Decidability issues for infinite-state processes – a survey

To appear in *Bulletin of the EATCS*

Søren Christensen
Datalogisk Afdeling
Matematisk Institut
Aarhus Universitet
Ny Munkegade
8000 Aarhus C
Denmark

Hans Hüttel
Afdeling for
Matematik og Datalogi
Aalborg Universitetscenter
Fredrik Bajersvej 7E
9220 Aalborg Ø
Denmark

November 21, 1996

Over the past decade much attention has been devoted to the study of process calculi such as CCS [Mil80, Mil89], ACP [BK84, BK88] and CSP [Hoa88] as the foundations of a mathematical model for nondeterminism and concurrency. Of particular interest has been the study of the behavioural semantics of these calculi as given by transition graphs arising from structural operational semantics in the tradition originated by [Plo81].

In this paper we survey recent developments and current trends within a new field of study within process algebra, namely that of decidability issues for processes with infinite transition graphs.

1 Behavioural equivalences

A particularly important question in process algebra as a whole is when processes can be said to *exhibit the same behaviour*. As a result, a plethora of *behavioural equivalences* exist today. The main rationale behind the various behavioural equivalences that have been proposed has been to capture behavioural aspects such as deadlock, livelock and causality that the language equivalence known from language theory does not take into account. For instance, consider the equality

$$ab + ac = a(b + c)$$

which holds for language equivalence, but is an identification that these other notions of behaviour and behavioural equivalence do not make, since the process expressions $ab + ac$ and $a(b + c)$ do not exhibit the same deadlock properties: after an initial a-action the former is only able to perform one action (b or c), whereas the latter has a choice between b and c.

Equivalences are usually classified according to their *coarseness*, i.e. how many identifications they make with respect to the branching behaviour of processes. This *linear time-branching time hierarchy* is illustrated in Figure 1 (after Rob van Glabbeek [vG90]) . The coarsest equivalences are the trace equivalence and the completed trace equivalence; the latter differs from trace equivalence in that only the *completed* trace languages are compared. This equivalence is therefore simply the language equivalence known from automata theory. Directly above we have the testing/failures equivalence investigated by Hennessy and deNicola et al. (see e.g. [Hen89]). At the top of the diagram is bisimulation equivalence (or bisimilarity), a notion introduced by Park in [Par81] and subsequently used by Milner and others in the CCS tradition (as exemplified in [Mil89]).

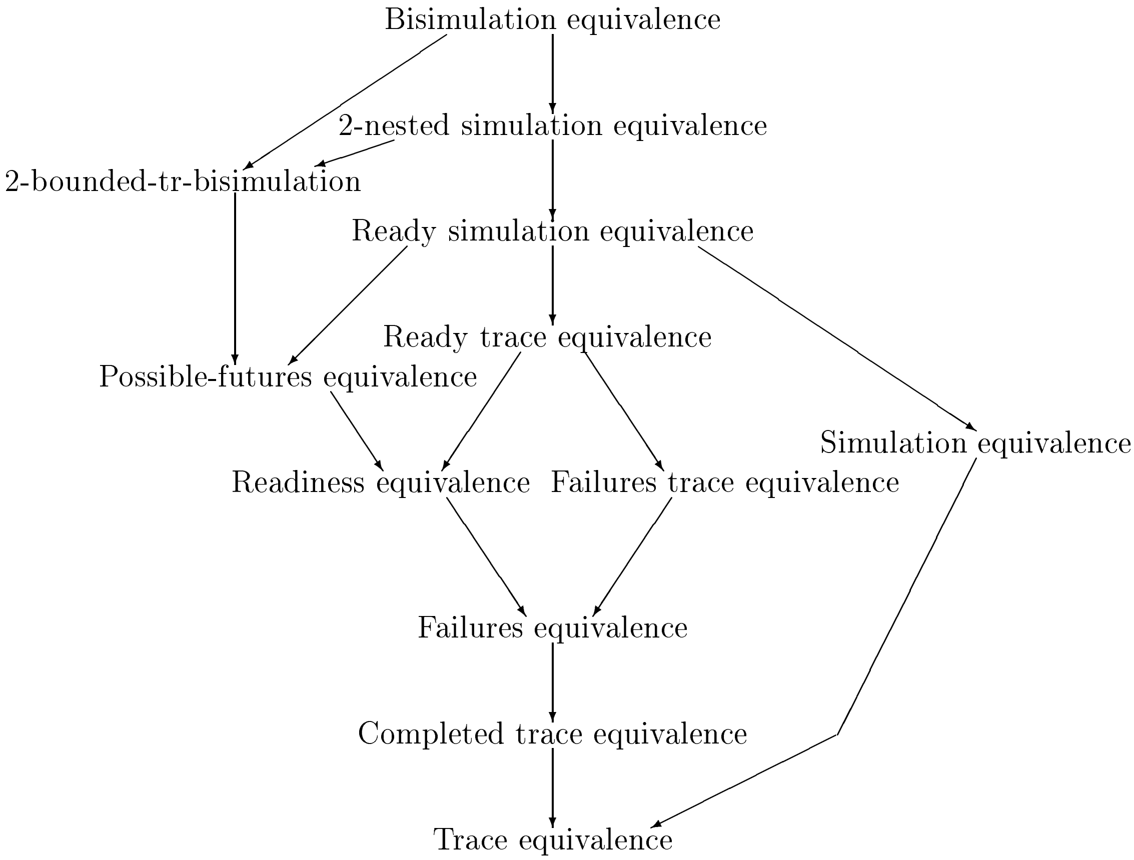

Figure 1: The linear time-branching time hierarchy of equivalences.

It has also been argued by some concurrency theoretists that behavioural equivalences should be judged according to how well they obey some computationally justifiable criterion of *observability*. For instance, while bisimulation equivalence has many nice mathematical properties it fails to have a computational justification in that (in)equivalence is not intuitively observable. Indeed, within the framework of testing some very unintuitive testing operators must be used. Abramsky has shown [Abr87] that bisimulation can be characterized by a test language that contains a testing operator that enumerates all next-states of the process subjected to testing. In [Gro89] Groote presents another test language; here lookahead in combination with the possibility to check for the absence of activity is needed. Bloom et al. argue [BIM90] that only completed traces should count as observations and define an equivalence which is a (completed) trace congruence under a 'reasonable' set of process constructs. This equivalence is the ready simulation equivalence of Figure 1.

The study of processes with unobservable (τ) actions leads to another linear/branching time hierarchy similar to Figure 1 except that we now have the corresponding *weak equivalences* based on the weak transition relation $\stackrel{a}{\Longrightarrow}$ where unobservable actions are disregarded. The best-known weak equivalence is Milner's weak bisimulation equivalence [Mil89]; however, it has been argued that this is not the proper weak version of bisimilarity since it does not reflect the changes in branching properties that may happen as the result of performing an unobservable action. Moreover, weak bisimulation is not robust under a simple notion of action refinement.

Finally, one should mention that there is a similar hierarchy of *preorders* on processes. Examples of such preorders include the simulation preorder [Par81], the testing/failures preorder [Hen89] as well as the ready simulation preorder of [BIM90].

2 Infinite-state systems

Milner [Mil84] has shown that the class of finite transition graphs corresponds to that of the transition graphs for *regular processes*, i.e. the recursively defined CCS processes over the signature $\{a., +\}$ where $a.$ is an action prefixing operator for every a in a set of atomic actions *Act* and $+$ is nondeterministic choice. Regular CCS processes correspond to the usual finite automata; their finitary trace languages are the finitary regular languages.

However, this result also says that as soon as we move beyond these constructs (known as the *dynamic* process constructs), recursively defined processes can have transition graphs with infinitely many states and trace languages that are no longer regular. This includes many realistic cases; in particular processes that are defined using various notions of parallel composition such as the asynchronous parallel operator $|$ of CCS.

Finite-state systems have been extensively studied, both within formal language theory and within the theory of process calculi. For finite-state systems all studied equivalences are known to be decidable, and sound and complete equational theories exist for various of these, see e.g. [Sal66, Mil84, Koz91, Rab92]. Furthermore, several automated tools have been designed for the analysis of finite-state systems (see e.g. [Mad92]). In fact, the theory of finite-state systems and their equivalences can now be said to be well-established. One may be led to wonder what the results will look like for *infinite-state* systems.

A major issue of pragmatic importance is that of providing decision algorithms for the various behavioural equivalences, algorithms that can be used in automated tools for verification. As all equivalences of Figure 1 are decidable for finite-state processes, the attention has focused on the *complexity* of the algorithms for the finite-state case.

However, it is also well known (see e.g. [HU79]) that language equivalence becomes undecidable when one moves beyond finite automata to context-free languages. A natural question is then whether the decidability can be extended beyond the finite-state case.

One should note that any decidability results must have their limitations. If the process language considered is sufficiently powerful, strong bisimulation equivalence ($\sim$) in a process language becomes undecidable. In fact, the general bisimilarity problem is not even r.e. For, using the CCS process operators of communication ($|$), restriction ($\backslash$) and action prefixing ($a.$) together with recursive definitions one can code any Turing machine M and input string w as a process expression $E_{M,w}$ such that all moves of M are represented by internal (τ-)actions of $E_{M,w}$ and such that the possible eventual halting is represented by a special success action. The encoding consists in expressing the tape of M as two stacks, one of which has been initialized to hold w. The stacks communicate with the finite control, represented by a regular process. The problem 'Does M diverge (i.e. have an infinite τ-computation) on input w ?' can then be expressed as

$$E_{M,w} \sim \mu x.(\tau.x) \tag{1}$$

where $\mu x.(\tau.x)$ stands for a process that performs a τ-action for ever. However, the problem (1) is undecidable as the above divergence problem is *not* r.e.

3 BPA – a decidable process calculus

The first example of an infinite-state process calculus that has a decidable equivalence problem is BPA (Basic Process Algebra) [BK88]. BPA processes are recursively defined processes over the signature $\{a, +, \cdot\}$ where a ranges over a set of atomic actions, $+$ is nondeterministic choice and $\cdot$ is sequential composition.

A BPA process is defined by a system of recursion equations of the form

$$X_0 \stackrel{\text{def}}{=} E_0(X_0 \ldots X_n) \quad \cdots \quad X_n \stackrel{\text{def}}{=} E_n(X_0 \ldots X_n)$$

where the E_i's are BPA expressions. A system of the above form where every occurrence of a variable in any expression E_i is within the scope of an atomic action is said to be *guarded*. Any guarded system of equations can effectively be put in the BPA equivalent of *Greibach Normal Form* (GNF), i.e. a system of equations where all equations are of the form $X_i \stackrel{\text{def}}{=} \sum_j a_{ij}\alpha_{ij}$ where the α_{ij}'s are compositions of process variables.

A special case is that of *normed BPA* processes. The *norm* of a process is defined as the least number of transitions necessary to terminate. A process is said to be normed if every state has a finite norm, i.e. you can terminate at any point during a computation. Even though normed BPA does not incorporate all regular processes, systems defined in this calculus can in general have infinitely many states.

There is an obvious correspondence between process equations of the form $X_i \stackrel{\text{def}}{=} \sum_j a_{ij}\alpha_{ij}$ and the GNF context-free productions $X \rightarrow a_{i1}\alpha_{i1} \mid \ldots \mid a_{ik}\alpha_{ik}$ – in other words, normed BPA processes correspond to context-free grammars without useless or empty productions. It is therefore easy to see that both trace equivalence and completed trace equivalence (or language equivalence) are undecidable for normed BPA processes.

But it also turns out that strong bisimilarity for normed BPA processes is *decidable*. This surprising fact was established by Baeten, Bergstra and Klop [BBK87]. Their proof consisted in showing that one can exhibit a decomposition of the process graph with certain regularities. Soon after this first proof, Didier Caucal gave a much shorter proof. The crux of his proof is a representation theorem for normed BPA. This states the maximal bisimulation for a normed BPA process is finitely representable by a confluent and strongly normalizing Thue system, and that there are only finitely many candidates for this Thue system.

The result was the first indication that we can go beyond finite-state systems while maintaining the decidability of bisimulation equivalence. Moreover, somewhat unexpectedly, we have gained something by using an equivalence different from language equivalence in a setting that involves structures from language theory.

Since then, Hans Hüttel and Colin Stirling have shown [HS91] how one can use a tableau technique inspired by the tableau techniques of modal logic for giving a decision procedure for the bisimulation problem. The idea is to give a set of goal-directed proof rules together with a number of termination conditions. The algorithm then essentially tries to construct a successful proof tree (a *tableau*) for the given initial equation. If the initial equation corresponds to a valid equality, then a proof tree can be built (the algorithm is *complete*). Conversely, if a proof tree can be built, then the initial equation must be valid (the algorithm is *sound*.) In this case, the proof tree yields a finite relation whose closure (in the appropriate sense) is a bisimulation containing the initial equation.

A further consequence of the algorithm sketched here is that we can obtain a *sound and complete axiomatization* for bisimulation equivalence for normed BPA. This axiom system is sequent-based, and a proof of an equation can be obtained by building a tableau for the equation and 'turning it upside down'.

The use of a tableau technique for equivalence problems is not new, though. It was used as early as in 1966 by Korenjak and Hopcroft [KH66] to show that the language equivalence problem for simple grammars is decidable[1]. The simple grammars are a class of context-free grammars that correspond to that of deterministic normed BPA processes. As the hierarchy of Figure 1 collapses for deterministic processes, the decidability result for normed BPA actually yields the Korenjak-Hopcroft theorem as a corollary.

The exact complexity of the bisimulation problem for normed BPA is not known at the time of writing, but Huynh and Tian [HT92] have shown that the bisimulation problem is in the polynomial-time hierarchy and is an element of Σ_2^p.

The question as to whether or not bisimulation is decidable for the full BPA calculus remained open for a number of years. But in [CHS92], Christensen, Hüttel and Stirling showed that the answer is yes, bisimulation is decidable for all BPA processes. The idea of the proof is to extend Caucal's finite representation result. This in turn yields semi-decision procedures for the positive and negative halves of the bisimulation problem. However, this decision algorithm does not support an equational theory for bisimilarity on BPA as was the case with the tableau system in the normed case.

But what about the other equivalences? Are they decidable, too? The answer is no. Huynh and Tian [HT90] proved that the failures and readiness equivalences are undecidable for normed BPA. Their proof consisted in giving a special class of normed BPA processes for which these equivalences coincide with language equivalence and then showing that the language equivalence problem for arbitrary normed BPA processes reduces to that for the special class. Later, Hans Hüttel and Jan Friso Groote [GH93] extended this undecidability result considerably, showing that *all* equivalences below bisimulation were in fact undecidable for normed BPA and thus for the full BPA calculus. This rests, among other things, on the fact that all these equivalences are defined in terms of the corresponding preorder; Emily Friedman's result [Fri76] that the language inclusion preorder is undecidable for simple grammars turns out to be crucial here.

[1] The notion of tableau systems arose in the area of logic with the work of Beth and Smullyan [Bet59, Smu68] but has its origin in Gentzen [Gen35] as "upside down" Gentzen sequent systems.

4 BPP – another decidable process calculus

Attention has since focused on the process calculus BPP (Basic Parallel Processes) [CHM93b]. The only difference between BPA and BPP is that a non-communicating parallel (full merge) operator takes the place of sequencing. BPP processes are thus recursively defined processes over the signature $\{a, +, \|\}$ where a ranges over a set of atomic actions, $+$ is nondeterministic choice and $\|$ is parallel composition without communication

Similar to BPA, a BPP process is defined by a system of recursion equations of the form

$$X_0 \stackrel{\text{def}}{=} E_0(X_0 \dots X_n) \quad \cdots \quad X_n \stackrel{\text{def}}{=} E_n(X_0 \dots X_n)$$

where the E_i's are now BPP expressions.

As in the case of BPA we say a system of BPP equations is *guarded* if every occurrence of a variable in any expression E_i is within the scope of an atomic action. In [Chr93] it is shown that any guarded system of BPP equations can effectively be put in a *normal form*. Here, a system of equations is in normal form if all equations are of the form $X_i \stackrel{\text{def}}{=} \sum_j a_{ij}\alpha_{ij}$ where the α_{ij}'s are *multisets* of process variables (the elements of the multiset correspond to concurrent components).

In [Chr93] it is demonstrated that w.r.t. language equivalence the two process calculi BPA and BPP are *incomparable*. For instance, the BPP process

$$X \stackrel{\text{def}}{=} a(X \parallel b \parallel c + b \parallel c) + b(X \parallel a \parallel c + a \parallel c) + c(X \parallel a \parallel b + a \parallel b)$$

cannot be language equivalent to any BPA process; and similarly, the BPA process $X \stackrel{\text{def}}{=} a(Xb + b)$ cannot be language equivalent to any BPP process.

In the setting of BPP Christensen, Hirshfeld and Moller have proved (see [CHM93b, CHM93a]) that bisimulation equivalence is decidable for the full BPP calculus by using a tableau technique similar to that used in [HS91]. As for the tableau system for normed BPA, a by-product of the decision procedure is a sound and complete sequent-style equational theory for bisimilarity. As BPP includes the regular processes this result may be seen as an extension of the complete axiomatisation of regular processes (Milner, [Mil84]).

Christensen has in [Chr93] extended the decidability result for BPP by replacing the pure merge operator $\|$ with the parallel composition of CCS thereby allowing for synchronisation between processes. Again the decision procedure is obtained via the use of tableaux and a by-product is a sound and complete sequent-based equational theory for bisimilarity

A natural question is whether the other equivalences within van Glabbeek's linear time-branching time hierarchy are decidable. Recently, Yoram Hirshfeld has shown that language equivalence is undecidable for BPP (see [Hir93]). The proof relies on showing that $ctr(E_1) \subseteq ctr(E_2)$ ($ctr(E)$ consists of the completed traces for E) if and only if $ctr(E_1 + E_2) = ctr(E_2)$ and then showing language inclusion to be undecidable on BPP. This, in turn, is achieved by a reduction from the Halting problem and relies on a construction somewhat related to Jančar's construction for proving bisimilarity to be undecidable on labelled Petri nets. Hirshfeld's construction also applies to show that trace equivalence as well as simulation equivalence are undecidable on BPP.

Later, Hüttel has extended Hirshfeld's result by showing that *none* of the equivalences below bisimilarity in the van Glabbeek hierarchy are decidable for BPP. For most of the equivalences, this is a direct consequence of the undecidability of ready simulation equivalence as this equivalence is near the top of the hierarchy. The reduction is based on an idea due to Hirshfeld [Hir93].

Thus the decidability records for the two (incomparable) process classes BPA and BPP are surprisingly similar: for both calculi, bisimilarity is decidable while all the other equivalences in the van Glabbeek hierarchy are undecidable.

5 Whither infinite transition graphs?

Lately, following work by Caucal and Monfort [CM90], interest in corresponding problems for pushdown automata has increased. PDAs turn out to have greater expressive power than context-free processes at the level of transition graphs, in the sense that there exist PDA transition graphs that are not bisimilar to any BPA transition graph. We are thus faced with a new question, namely whether bisimilarity is decidable for PDAs. At the time of writing, this question remains open. But recently Meitus [Mei93] has managed to show that language equivalence is in fact *decidable* for *deterministic* PDAs.

The class of BPP is naturally incorporated in a class of labelled Petri nets (see [Chr93]) and an obvious question is whether the decidability result for bisimilarity on BPP can be extended to the class of labelled Petri nets. However, this was shown by Jančar to be an unsolvable problem; he proved this (see [Jan93]) by a reduction from the Halting problem using Minsky two-counter machines. This result also shows that by adding a notion of *forced* synchronisation on top of BPP bisimilartity will become undecidable (see [Chr93]).

Of particular interest regarding BPP (with merge replaced by CCS parallel composition) is the decidability of weak bisimulation equivalence and observational congruence. In [Chr93] it is demonstrated that both of these equivalences are semi-decidable. However, it is not known how to obtain semi-decidability of the corresponding inequivalences and therefore the problem remains open.

Another important field of study is model checking for infinite transition graphs – can we decide whether a logical formula F is true at state s ? Burkart and Steffen [BS92] have answered this in the affirmative for context-free processes. This is in fact a consequence of the theorem by Muller and Schupp [MuS85] that the second-order monadic theory for PDA transition graphs is decidable.

References

[Abr87] S. Abramsky. Observational equivalence as a testing equivalence. *Theoretical Computer Science*, 53:225–241, 1987.

[BBK87] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. Technical Report CS-R8632, CWI, September 1987.

[Bet59] E.W. Beth. *The Foundations of Mathematics.* North-Holland Publishing Co., 1959.

[BIM90] B. Bloom, S. Istrail, and A. Meyer. Bisimulation can't be traced. Technical Report TR 90-1150, Cornell University, August 1990.

[BK84] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.

[BK88] J.A. Bergstra and J.W. Klop. Process theory based on bisimulation semantics. In J.W. de Bakker, W.P de Roever, and G. Rozenberg, editors, *LNCS 354*, pages 50–122. Springer-Verlag, 1988.

[BS92] O. Burkart and B. Steffen. Model checking for context-free processes. In W.R. Cleaveland, editor, *Proceedings of CONCUR '92*, pages 123–137. Springer-Verlag, LNCS 630, August 1992.

[CHM93a] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for all basic parallel processes. In *Proceedings of CONCUR '93.* Springer-Verlag, 1993.

[CHM93b] S. Christensen, Y. Hirshfeld, and F. Moller. Decomposability, decidability and axiomatisability for bisimulation equivalence on basic parallel processes. In *Proceedings of LICS '93.* IEEE, Computer Society Press, June 1993.

[CHS92] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. In W.R. Cleaveland, editor, *Proceedings of CONCUR '92, LNCS 630*, pages 138–147. Springer-Verlag, August 1992.

[Chr93] S. Christensen. *Decidability and Decomposition in Process Algebras.* Ph.D. Thesis, University of Edinburgh, 1993.

[CM90] D. Caucal and R. Monfort. On the transition graphs of automata and grammars. Technical report, IRISA, 1990.

[Fri76] E.P. Friedman. The inclusion problem for simple languages. *Theoretical Computer Science*, 1:297–316, 1976.

[Gen35] G. Gentzen. Untersuchungen über das Logischen Schliessen. *Mathematishe Zeitschrift*, Vol 39, pp 176–210, 1935.

[GH93] J.F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 1993. to appear.

[Gro89] J.F. Groote. Transition system specifications with negative premises. Report CS-R8950, CWI, 1989. An extended abstract appeared in J.C.M. Baeten and J.W. Klop, editors, *Proceedings of CONCUR 90*, Amsterdam, *LNCS 458*, pages 332–341. Springer-Verlag, 1990.

[Hen89] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1989.

[Hir93] Y. Hirshfeld. Finitely generated processes, petri nets and the equivalence problem. In *Proceedings of CSL 93*, 1993.

[Hoa88] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1988.

[HS91] H. Hüttel and C. Stirling. Actions speak louder than words: Proving bisimilarity for context-free processes. In *Proceedings of 6th Annual Symposium on Logic in Computer Science (LICS 91)*, pages 376–386. IEEE Computer Society Press, 1991.

[HT90] Dung T. Huynh and Lu Tian. On deciding readiness and failure equivalences for processes. Technical Report UTDCS-31-90, University of Texas at Dallas, September 1990.

[HT92] Dung T. Huynh and Lu Tian. Deciding bisimilarity of normed context-free processes is in $\sum_2^p$. Technical Report UTDCS-1-92, University of Texas at Dallas, January 1992. To appear in *Theoretical Computer Science*.

[HU79] J. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[Jan93] P. Jančar. Decidability Questions for Bisimilarity of Petri Nets and Some Related Problems. University of Edinburgh, LFCS report Series, ECS-LFCS-93-261, 1993.

[KH66] A.J. Korenjak and J.E. Hopcroft. Simple deterministic languages. In *Proceedings of Seventh Annual IEEE Symposium on Switching and Automata Theory*, pages 36–46, 1966.

[Mad92] E. Madelaine. Verification tools from the CONCUR project. *Bulletin of the EATCS*, 47, pp 110–126, June 1992.

[Mei93] Y. Meitus. Decidability of the equivalence problem for deterministic pushdown automata *Cybernetics and Systems Analysis*, 28:672–690, May 1993.

[Mil80] Robin Milner. *A Calculus of Communicating Systems, LNCS 92*. Springer-Verlag, 1980.

[Mil84] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.

[MuS85] Muller, D. and Schupp, P. The Theory of Ends, Pushdown Automata and Second Order Logic *Theoretical Computer Science*, 37:51–75, 1985.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings of 5th GI Conference LNCS 104*, pages 167–183. Springer-Verlag, 1981.

[Plo81] Gordon Plotkin. A structural approach to operational semantics. Technical Report FN-19, Computer Science Department, Aarhus University, 1981.

[vG90] R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, CWI/Vrije Universiteit, 1990.

[Koz91] D. Kozen. A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. In *Proceedings of LICS 91*, pp 214–225, IEEE Computer Society Press, 1991.

[Rab92] A. Rabinovich. A Complete Axiomatisation for Trace Congruence of Finite State Behaviours. Draft, Department of Computer Science, The University of Texas at El Paso, 1992.

[Sal66] A. Salomaa. Two Complete Axiom Systems for the Algebra of Regular Events. *Journal of the ACM* 13, pp 158–169, 1966.

[Smu68] R.M. Smullyan. *First-Order Logic*. Springer-Verlag, Berlin, 1968.