Reactive Turing Machines and ACP$_\tau$

Paul van Tilburg

(joint work with Jos Baeten and Bas Luttik)

YR-CONCUR 2010,
September 4, 2010

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Outline

- Historical context
- Motivation
- Reactive Turing machine
- Results
- Related & future work

Turing Machines

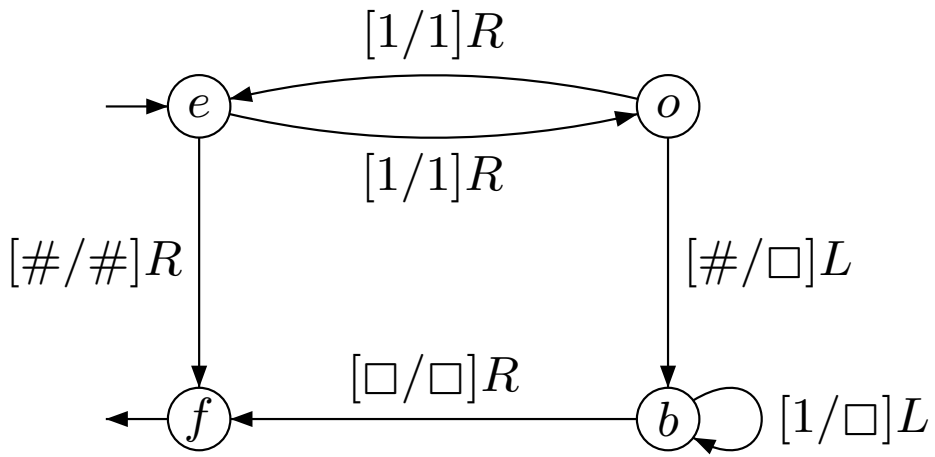
A(utomatic)-machines (later: Turing machines)

- Defined by Alan Turing in 1936
- Infinite memory in the form of a tape
- Head that reads/writes one symbol at a time
- Finite control of the head
  - reading and writing
  - moving

Turing Machines

A(utomatic)-machines (later: Turing machines)

- Defined by Alan Turing in 1936
- Infinite memory in the form of a tape
- Head that reads/writes one symbol at a time
- Finite control of the head
  - reading and writing
  - moving



$\mathcal{M} = (\mathcal{S}, \mathcal{D}, \rightarrow, \uparrow, \downarrow)$ with

$\mathcal{S} = \{e, o, f, b\}$

$\mathcal{D} = \{1, \#\}$

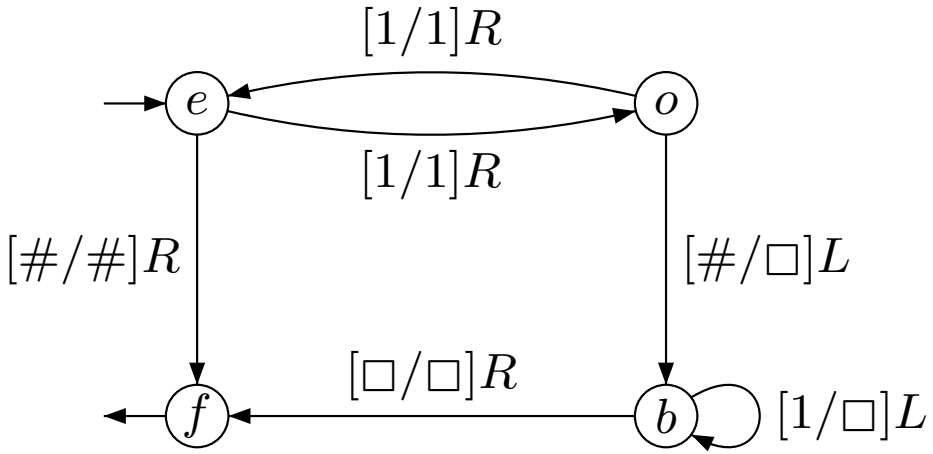
$\uparrow = e$

$\downarrow = \{f\}$

Turing Machines

A(utomatic)-machines (later: Turing machines)

- Defined by Alan Turing in 1936
- Infinite memory in the form of a tape
- Head that reads/writes one symbol at a time
- Finite control of the head
  - reading and writing
  - moving



Input: $\ldots\overline{1}111\#\ldots$

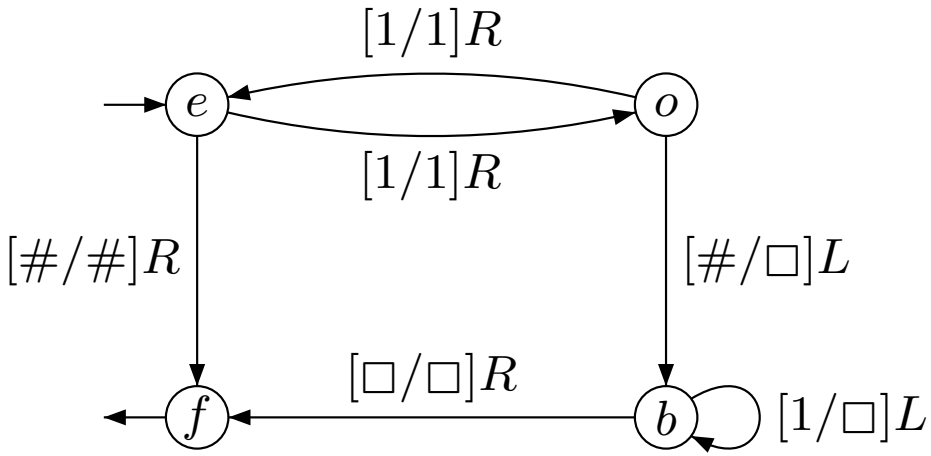
$\Downarrow$

Output: $\ldots1111\#\,\overline{\square}\ldots$

Turing Machines

A(utomatic)-machines (later: Turing machines)

- Defined by Alan Turing in 1936
- Infinite memory in the form of a tape
- Head that reads/writes one symbol at a time
- Finite control of the head
  - reading and writing
  - moving



Input: $\ldots \overline{1}\ 1\ 1\ \#\ldots$

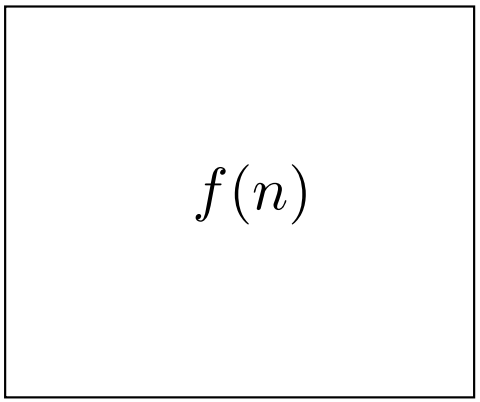
$\Downarrow$

Output: $\ldots \overline{\square}\ldots$

Turing Machines

A(utomatic)-machines (later: Turing machines)

- Defined by Alan Turing in 1936
- Infinite memory in the form of a tape
- Head that reads/writes one symbol at a time
- Finite control of the head
  - reading and writing
  - moving



Input: $\ldots \overline{1}\ 1\ \ 1\ \#\ldots$

$\Downarrow$

Output: $\ldots \overline{\square} \ldots$

Turing Machines (2)

Universal Turing machines

- Church-Turing thesis:
  "Everything computable is computable by a Turing machine"
- Models a computation/function
  - The TM converts input on the tape to output
- Model is also close to a computer of the '70s (program, CPU, RAM)
  - Input available at the start
  - Calculation is performed
  - Output generated at the end
- Criticism possible on suitability as a theoretical model of a modern-day computer
- Still, the TM entered the books as theoretical model
- (However, still works well for models of computations!)

Reactive Systems

"A Turing machine cannot fly a plane, but a real computer can!"

Properties

- Non-termination
- Interaction (with the environment)

Examples

- Cloud computing
- Mobile phones
- ...
- Processes of an operating system
- Objects in a virtual machine

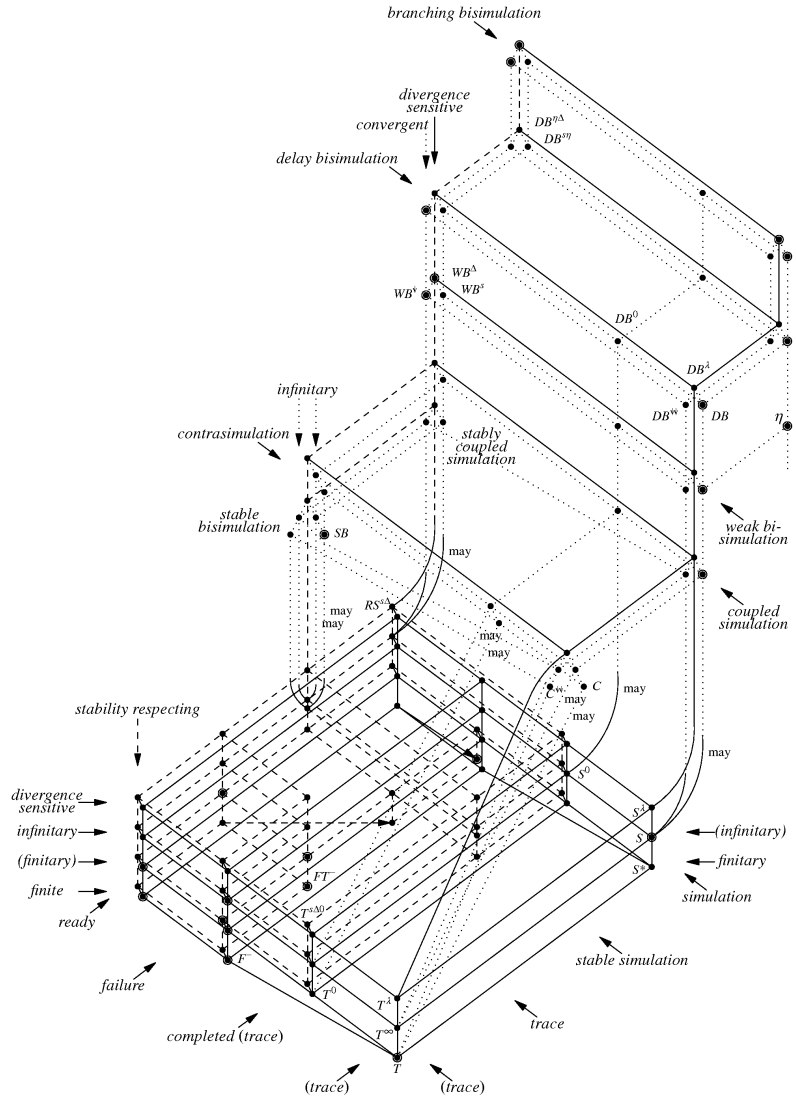
Concurrency Theory

- Started with Petrinets in the ’60s
- Given boost by Milner in the ’70s
- Goals for concurrency theory according to Milner:
  - Study concurrency and interaction in isolation
  - Only a single combiner for combining processes
- This work is done within the MoCAP project
  - Consider definitions and results from automata theory
  - . . . using a process-theoretic point of view
  - Obtain stronger results using concurrency theory
  - For example by considering (branching) bisimulation
- Side-goal: the design and teaching of a new course
  - In a theoretical course the model could prove useful

Benefits

Linear time—branching time spectrum

- By Van Glabbeek in 1993
- Spectrum gives us many equivalences
- Goal: be as high in the spectrum as possible
- Branching bisimulation



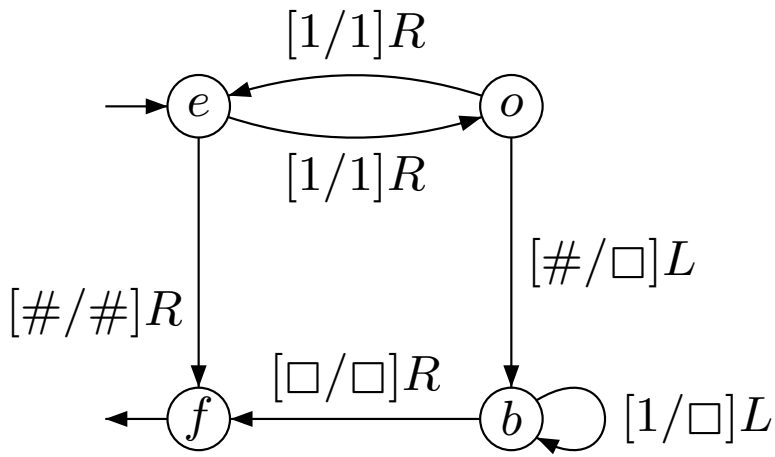
Conceptual Model

To summarise

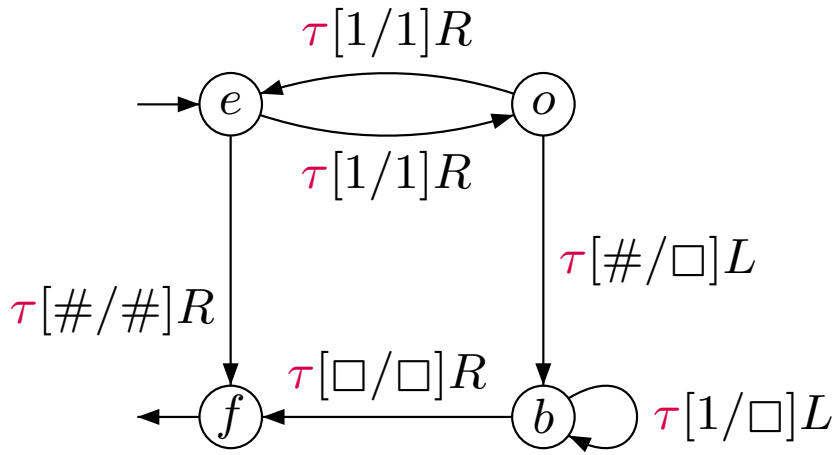
- We want a conceptual model of a computer rather than a model of computation
- We want to have non-termination and to make interaction explicit
- We use concurrency theory to have a plethora of process calculi, behavioural equivalences at our disposal
- Finite control is a program running on the CPU, tape is memory, interaction possible via network or I/O to user

- We aim to integrate computability and concurrency theory
- Our aim is not to increase the computational power of the traditional model nor to investigate the extra expressivity of interaction

Let's consider an example:

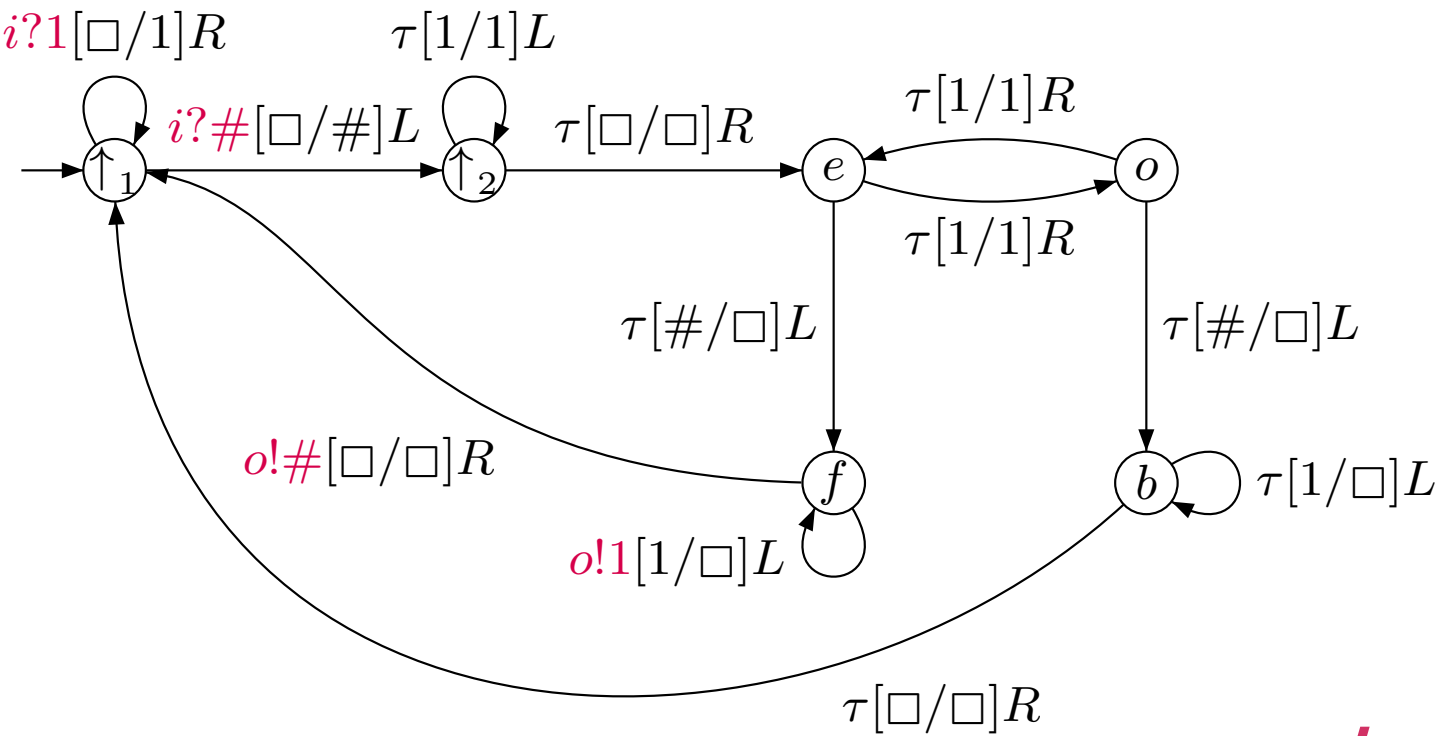


Let's consider an example:

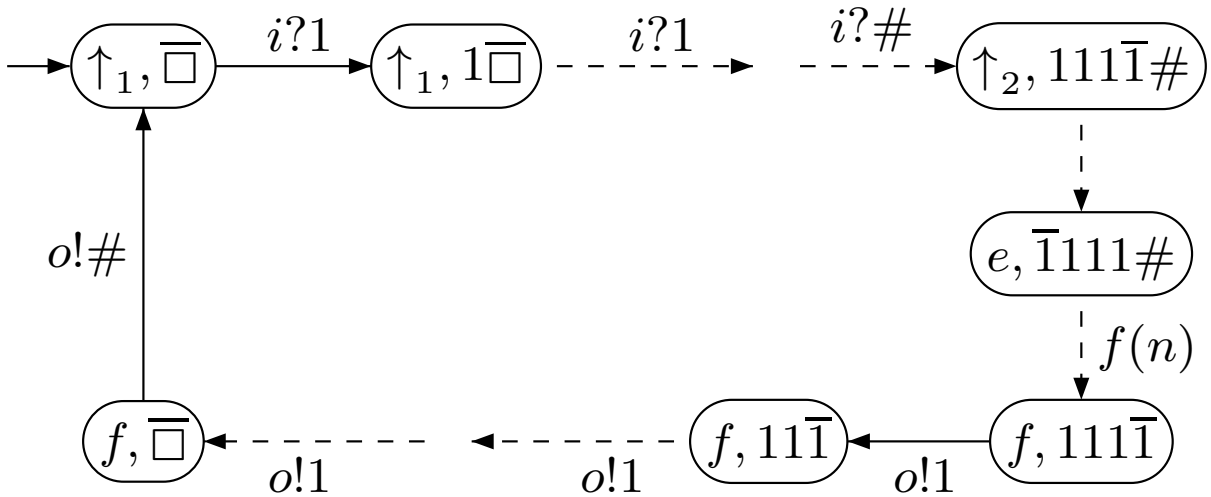


Reactive Turing Machines

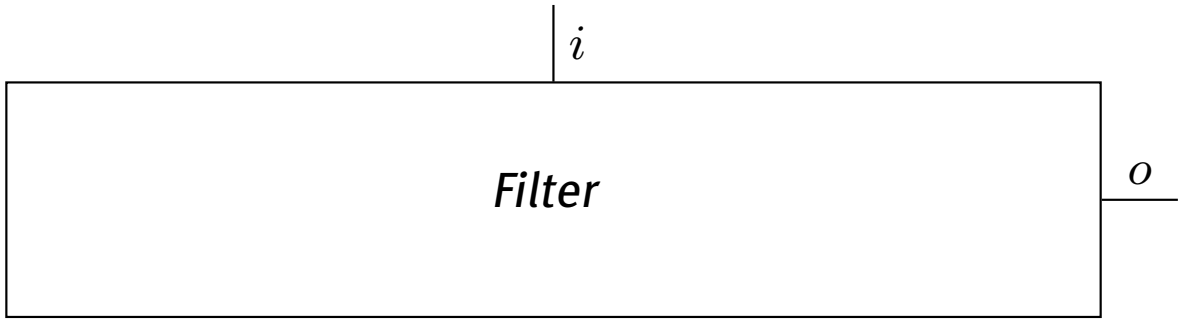
Let's consider an example:



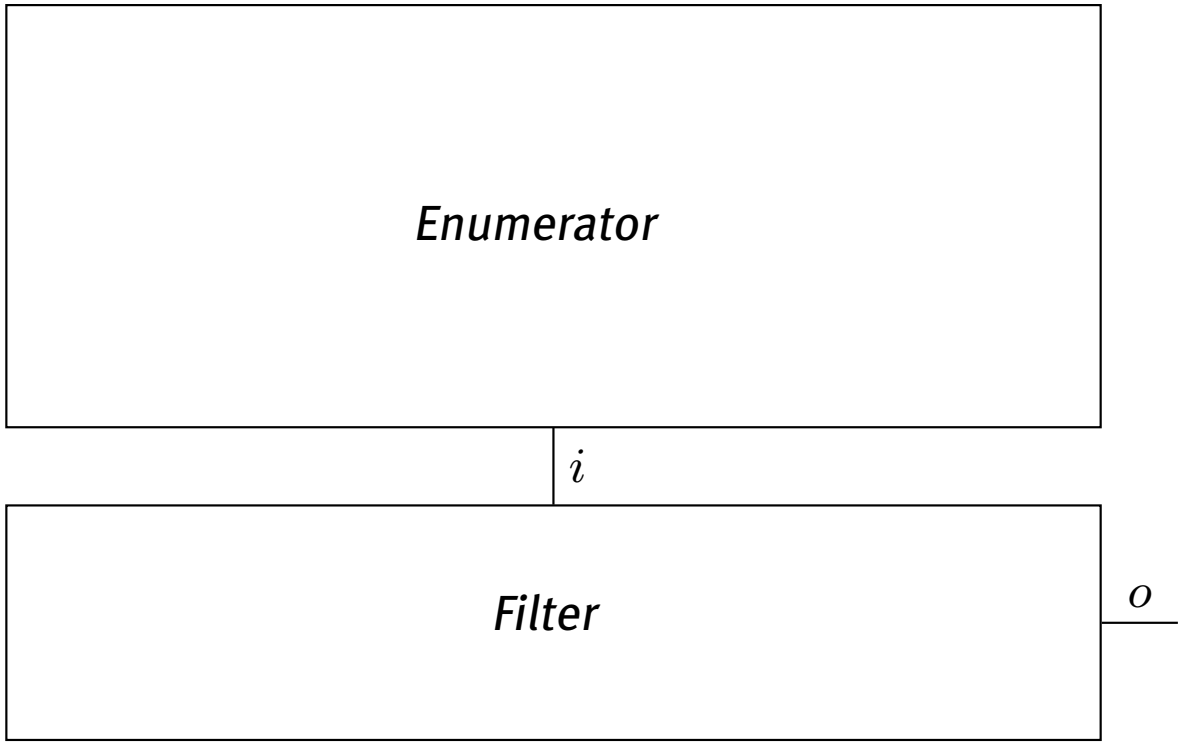
Reactive Turing Machines (2)



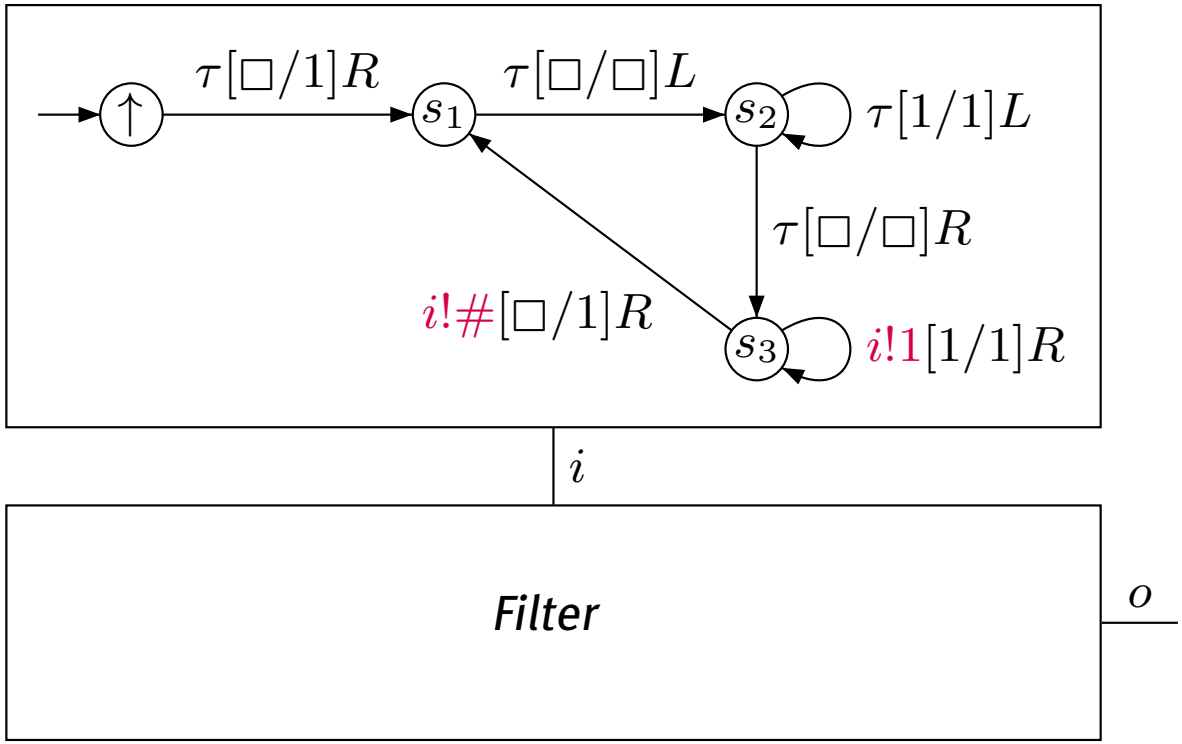
Reactive Turing Machines (3)



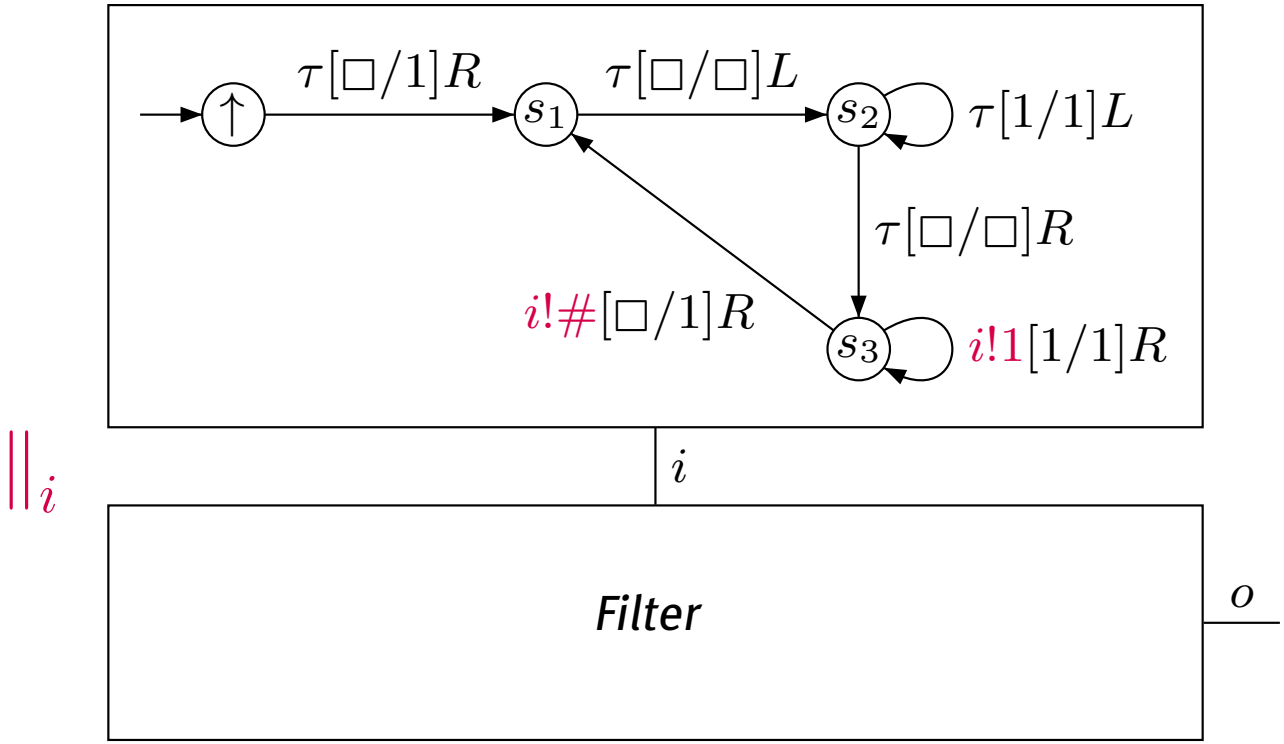
Reactive Turing Machines (3)



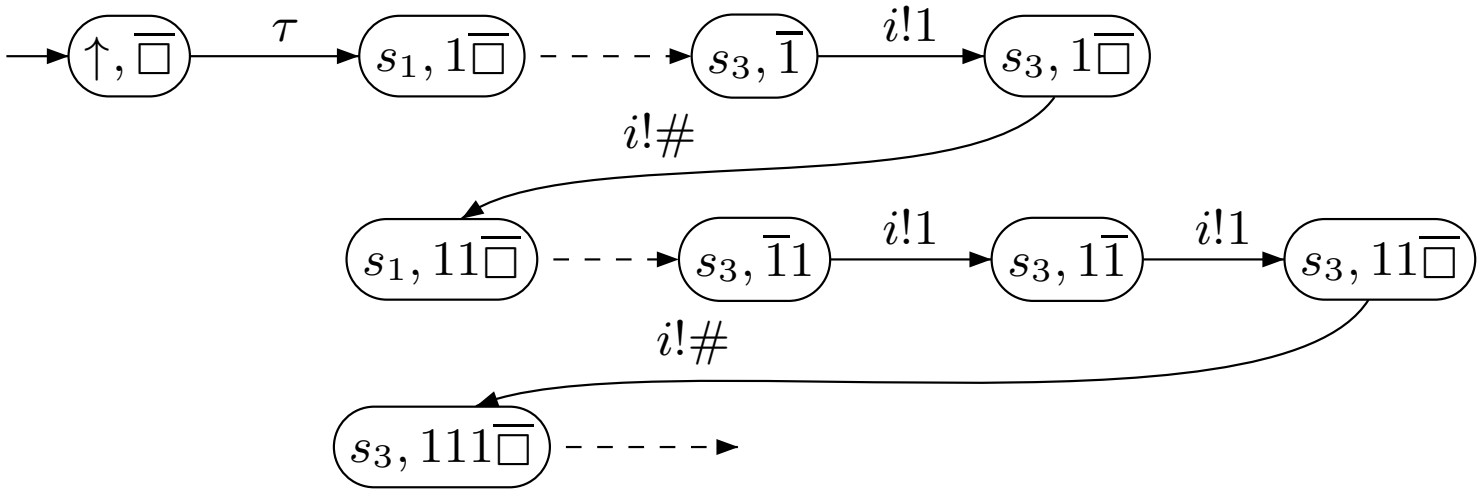
Reactive Turing Machines (3)



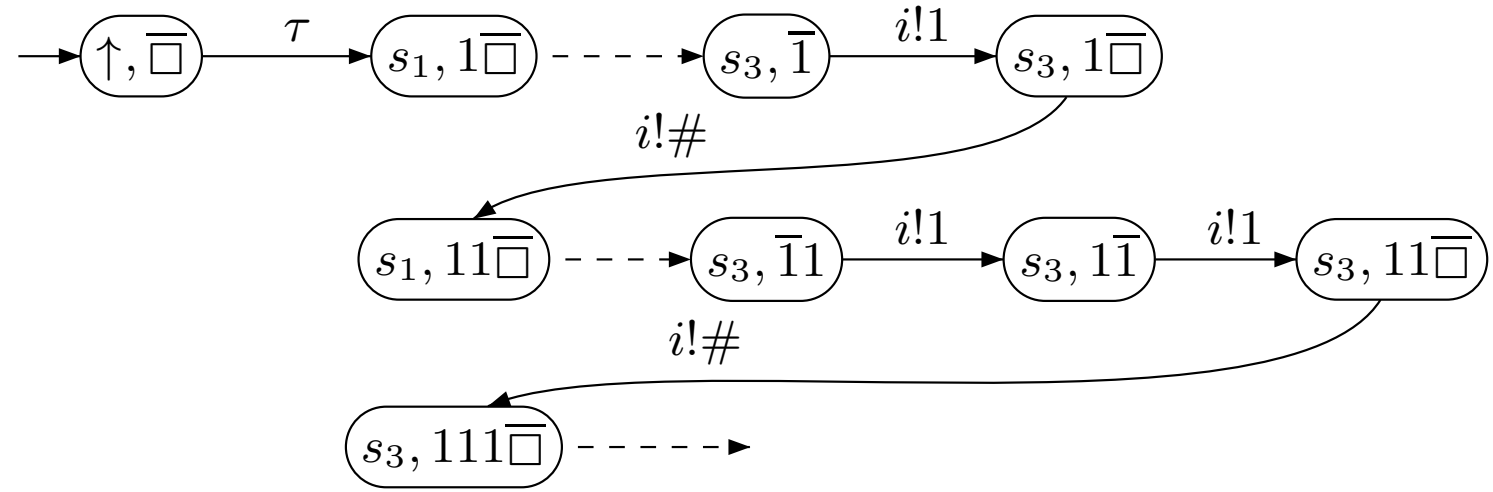
Reactive Turing Machines (3)

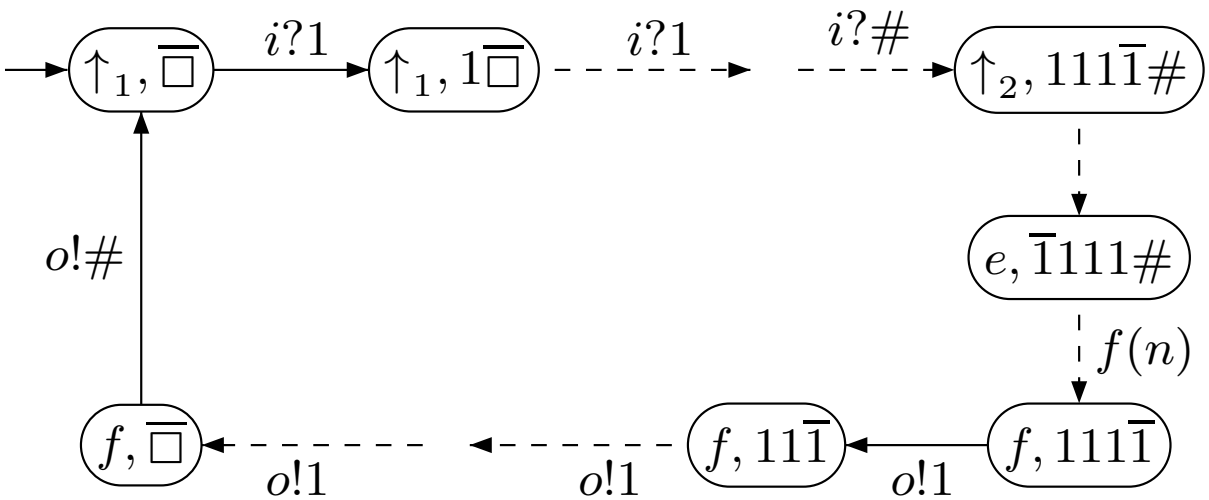


Reactive Turing Machines (4)

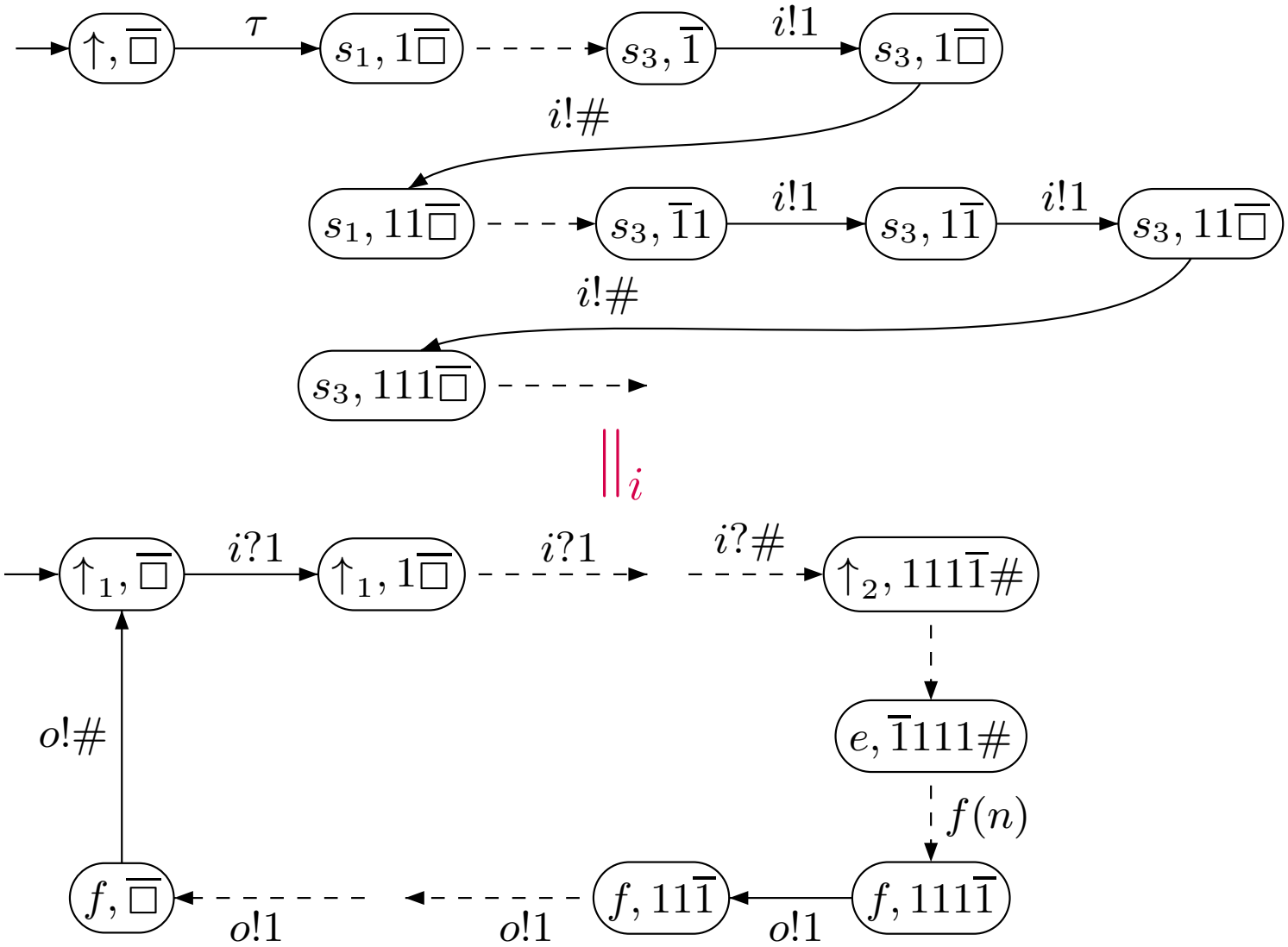


Reactive Turing Machines (4)

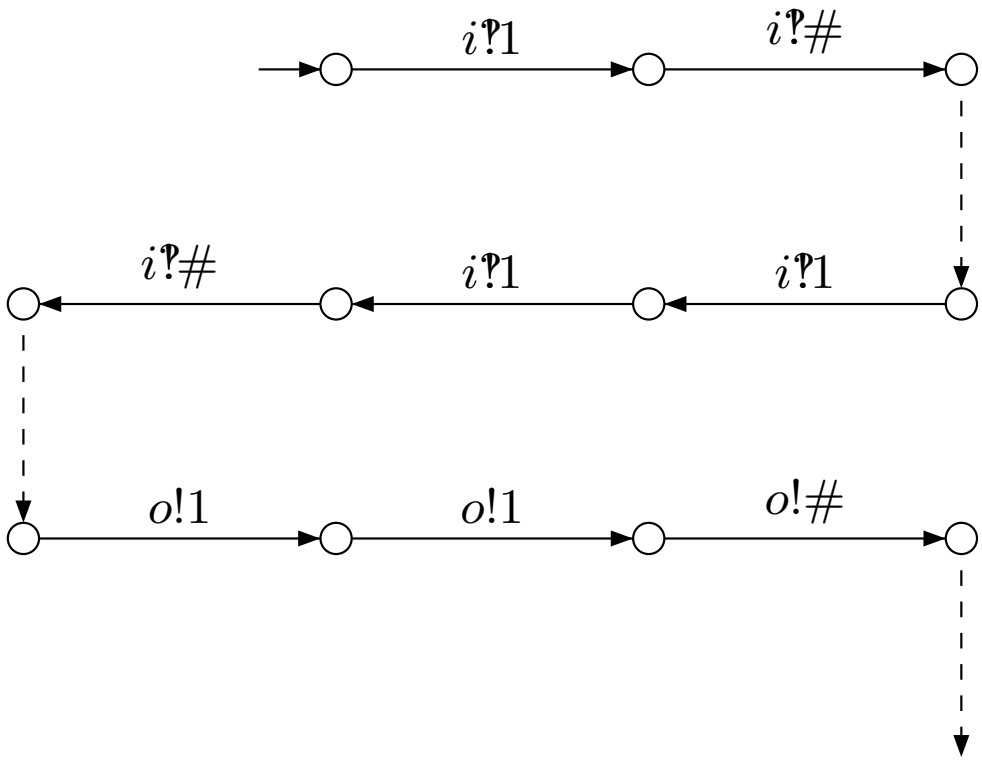




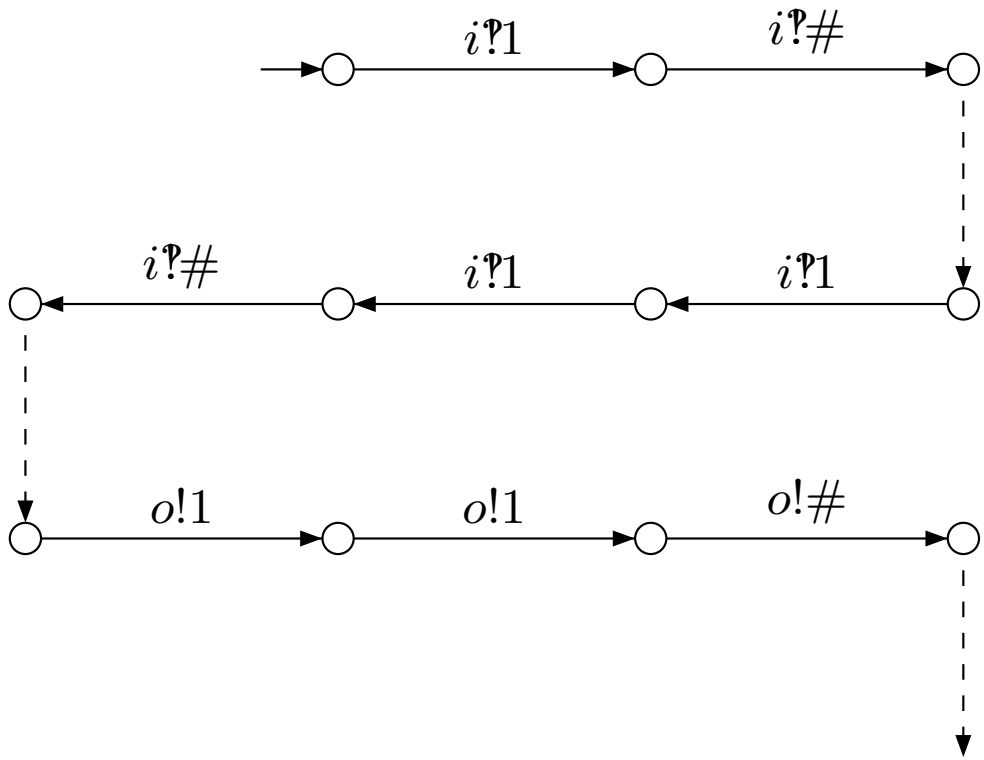
Reactive Turing Machines (4)



Reactive Turing Machines (4)



Reactive Turing Machines (4)



- The examples are deterministic RTM for simplification purposes

Technical Results

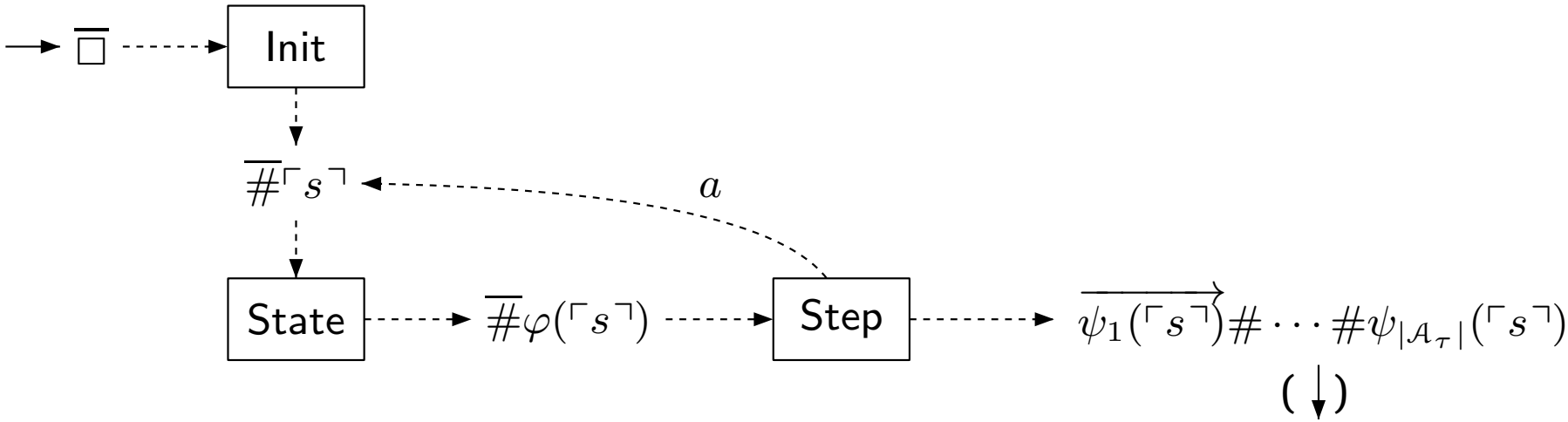
Expressivity

1. Effective transition systems are branching bisimilar with transition systems associated with with an RTM

Technical Results

Expressivity

1. Effective transition systems are branching bisimilar with transition systems associated with with an RTM
2. Deterministic computable transition systems are branching bisimilar with transition systems associated with with an RTM



Technical Results

Expressivity

1. Effective transition systems are branching bisimilar with transition systems associated with with an RTM
2. Deterministic computable transition systems are branching bisimilar with transition systems associated with with an RTM

Notes

- In one and only one state we have to make the choice!
- In case of bounded computable transition systems we can be divergence-sensitive!

Corollary

Parallelism does not add computational power

Technical Results

Expressivity

1. Effective transition systems are branching bisimilar with transition systems associated with with an RTM
2. Deterministic computable transition systems are branching bisimilar with transition systems associated with with an RTM
3. For every RTM there exists a finite recursive specification in ACP_τ such that the respective associated transition systems are branching bisimilar

Contributions

We have established

- ...a conceptual model of a computer
- ...that integrates computability and concurrency theory
- ...and implies the classical Turing machine

Related and Future Work

Related work

- Persistent Turing machines by Goldin, Smolka, Attie, Sonderegger
- Interactive Turing machines with advice by Van Leeuwen & Wiedermann
- ...

Future work

- Universal reactive Turing machine
- Variant definitions (e.g. different termination conditions)
- Relation with persistent Turing machine and interactive Turing machine with advice
- Relation with process calculi, e.g. π-calculus

Thank you!

Questions?