Executable Behaviours and the $\pi$-Calculus

Bas Luttik    Fei Yang

June 5, 2015
From Computability to Executability

Evaluating Expressiveness w.r.t. Executability

Expressiveness of the $\pi$-Calculus
Reactive Turing Powerfulness of the $\pi$-Calculus
Executability of the $\pi$-Calculus Processes
Church Turing Thesis: every computable function can be computed with a Turing Machine.
Church Turing Thesis: every computable function can be computed with a Turing Machine
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.
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
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 $\pi$-Calculus
  Reactive Turing Powerfulness of the $\pi$-Calculus
  Executability of the $\pi$-Calculus Processes
Outline

From Computability to Executability

Evaluating Expressiveness w.r.t. Executability

Expressiveness of the $\pi$-Calculus
  Reactive Turing Powerfulness of the $\pi$-Calculus
  Executability of the $\pi$-Calculus Processes
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”
We associate with every configuration (control state, tape instance) a state, and associate with every execution step a labelled transition.
A transition system is called executable if it is behaviourally equivalent to the transition system of an RTM.
A transition system is called **executable** if it is **behaviorally equivalent** to the transition system of an RTM.

The notion of behavioral equivalence is a **parameter** of executability.
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

1. Can we specify every executable LTS by the LTS associated with \( \mathcal{M} \)? (reactive Turing powerfulness?)

2. Is every LTS associated with the process specifiable by \( \mathcal{M} \) executable? (executability)
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 $\pi$-Calculus
  Reactive Turing Powerfulness of the $\pi$-Calculus
  Executability of the $\pi$-Calculus Processes
We presuppose a countably infinite set $\mathcal{N}$ of names.

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

$$
\pi := \overline{xy} \mid x(z) \mid \tau \quad (x, y, z \in \mathcal{N})
$$

$$
P := M \mid P \mid P \mid (z)P \mid !P
$$

$$
M := 0 \mid \pi.P \mid M + M
$$
Suppose $P = \bar{x}z.P', \ Q = x(y).Q'$.
Then $(z)(P \mid R) \mid Q \xrightarrow{\tau} P' \mid (z)(R \mid Q'')$, where $Q'' = \{z/y\}Q'$. 

---

![Diagram with arrows and variables]

---

TU/e Technische Universiteit Eindhoven University of Technology
Expressiveness of the $\pi$-calculus

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

2. Is every LTS associated with the process specifiable in the $\pi$-calculus executable? (executability?)
Expressiveness of the $\pi$-calculus

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

2. Is every LTS associated with the process specifiable in the $\pi$-calculus executable? (executability?)
Expressiveness of the $\pi$-calculus

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

2. Is every LTS associated with the process specifiable in the $\pi$-calculus executable? (executability?)
From Computability to Executability

Evaluating Expressiveness w.r.t. Executability

Expressiveness of the $\pi$-Calculus
Reactive Turing Powerfulness of the $\pi$-Calculus
Executability of the $\pi$-Calculus Processes
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.
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.

\[ S_{s,d} \overset{\text{def}}{=} \sum_{(s,d,a,e,m,t) \in \rightarrow M} \overline{a.\text{write } e.\overline{m}.\text{read}(f).S_{t,f}} \]

![Diagram]

- 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.

\[ S_{s,d} \overset{\text{def}}{=} \sum_{(s,d,a,e,m,t) \in \rightarrow M} \overline{a.\text{write } e.\overline{m}.\text{read}(f).S_{t,f}} \]
Theorem

For every executable transition system $T$ there exists a $\pi$-term $P$, such that $T \xrightarrow{b} T(P)$. 

π-calculus is reactive Turing powerful modulo divergence-preserving branching bisimilarity.
Expressiveness of the $\pi$-Calculus

Theorem
For every executable transition system $T$ there exists a $\pi$-term $P$, such that $T \leftrightarrow^b T(P)$.

$\pi$-calculus is reactive Turing powerful modulo divergence-preserving branching bisimilarity.
Outline

From Computability to Executability

Evaluating Expressiveness w.r.t. Executability

Expressiveness of the $\pi$-Calculus
  Reactive Turing Powerfulness of the $\pi$-Calculus
  Executability of the $\pi$-Calculus Processes
Simulating $\pi$-processes with RTMs

Infinitely many names vs. finitely many action labels
Simulating $\pi$-processes with RTMs

Infinitely many names vs. finitely many action labels

We cannot simulate every $\pi$-process with RTM :(
Simulating $\pi$-processes with RTMs

Infinitely many names vs. finitely many action labels

We cannot simulate every $\pi$-process with RTM :( 

Two choices:

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

Restrict the $\pi$-calculus with finitely many names.
An infinite alphabet of data symbols or control states is required.
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.*
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!
Free names are restricted to a finite set.
Restrict the $\pi$-calculus with finitely many names

**Free** names are restricted to a finite set.

**Bound** names are considered as secret channels.
Restrict the $\pi$-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 $\pi$-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 $\pi$-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 $N'$ and a $\pi$-term $P$, we define the labelled transition system of $P$ over $N'$ as $\mathcal{T}(P) \upharpoonright N'$, where

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

$\mathcal{T}(P) \upharpoonright N'$ actually collects exactly all the behaviour of $P$ regarding to $N'$. 
Theorem
Every closed $\pi$-term with finitely many observable names is executable up to branching bisimilarity, but there exist closed $\pi$-terms with finitely many observable names that are not executable up to divergence-preserving branching bisimilarity.
Theorem

Every closed $\pi$-term with finitely many observable names is executable up to branching bisimilarity, but there exist closed $\pi$-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 $\pi$-calculus
  - Reactive Turing powerfulness
  - Executability
Thank You!