regular languages

- Their process term counterparts cannot!

- Process terms have calculation rules (axioms). E.g.:
  
  \[
  \begin{align*}
  (A3) \quad x + x &= x \\
  (A4) \quad (x + y)z &= xz + yz
  \end{align*}
  \]

- Process theory: additional operators (\(\parallel\), |, and \(\llfloor\)) for describing parallel behaviour which are not present in automata theory.
Grammars and Recursive Specifications

The context-free process $S$:

- Grammars can also describe formal languages
- Right-linear grammars are equivalent to recursive specifications
The context-free process $S$:

\[ S \rightarrow \text{start} \cdot M \cdot S + \text{done} \]

We can give both for the automaton above:

\[
\begin{align*}
S & \rightarrow \text{start} \cdot M \cdot S + \text{done} \\
M & \rightarrow \text{move} \cdot M + \text{stop}
\end{align*}
\]
Preliminary Result

(Jos Baeten, Bas Luttik, Clemens Grabmayer)

\[
S \rightarrow \text{start} \cdot M \cdot S + \text{done} \\
M \rightarrow \text{move} \cdot M + \text{stop}
\]

- Automata theory: context-free language can be accepted by push-down automaton
- This specialised automaton employs a stack
- Process theory: context-free process can be transformed into a regular process communicating with a Stack process

\[
S = \text{start} \cdot \text{push}(S) \cdot M + \text{done} \cdot E_\theta \\
M = \text{move} \cdot M + \text{stop} \cdot E_\theta \\
E_\theta = \text{pop}(V) \cdot V + \text{empty}
\]
Research Questions

- New operators, new languages: expressiveness of these new languages?
- Finite axiomatisations?
- Extension of Chomsky hierarchy?
- More transformations?
Research Team

- prof.dr. J.C.M. Baeten,
- dr. C.A. Grabmayer,
- prof.dr. J. Karhumäki,
- dr. B. Luttik,
- prof.dr.ir. C.A. Middelburg,
- ir. P.J.A. van Tilburg.
Questions?