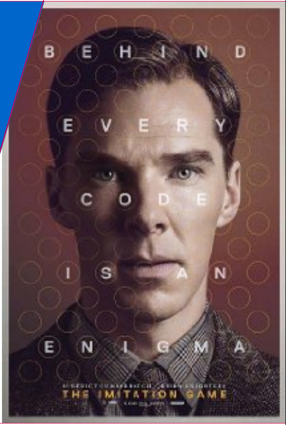
Executable Behaviours and the π-Calculus

Bas Luttik   Fei Yang



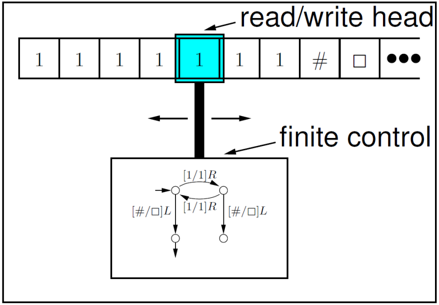
Technische Universiteit
Eindhoven
University of Technology

June 5, 2015

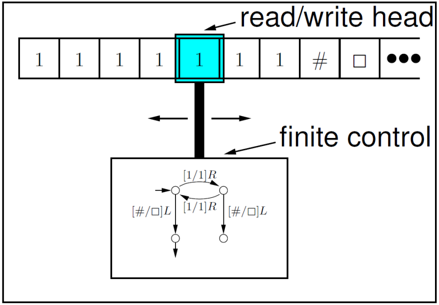
Where innovation starts

Outline

Turing machine and Computers



Turing machine and Computers



Church Turing Thesis: every computable function can be computed with a Turing Machine

How accurate is CT Thesis

The CT thesis is sometimes paraphrased as:
"A TM can do everything a real computer can do"

How accurate is CT Thesis

The CT thesis is sometimes paraphrased as:
"A TM can do everything a real computer can do"

Question:
"Is the statement valid for interactive computation?"

How accurate is CT Thesis

The CT thesis is sometimes paraphrased as:
“A TM can do everything a real computer can do”

Question:
“Is the statement valid for interactive computation?”

How accurate is CT Thesis

The CT thesis is sometimes paraphrased as:
"A TM can do everything a real computer can do"

Question:
"Is the statement valid for interactive computation?"

A TM cannot fly an aircraft.

How accurate is CT Thesis

The CT thesis is sometimes paraphrased as:
“A TM can do everything a real computer can do”

Question:
“Is the statement valid for interactive computation?”

A TM cannot fly an aircraft.

But a bunch of reactive computing systems operating concurrently can!

How accurate is CT Thesis

The CT thesis is sometimes paraphrased as:
“A TM can do everything a real computer can do”

Question:
“Is the statement valid for interactive computation?”

A TM cannot fly an aircraft.

But a bunch of reactive computing systems operating concurrently can!

Concurrency Theory is introduced to study such systems.

- Interaction: between parallel components

Computation in Concurrency Theory

- Interaction: between parallel components
- Non-termination : infinitely long execution sequence (divergence)

Computation in Concurrency Theory

- Interaction: between parallel components
- Non-termination : infinitely long execution sequence (divergence)
- Non-determinism : nondeterministic behaviours

Computation in Concurrency Theory

- Interaction: between parallel components
- Non-termination : infinitely long execution sequence (divergence)
- Non-determinism : nondeterministic behaviours

Concurrency Theory + CT Thesis?

Computation in Concurrency Theory

- Interaction: between parallel components
- Non-termination : infinitely long execution sequence (divergence)
- Non-determinism : nondeterministic behaviours

Concurrency Theory + CT Thesis?

Concurrency

Computation in Concurrency Theory

- Interaction: between parallel components
- Non-termination : infinitely long execution sequence (divergence)
- Non-determinism : nondeterministic behaviours

Concurrency Theory + CT Thesis?

Concurrency +Computability

Computation in Concurrency Theory

- Interaction: between parallel components
- Non-termination : infinitely long execution sequence (divergence)
- Non-determinism : nondeterministic behaviours

Concurrency Theory + CT Thesis?

Concurrency +Computability = Executability

Absolute Expressiveness

Given a process calculus:

Absolute Expressiveness

Given a process calculus:

- In classical theory of computability:

Absolute Expressiveness

Given a process calculus:

- In classical theory of computability:
  - Is it Turing powerful?

Absolute Expressiveness

Given a process calculus:

- In classical theory of computability:
  - Is it Turing powerful?
  - Is it computable?

Absolute Expressiveness

Given a process calculus:

- In classical theory of computability:
  - Is it Turing powerful?
  - Is it computable?

- In theory of executability:

Absolute Expressiveness

Given a process calculus:

- In classical theory of computability:
  - Is it Turing powerful?
  - Is it computable?

- In theory of executability:
  - Is it reactive Turing powerful?

Absolute Expressiveness

Given a process calculus:

- In classical theory of computability:
  - Is it Turing powerful?
  - Is it computable?
- In theory of executability:
  - Is it reactive Turing powerful?
  - Is it executable?

Outline

From Computability to Executability

Evaluating Expressiveness w.r.t. Executability

Expressiveness of the π-Calculus

- Reactive Turing Powerfulness of the π-Calculus
- Executability of the π-Calculus Processes

Outline

Reactive Turing Machines

$\mathcal{A}$: set of actions; τ: a special action ($\notin \mathcal{A}$), for unobservable actions.

$\mathcal{A}_\tau = \mathcal{A} \cup \{\tau\}$.

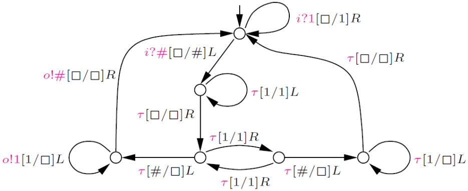
A reactive Turing machine (RTM) is a classical Turing machine with an action from some set $\mathcal{A}_\tau$ associated with every transition.

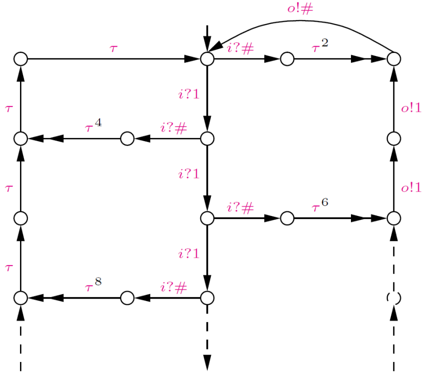
So RTMs have two types of transitions:

1. $s \xrightarrow{a[d/e]M} t$ means "externally observable, as execution of a"
2. $s \xrightarrow{\tau[d/e]M} t$ means "internal, unobservable transition"

M is ether "moving left" or "moving right"

Labelled Transition System of an RTM





We associate with every configuration (control state, tape instance) a state, and associate with every execution step a labelled transition.

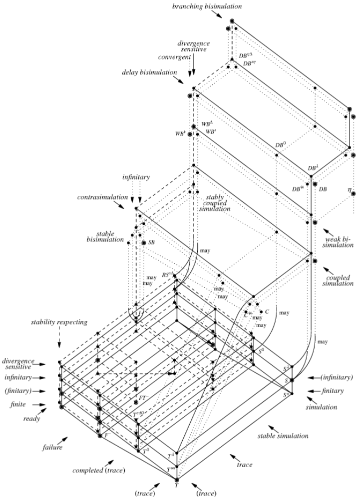
Executability and Behavioural Equivalence

A transition system is called executable if it is behaviourally equivalent to the transition system of an RTM.

Executability and Behavioural Equivalence

A transition system is called executable if it is behaviourally equivalent to the transition system of an RTM.

The notion of behavioural equivalence is a parameter of executability.

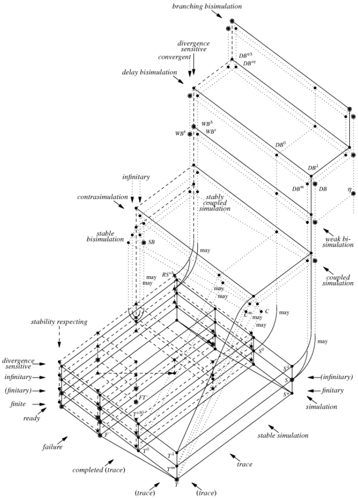


Executability and Behavioural Equivalence

A transition system is called executable if it is behaviourally equivalent to the transition system of an RTM.

The notion of behavioural equivalence is a parameter of executability.

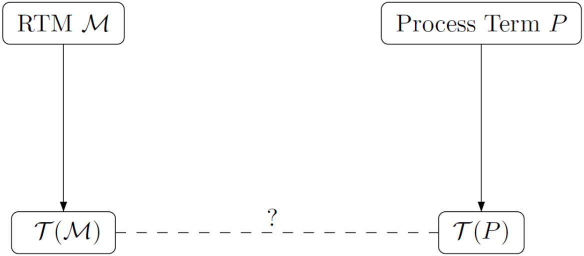
We start from (divergence-preserving) branching bisimilarity



Evaluating Expressiveness

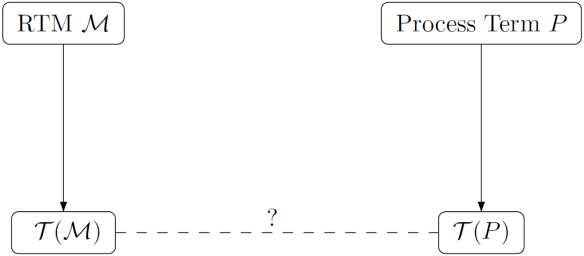
$$\begin{tikzcd} \text{RTM } \mathcal{M} \arrow[d] & \text{Process Term } P \arrow[d] \\ \mathcal{T}(\mathcal{M}) \arrow[r, dash, dashed, "?"] & \mathcal{T}(P) \end{tikzcd}$$

Evaluating Expressiveness



1. Can we specify every executable LTS by the LTS associated with *P*? (reactive Turing powerfulness?)

Evaluating Expressiveness



1. Can we specify every executable LTS by the LTS associated with *P*? (reactive Turing powerfulness?)

2. Is every LTS associated with the process specifiable by *P* executable? (executability)

Outline

From Computability to Executability

Evaluating Expressiveness w.r.t. Executability

Expressiveness of the π-Calculus

- Reactive Turing Powerfulness of the π-Calculus
- Executability of the π-Calculus Processes

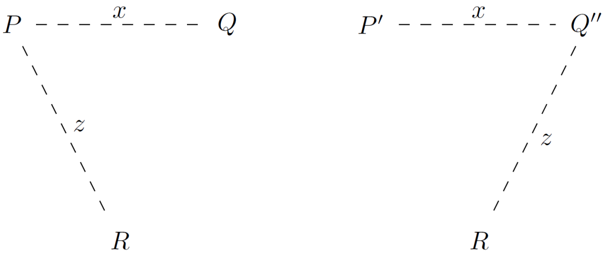
We presuppose a countably infinite set $\mathcal{N}$ of names.

The prefixes, processes and summations of the π-calculus are, respectively, defined by the following grammar:

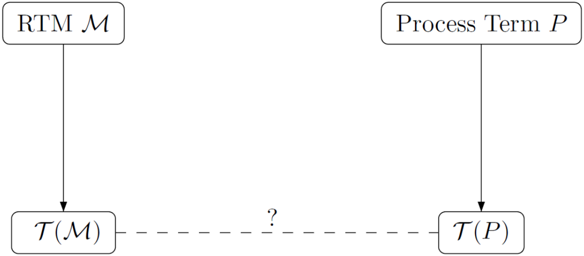
$$\begin{aligned}
\pi &:= \overline{x}y \mid x(z) \mid \tau \qquad (x, y, z \in \mathcal{N}) \\
P &:= M \mid P \mid P \mid (z)P \mid !P \\
M &:= 0 \mid \pi.P \mid M + M \quad .
\end{aligned}$$

Link Mobility

Suppose $P = \overline{x}z.P'$, $Q = x(y).Q'$.
Then $(z)(P \mid R) \mid Q \xrightarrow{\tau} P' \mid (z)(R \mid Q'')$, where $Q'' = \{z/y\}Q'$.



Expressiveness of the π-calculus



Expressiveness of the π-calculus

$$
\begin{tikzcd}
\boxed{\text{RTM } \mathcal{M}} \arrow[d] & & \boxed{\text{Process Term } P} \arrow[d] \\
\boxed{\mathcal{T}(\mathcal{M})} \arrow[rr, dash, dashed, "?"] & & \boxed{\mathcal{T}(P)}
\end{tikzcd}
$$

1. Can we specify every executable LTS in the π-calculus? (reactive Turing powerfulness?)

Expressiveness of the π-calculus

$$
\begin{tikzcd}
\boxed{\text{RTM } \mathcal{M}} \arrow[d] & & \boxed{\text{Process Term } P} \arrow[d] \\
\boxed{\mathcal{T}(\mathcal{M})} \arrow[rr, dash, dashed, "?"] & & \boxed{\mathcal{T}(P)}
\end{tikzcd}
$$

1. Can we specify every executable LTS in the π-calculus? (reactive Turing powerfulness?)

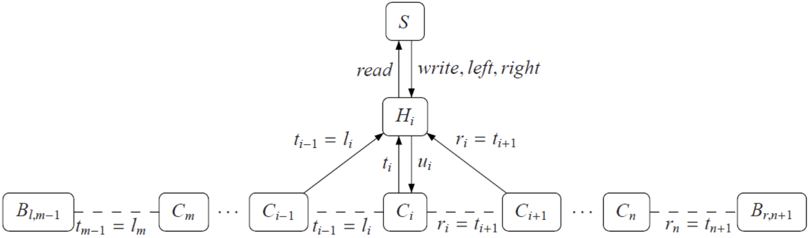
2. Is every LTS associated with the process specifiable in the π-calculus executable? (executability?)

Outline

Specification of an RTM

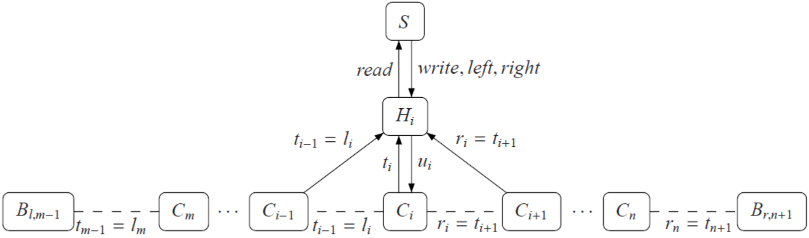
The specification contains two parts:

1. A generic process to specify the tape of a machine, and
2. a bunch specific processes for transition rules.



Tape

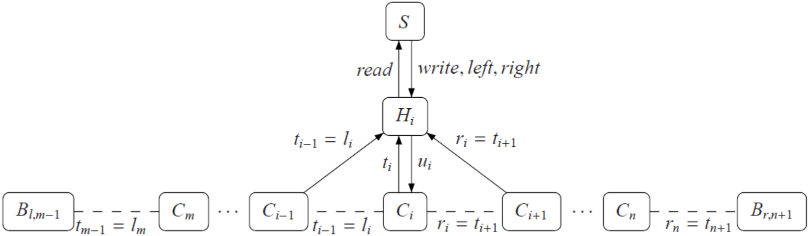
1. Tape head: read, write, move
2. Cells: an ordered sequence to record data
3. Generator: a facility to generate new cells



Control of the machine

The transition rules of RTMs are of the form:

$$s \xrightarrow{a[d/e]M} t$$

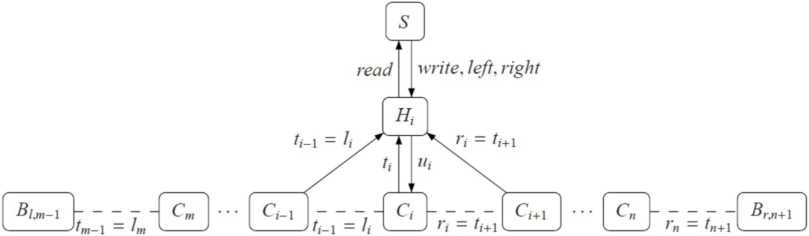


Control of the machine

The transition rules of RTMs are of the form:

$$s \xrightarrow{a[d/e]M} t$$

The state s and data d determine the set of subsequent transitions.



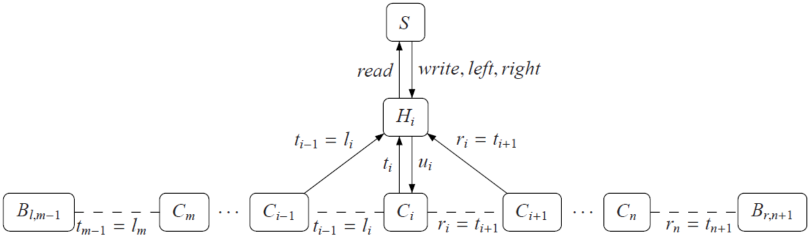
Control of the machine

The transition rules of RTMs are of the form:

$$s \xrightarrow{a[d/e]M} t$$

The state s and data d determine the set of subsequent transitions.

$$S_{s,d} \stackrel{\text{def}}{=} \sum_{(s,d,a,e,m,t)\in \longrightarrow_{\mathcal{M}}} a.\overline{write}\, e.\overline{m}.read(f).S_{t,f}$$



Theorem

For every executable transition system T there exists a π-term P, such that $T \underline{\leftrightarrow}^{\Delta}_{b} \mathcal{T}(P)$.

Theorem

For every executable transition system T there exists a π-term P, such that $T \mathrel{\underline{\leftrightarrow}}_b^{\Delta} \mathcal{T}(P)$.

π-calculus is reactive Turing powerful modulo divergence-preserving branching bisimilarity.

Outline

From Computability to Executability

Evaluating Expressiveness w.r.t. Executability

Expressiveness of the π-Calculus

- Reactive Turing Powerfulness of the π-Calculus
- Executability of the π-Calculus Processes

Simulating π-processes with RTMs

Infinitely many names vs. finitely many action labels

Simulating π-processes with RTMs

Infinitely many names vs. finitely many action labels

We cannot simulate every π-process with RTM :(

Simulating π-processes with RTMs

Infinitely many names vs. finitely many action labels

We cannot simulate every π-process with RTM :(

Two choices:

Extend the formalism of RTMs to an infinite set of actions.

Restrict the π-calculus with finitely many names.

Extend RTMs with infinitely many actions

An infinite alphabet of data symbols or control states is required.

Extend RTMs with infinitely many actions

An infinite alphabet of data symbols or control states is required.

Theorem

Every effective transition system can be simulated up to divergence-preserving branching bisimilarity by an RTM with infinite sets of action symbols and data symbols.

Extend RTMs with infinitely many actions

An infinite alphabet of data symbols or control states is required.

Theorem

Every effective transition system can be simulated up to divergence-preserving branching bisimilarity by an RTM with infinite sets of action symbols and data symbols.

Not realistic!

Restrict the π-calculus with finitely many names

Free names are restricted to a finite set.

Restrict the π-calculus with finitely many names

Free names are restricted to a finite set.

Bound names are considered as secret channels.

Restrict the π-calculus with finitely many names

Free names are restricted to a finite set.

Bound names are considered as secret channels.

An alternative semantics

For a finite set of names $\mathcal{N}'$ and a π-term P, we define the labelled transition system of P over $\mathcal{N}'$ as $\mathcal{T}(P) \upharpoonright \mathcal{N}'$, where

- all the transitions with a free name not in $\mathcal{N}'$ are excluded, and
- bound output with a label $\overline{x}(z)$ are renamed to $\nu\overline{x}$.

Restrict the π-calculus with finitely many names

Free names are restricted to a finite set.

Bound names are considered as secret channels.

An alternative semantics

For a finite set of names $\mathcal{N}'$ and a π-term P, we define the labelled transition system of P over $\mathcal{N}'$ as $\mathcal{T}(P) \restriction \mathcal{N}'$, where

- all the transitions with a free name not in $\mathcal{N}'$ are excluded, and
- bound output with a label $\overline{x}(z)$ are renamed to $\nu\overline{x}$.

$\mathcal{T}(P) \restriction \mathcal{N}'$ actually collects exactly all the behaviour of P regarding to $\mathcal{N}'$.

Executability of the π-calculus

Theorem

Every closed π-term with finitely many observable names is executable up to branching bisimilarity, but there exist closed π-terms with finitely many observable names that are not executable up to divergence-preserving branching bisimilarity.

Theorem

Every closed π-term with finitely many observable names is executable up to branching bisimilarity, but there exist closed π-terms with finitely many observable names that are not executable up to divergence-preserving branching bisimilarity.

It is executable modulo branching bisimilarity, and but not modulo divergence-preserving branching bisimilarity.

Conclusion

- The notion of reactive Turing machine and executability

Conclusion

- The notion of reactive Turing machine and executability
- A framework to evaluate the expressiveness for a model of concurrency

Conclusion

- The notion of reactive Turing machine and executability
- A framework to evaluate the expressiveness for a model of concurrency
- An application to the π-calculus
  - Reactive Turing powerfulness
  - Executability

Thank You!