Integrating Automata Theory and Process Theory
(status and open problems)

Bas Luttik

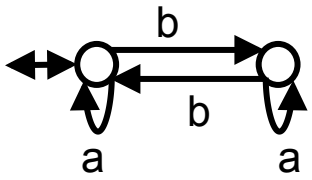
(based on joint work with Jos Baeten, Paul van Tilburg and Fei Yang)

Open Problems in Concurrency Theory

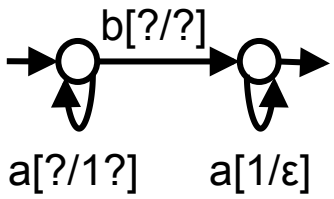
Bertinoro, 19 June 2014

Automata Theory (classical view)

DFA/NFA



PDA



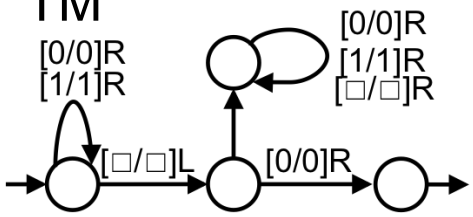
FORMAL LANGUAGE

Regular expression/

Linear grammar

Context-free grammar

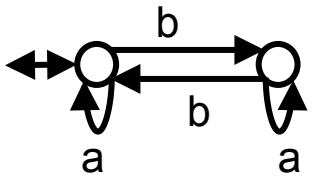
TM



Unrestricted grammar

Automata Theory (classical view)

DFA/NFA

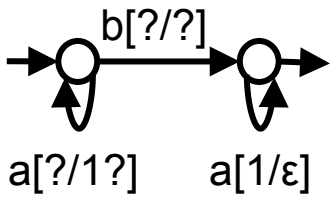


Regular Language

Regular expression/

Linear grammar

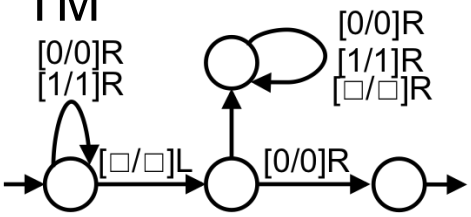
PDA



Context-free Language

Context-free grammar

TM



Recursively enumerable Language

Unrestricted grammar

Automata and Process Theory in a single course

Automata Theory & Formal Languages

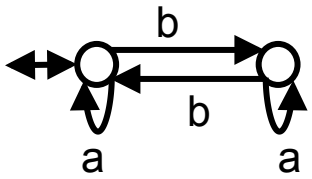
- From NFAs to DFAs (determinizing)
- From REs to NFAs v.v.
- Pumping lemma's
- Correspondence between CFGs and PDAs
- Formal syntax (e.g. of regular expressions, grammars)
- Halting problem

Concurrency Theory

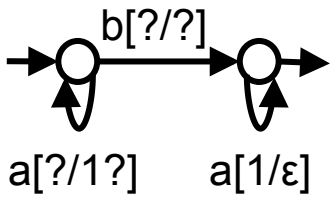
- Transition systems
- Process specification (e.g. using parallel composition)
- Structural operational semantics
- Behavioural equivalence (bisimilarity)
- Abstraction
- Axioms

Automata Theory (more modern view)

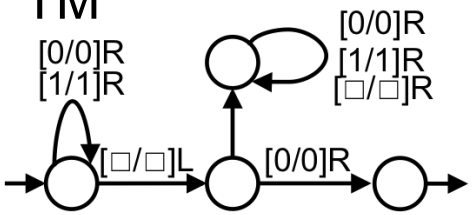
DFA/NFA



PDA



TM



TRANSITION SYSTEM

S
O
S

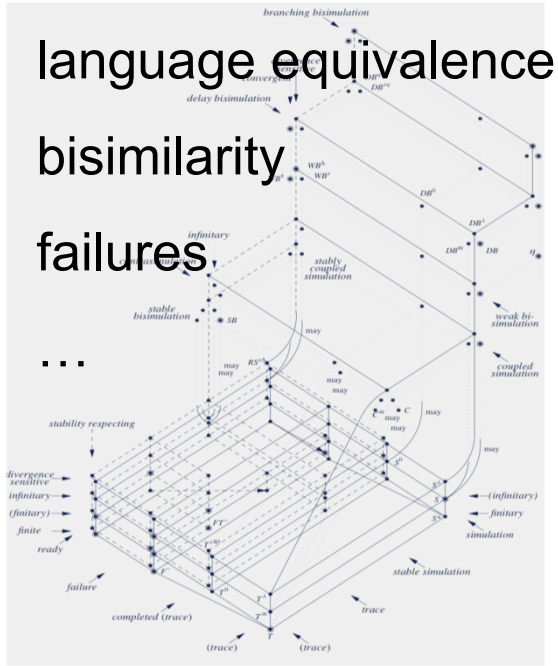
Regular expression/
Closed $BPA^*_{0,1}$-term
Linear grammar/
Recursive spec.
over $BCCSP_{0,1}$

language equivalence

bisimilarity

failures

…



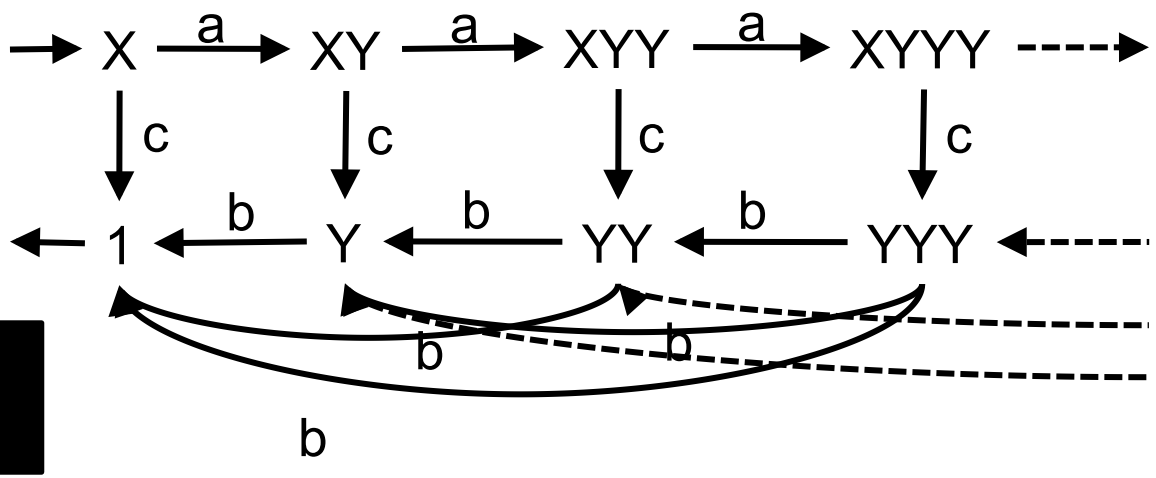
Decidability of Bisimilarity

The process theoretic counterpart of a context-free grammar is, then, a finite guarded recursive specification over $BPA_{0,1}$.

Consider the following guarded recursive specification over $BPA_{0,1}$:

$X = aXY + c$

$Y = b + 1$



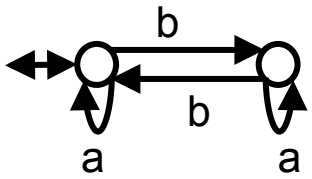
No bound on branching degree!

Open problem:

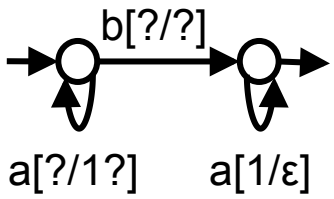
Is bisimilarity decidable for $BPA_{0,1}$ with guarded recursion?

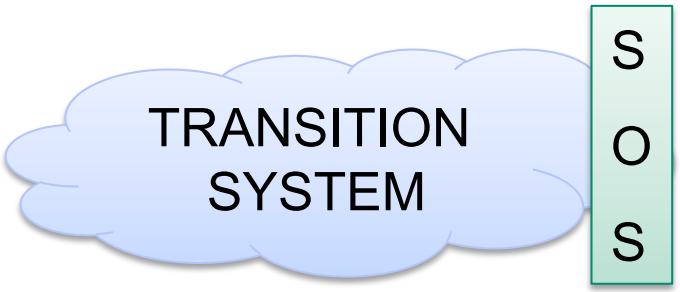
Automata Theory (more modern view)

DFA/NFA



PDA

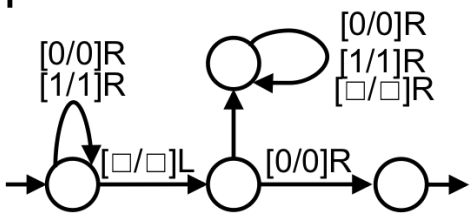




Regular expression/
Closed $BPA^*_{0,1}$-term
Linear grammar/
Recursive spec.
over $BCCSP_{0,1}$

Context-free
grammar
Recursive spec.
over $BPA_{0,1}$

TM



Unrestricted
grammar

???????

Reactive Turing machines

Design criteria

- Conservativity

  There should be a straightforward embedding of classical Turing machines into our new formalism

- Reactivity

  There should be a straightforward embedding of classical Turing machines into our new formalism

- Concurrency

  It should be possible to model some form of concurrency

Definition

A **reactive Turing machine (RTM)** is an ordinary Turing machine with an action from some set $A \cup \{\tau\}$ associated with every transition:

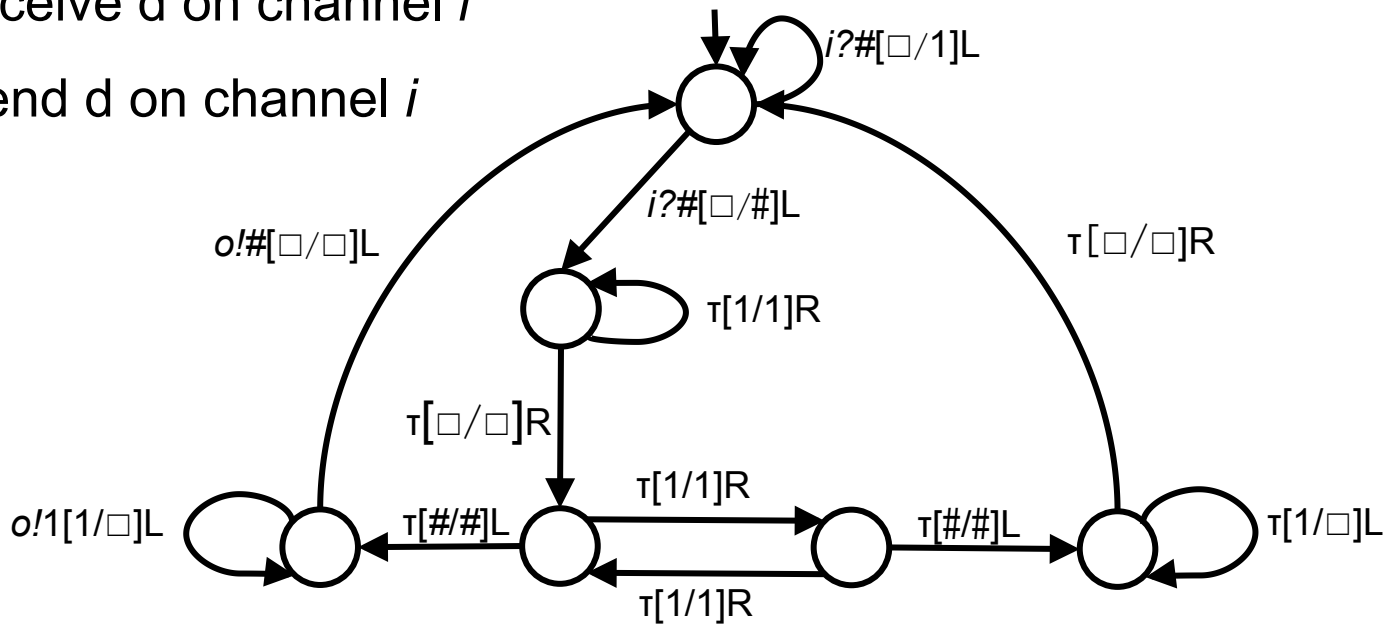
$s \xrightarrow{a[d/e]M} t$ means "externally observable, as execution of *a*"

$s \xrightarrow{\tau[d/e]M} t$ means "internal (unobservable) transition"

Example

i?d: receive d on channel *i*

o!d: send d on channel *i*



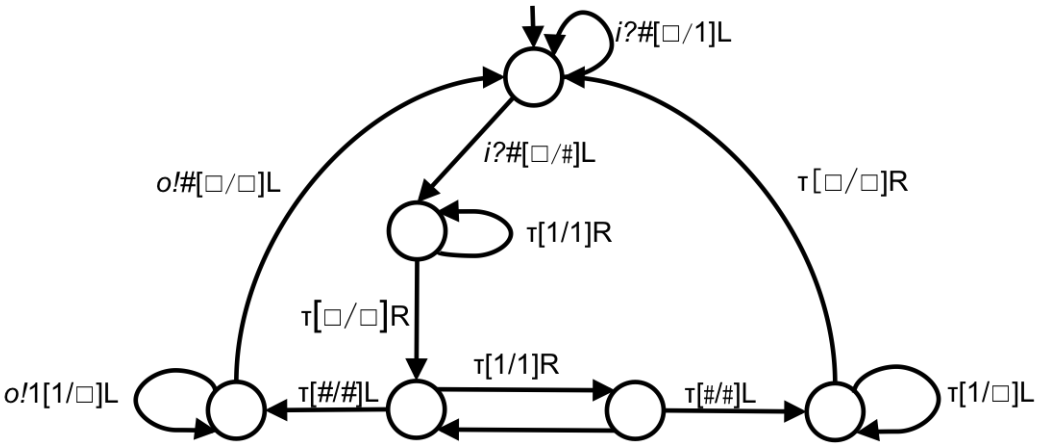
tape:

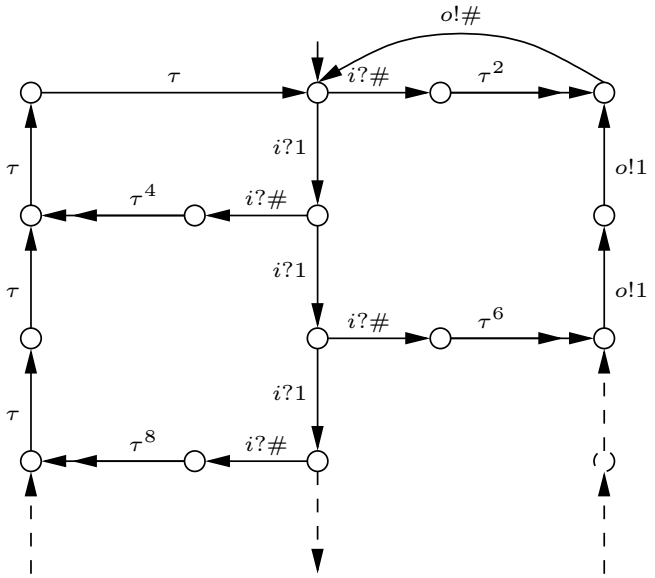
…				1	1	#			…

externally initiated/observed actions: *i?*1 *i?*1 *i?*# *o!*1 *o!*1 *o!*# *i?*1 *i?*# *o!*#

Operational semantics

We can now associate with every RTM a transition system.





We call a transition system is **executable** if it is the transition system associated with some RTM

How robust is this notion?

Expressiveness

A transition system is **effective** if its transition relation and termination predicate are recursively enumerable (as sets).

A finitely branching transition system is **computable** if there exists a recursive function associating with every state its set of outgoing transitions (and also the characteristic function of the termination predicate is recursive).

A transition system is **boundedly branching** if there exists a bound on the branching degrees of its states.

Theorem

1. The transition system associated with an RTM is computable and boundedly branching.
2. Every *boundedly branching* computable transition system is divergence-preserving branching bisimilar to that of an RTM.

The role of divergence

Theorem [Phillips 1993]

Every effective transition system is branching bisimilar to a computable transition system whose states have a branching degree less or equal 2.

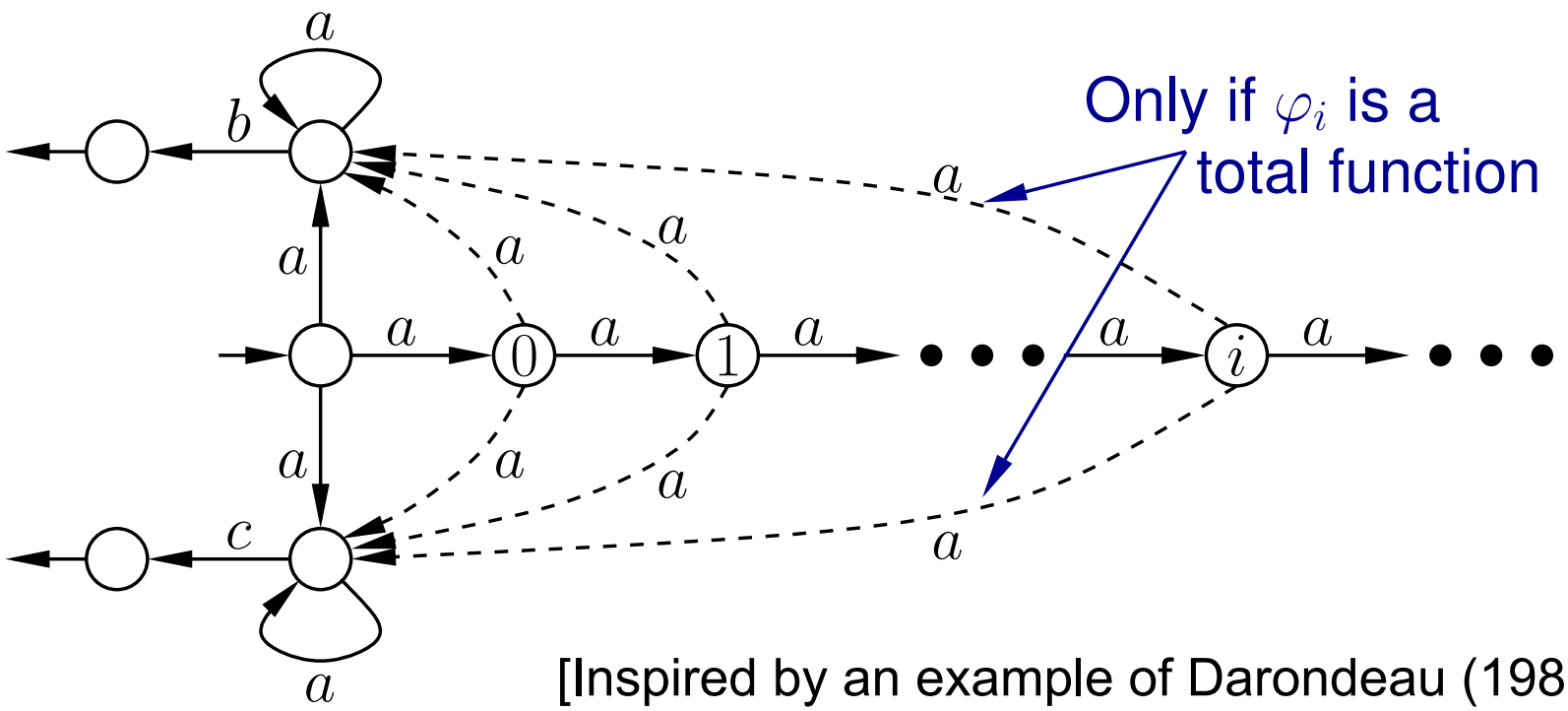
Phillips' result essentially depends on the use of a divergence to enumerate the outgoing transitions of state.

This motivates to adopt the refined view of divergence-preserving branching bisimilarity whenever possible.

Corollary

Every effective transition system is branching bisimilar to a the transition system associated with an RTM.

Not executable



[Inspired by an example of Darondeau (1989)]

Note that the language (i.e., $\{a^n b, a^n c \mid n \geq 1\}$ associated with the transition system above is context-free.

The behaviour is, however, not executable up to branching bisimilarity.

Parallel composition of RTMs

Since we have a transition system semantics for RTMs, we can define a notion parallel composition on RTMs (in any way we like!).

Here's just one proposal:

Let C be a set of **channels** and D be a set of **data**, and let

$A = \{c!\mathrm{d}, c?\mathrm{d} \mid c \in C, \mathrm{d} \in \mathrm{D}\}$.

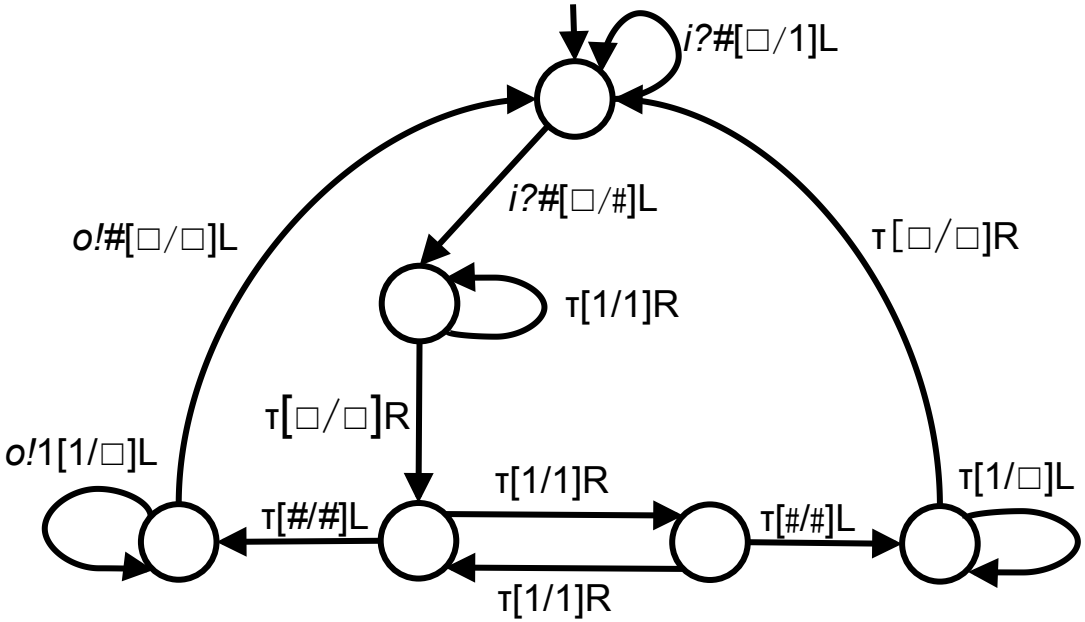
Let $C' \subseteq C$, and let M_1 and M_2 be RTMs.

We denote by $[M_1 \mid M_2]_{C'}$ the **parallel composition** of M_1 and M_2, communicating along channels in C'.

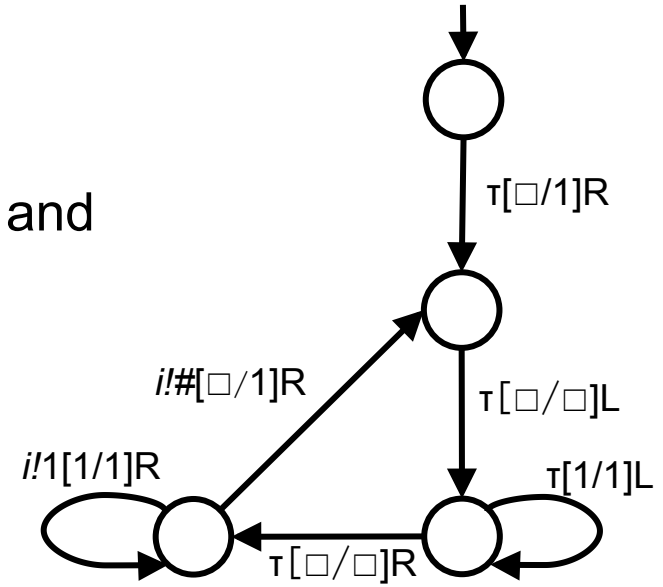
The transition system associated with $[M_1 \mid M_2]_{C'}$ is the parallel composition of the transitions associated with M_1 and M_2.

Example: parallel composition

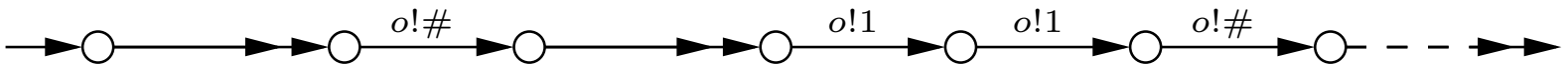
The transition system associated with the parallel composition of



and



is



Universality

Denote by $\underline{M}$ the RTM that outputs a description (i.e., encoding) of M along some channel u.

Definition

An RTM U is **universal** if, for every RTM M, the transition system associated with M is *behaviourally equivalent* to $[U \mid \underline{M}]_{\{u\}}$.

Taking branching bisimilarity as behavioural equivalence, there exist universal RTMs.

Taking *divergence-preserving* branching bisimilarity as behavioural equivalence, there do not exist universal RTMs!

Universality up to a branching degree

Denote by $\underline{M}$ the RTM that outputs a description (i.e., encoding) of M along some channel u.

Definition

Let B be a natural number. Then U is **universal up to B** if, for every RTM M whose associated transition system T_M has a branching degree bounded by B, T_M is *behaviourally equivalent* to $[U \mid \underline{M}]_{\{u\}}$.

Theorem

For every B there exists an RTM that is universal up to B.

Simulating RTMs in a process calculus

The behaviour of an arbitrary RTM can be simulated (up to divergence-preserving branching bisimilarity) by a finite guarded recursive specification over a process calculus with

- inaction (0)
- successful termination (1)
- action prefix (a.p)
- choice (p+q)
- parallel composition with (enforced) communication ($[p|q]_{C'}$)

We now also have a proposal for simulating RTMs (without termination!) in π-calculus.

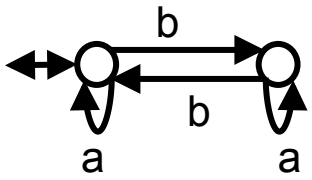
Open problems:

Are π-processes executable*? Up to which behavioural equivalence?*

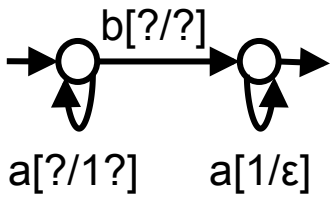
How to deal with unbounded branching stemming from input prefix?

Integrated Automata and Process Theory

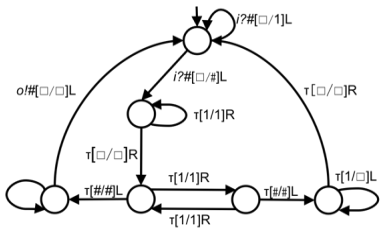
DFA/NFA



PDA



RTM



TRANSITION SYSTEM

S
O
S

Regular expression/
Closed $BPA^*_{0,1}$-term
Linear grammar/
Recursive spec.
over $BCCSP_{0,1}$

Context-free
grammar
Recursive spec.
over $BPA_{0,1}$

Recursive specs
over BCP_τ

pi-calculus?

Some concluding remarks

We have extended Turing machines with interaction, style concurrency theory.

RTMs may serve as an absolute expressiveness criterion for process calculi.

Other interactive variants of the Turing machine have been proposed in the literature (most notably: persistent Turing machines [GSAS04] and interactive Turing machines [vLW01]). These proposals add interaction in a less general form and can be simulated by our notion.

We are aiming for an Executability Thesis:

> *A process is executable (i.e., describes the behaviour of a computing system) if, and only if, it can be simulated by an RTM.*