Reactive Turing Machines and ACP$_\tau$

Paul van Tilburg*
Eindhoven University of Technology, The Netherlands
p.j.a.v.tilburg@tue.nl

1 Introduction

The Turing machine [8] is widely accepted as a computational model suitable for exploring the theoretical boundaries of computing. Motivated by the existence of universal Turing machines, many textbooks on the theory of computation (e.g., [7, 5]) present the Turing machine as an accurate theoretical model of the computer. In fact, Sipser even goes as far as writing that "[a] Turing machine can do everything a real computer ran do." [7] This statement is sometimes referred to as the *strong* Church-Turing thesis, as opposed to the normal Church-Turing thesis according to which every *effectively calculable function* is computable by a Turing machine.

There are, however, limitations to using the Turing machine as a theoretical model of a computer. Computers have evolved into systems that no longer just perform computations but they interact with each other and their users; they are reactive systems. The classical Turing machine operates from the assumption that the input is available on the tape at the beginning, a terminating computation is performed, and the output is left on the tape at the end. However, it abstracts from two important aspects of modern day computing: interaction and non-termination.

We propose an extension of the classical Turing machine called the *reactive Turing machine* (RTM). To show that the RTM still can be used as a computational model, we show that effective transition systems can be simulated by reactive Turing machines up to branching bisimulation, although this introduces divergence. We also show that deterministic computable transition systems can be simulated without introducing divergence. We choose branching bisimulation, as it is the strongest equivalence used in concurrency theory. For a detailed definition and motivations to use branching bisimulation as the preferred notion of equivalence, see [9]. Finally, because we want to integrate automata and concurrency theory, we try to make the hidden interaction between finite control and tape in the Turing machine model explicit. We do this by giving a *finite recursive* ACP$_\tau$ *specification* of RTMs.

In earlier work, variations of the Turing machine were proposed to address interaction. This was done to investigate whether adding interaction leads to an increase in computational power. In [4], Goldon, Smolka, Attie and Sondregger introduce *persistent Turing machines* (PTMs). A PTM is a nondeterministic Turing machine with three tapes (input, output, working) where computations can be repeated retaining the contents of the working tape. Van Leeuwen and Wiedermann discuss interaction and Turing machines on a higher level [10, 11, 12]. They introduce the *interactive Turing machine with advice* as a Turing machine that can, when needed, access some advice function that allows for inserting external information into the computation. In contrary to the previously mentioned work, our goal is not to increase computational power of existing models but to obtain an accurate *conceptual model* of

*The work presented in this paper has been performed under the supervision of Jos Baeten and Bas Luttik. The research is supported by the project "Models of Computation: Automata and Processes" (nr. 612.000.630) of the Netherlands Organization for Scientific Research (NWO).

Submitted to:
YR-CONCUR 2010

the reactive, modern day computer. In this model we strive to integrate computability and concurrency theory.

For more elaborate definitions and proofs please refer to the full version of the paper, available online at: `http://www.win.tue.nl/~pvantilb/papers/RecEnumExecRTM_20100601.pdf`.

2 Reactive Turing machines

In the past, concurrency theory branched off from automata theory and focused on interaction. In our work we put the classical Turing machine in a process-theoretic setting to get a familiar notion of interaction, thereby enabling the theoretical model of the computer to be studied using well-known tools and results from both theories.

We fix a finite set of action labels $\mathcal{A}_\tau$ (including the silent step τ) and set of data $\mathcal{D}_\Box$ (including the blank symbol $\Box$). We adapt the conventional Turing machine [5, 7] as little as possible: all transactions will be labelled by an action from $\mathcal{A}_\tau$. Additionally, we allow for termination regardless of the contents of the tape by distinguishing a set of final (accepting) states. This leads to the following definition.

Definition 2.1. *A* reactive Turing machine *(RTM)* $\mathcal{M}$ *is defined as a quadruple* $(\mathcal{S},\rightarrow,\uparrow,\downarrow)$ *consisting of a finite set of* states $\mathcal{S}$*, a distinguished* initial state $\uparrow \in \mathcal{S}$*, a subset of* final states $\downarrow \subseteq \mathcal{S}$*, and a* $(\mathcal{D}_\Box \times \mathcal{A}_\tau \times \mathcal{D}_\Box \times \{L,R\})$*-labelled transition relation*

$$\rightarrow \subseteq \mathcal{S} \times \mathcal{D}_\Box \times \mathcal{A}_\tau \times \mathcal{D}_\Box \times \{L,R\} \times \mathcal{S} \ .$$

The intuitive meaning of a transition (s,d,a,e,M,t) is that whenever $\mathcal{M}$ is in state s and reads symbol d on the tape, it may execute the action a, write symbol e on the tape (replacing d), move the read/write head one position to the left or one position to the right on the tape (depending on whether $M = L$ or $M = R$), and then continue in state t. Note that the conventional Turing machine can be obtained by labelling all the transitions with the silent step τ.

A *tape instance* is a sequence $\delta \in (\mathcal{D}_\Box \cup \overline{\mathcal{D}_\Box})^*$, such that δ contains exactly one element of $\overline{\mathcal{D}_\Box}$ which denotes the set of *marked* tape symbols $\overline{\mathcal{D}_\Box} = \{\overline{d} \mid d \in \mathcal{D}_\Box\}$. A *configuration* of an RTM is a pair (s,δ) consisting of a state $s \in \mathcal{S}$, and a tape instance δ.

Before we formalise the intuition of transitions of an RTM it is convenient to introduce some notation to be able to concisely denote the new placement of the tape head marker. Let δ be an element of $\mathcal{D}_\Box^*$. Then by $\overleftarrow{\delta}$ (and $\overrightarrow{\delta}$) we denote the element of $(\mathcal{D}_\Box \cup \overline{\mathcal{D}_\Box})^*$ obtained by placing the tape head marker on the right-most (and left-most) symbol of δ if it exists, and $\overline{\Box}$ otherwise.

We can now associate a transition system $\mathcal{T}(\mathcal{M})$ with every RTM $\mathcal{M}$.

Definition 2.2. *Let* $\mathcal{M} = (\mathcal{S},\rightarrow_\mathcal{M},\uparrow_\mathcal{M},\downarrow_\mathcal{M})$ *be an RTM. The transition system* $\mathcal{T}(\mathcal{M})$ *associated with* $\mathcal{M}$ *is defined as follows:*

1. *its set of states is the set of configurations of* $\mathcal{M}$*;*
2. *its transition relation* $\rightarrow$ *is the least relation satisfying, for all* $a \in \mathcal{A}_\tau$*,* $d,e \in \mathcal{D}_\Box$ *and* $\delta_L,\delta_R \in \mathcal{D}_\Box^*$*:*

$$(s,\delta_L\overline{d}\delta_R) \xrightarrow{a} (t,\overleftarrow{\delta_L e\delta_R}) \text{ iff } s \xrightarrow{d,a,e,L}_\mathcal{M} t \ \text{, and conversely}$$
$$(s,\delta_L\overline{d}\delta_R) \xrightarrow{a} (t,\delta_L e\overrightarrow{\delta_R}) \text{ iff } s \xrightarrow{d,a,e,R}_\mathcal{M} t \ .$$

3. *its initial state is the configuration* $(\uparrow_\mathcal{M},\overline{\Box})$*; and*
4. *its set of final states is the set of terminating configurations* $\{(s,\delta) \mid s \in \downarrow_\mathcal{M}\}$.

3 Simulation of effective and computable transition systems

Next, we consider two well-known classes of transition systems and prove that they can be simulated by RTMs up to branching bisimulation (with explicit divergence).

Effective transition systems are transition systems for which there exists a coding to natural numbers of its states such that the transition relation and final state set are recursively enumerable.

We simulate effective transition systems on an RTM by putting the code of the state on the tape and either determining whether this state can terminate or randomly picking a natural number and testing whether this codes a valid transition from the current state. Since the transition relation and the final state set are recursively enumerable, there exist a procedure that terminates if a transition is valid or a state is indeed final. If this is not the case, then it may take infinite long. However, on each point in the procedures it is possible to abort and restart. This is necessary if we want to get the close correspondence of branching bisimilarity; we have to take care that no choice is made by a silent step. If the number codes a valid transition then the procedure terminates. Subsequently, after decoding, the action can be performed and the code of the next state will be on the tape.

Because these procedures can be aborted and restarted up until the moment an action is performed or termination occurs, no choice is made (by silent steps). Hence, we can abstract from all the silent steps and obtain the following result.

Theorem 3.1. *Every effective transition system is branching bisimilar with a transition system associated with an RTM.*

In our work we have also considered *computable transition systems*. A transition system is computable if it is finitely branching and there exits a coding to natural numbers of its states such that the final state set is decidable and there is a recursive function that lists the outgoing transitions.

Simulation on an RTM can be done in a similar way as described previously. However, we can now list the transitions and pick a specific one instead of having a possibly infinite procedure. Also the procedure that determines whether the state is a final state or not always terminates. Hence, simulation can be done without introducing divergence.

Theorem 3.2. *Every deterministic computable transition system is branching bisimilar with explicit divergence with a transition system associated with an RTM.*

Given the above theorems, we see that if we define the parallel (or sequential) composition of RTMs as the parallel (or sequential) composition of their associated transition systems, this does not add extra computational power. Since the respective composition of two effective (or computable) transition systems is again an effective (or computable) transition system, which can be simulated by a single RTM.

In our paper we present two different kinds of RTMs to deal with both classes. Iain Phillips has shown that effective transition systems can be reduced to computable transition systems in [6]. His reduction also introduces divergence.

4 ACP$_\tau$ specifications

In our model of the RTM (or the model of the conventional Turing machine) the interaction between the finite control and the tape is still implicit. In previous work we have made the interaction of finite control with a stack and bag for respectively pushdown and basic parallel processes explicit [2, 3]. We want to do the same for the RTM by giving a finite recursive specification for the finite control and for the memory, which in this case is the tape.

In [1] it is shown that every computable transition system is definable by an ACP$_\tau$ specification; the behaviour of a conventional Turing machine is simulated using some finite control and two stack processes, but it does not allow for intermediate termination. We model the tape using a queue process to allow for intermediate termination. We obtain the following result.

Theorem 4.1. *For every reactive Turing machine $\mathcal{M}$ there exists a finite recursive specification $E_\mathcal{M}$ such that $\mathcal{T}(\mathcal{M})$ is branching bisimilar with explicit divergence with $\mathcal{T}(E_\mathcal{M})$.*

Here, the finite recursive specification $E_\mathcal{M}$ contains a translation of the finite transition relation $\rightarrow_\mathcal{M}$ and a finite recursive specification of the queue. The initial variables of both parts are put in parallel to obtain the correspondence with $\mathcal{T}(\mathcal{M})$. We can combine this result with our results for effective and deterministic computable transition systems.

Corollary 4.2. *For every effective transition system L there exists a recursive specification E_L over* ACP$_\tau$ *such that L is branching bisimilar with $\mathcal{T}(E_L)$. Moreover, if L is computable and deterministic, then L is branching bisimilar with explicit divergence with $\mathcal{T}(E_L)$.*

References

[1] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. On the consistency of Koomen's fair abstraction rule. *Theoretical Computer Science*, 51:129–176, 1987.

[2] J. C. M. Baeten, P. J. L. Cuijpers, and P. J. A. van Tilburg. A context-free process as a pushdown automaton. In F. van Breugel and M. Chechik, editors, *Proceedings CONCUR'08*, number 5201 in Lecture Notes in Computer Science, pages 98–113, 2008.

[3] J. C. M. Baeten, P. J. L. Cuijpers, and P. J. A. van Tilburg. A basic parallel process as a parallel pushdown automaton. In D. Gorla and T. Hildebrandt, editors, *Proceedings EXPRESS'08*, number 242 in Electronic Notes in Theoretical Computer Science, pages 35–48, 2009.

[4] D. Q. Goldin, S. A. Smolka, P. C. Attie, and E. L. Sonderegger. Turing machines, transition systems, and interaction. *Inf. Comput.*, 194(2):101–128, 2004.

[5] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Pearson, 2006.

[6] I. C. C. Phillips. A note on expressiveness of process algebra. In G. L. Burn, S. Gay, and M. D. Ryan, editors, *Proceedings of the First Imperial College Department of Computing Workshop on Theory and Formal Methods*, Workshops in Computing, pages 260–264. Springer-Verlag, 1993.

[7] M. Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[8] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[9] R. J. van Glabbeek. What is Branching Time Semantics and why to use it? *Bulletin of the EATCS*, 53:190–198, 1994.

[10] J. van Leeuwen and J. Wiedermann. On algorithms and interaction. In M. Nielsen and B. Rovan, editors, *MFCS*, volume 1893 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2000.

[11] J. van Leeuwen and J. Wiedermann. A theory of interactive computation. In D. Q. Goldin, S. A. Smolka, and P. Wegner, editors, *Interactive Computation: The New Paradigm*, pages 119–142. Springer, 2006.

[12] J. Wiedermann and J. van Leeuwen. How we think of computing today. In A. Beckmann, C. Dimitracopoulos, and B. Löwe, editors, *CiE*, volume 5028 of *Lecture Notes in Computer Science*, pages 579–593. Springer, 2008.