A regular viewpoint on processes and algebra[1]

Kamal Lodaya
The Institute of Mathematical Sciences
CIT Campus, Chennai 600 113, India
Google: Lodaya

Summary. While different algebraic structures have been proposed for the treatment of concurrency, finding solutions for equations over these structures needs to be worked on further.

This article is a survey of process algebra from a very narrow viewpoint, that of finite automata and regular languages. What have automata theorists learnt from process algebra about finite state concurrency? The title is stolen from [30].

There is a recent survey article [7] on finite state processes which deals extensively with rational expressions. The aim of the present article is different. How do standard notions such as Petri nets, Mazurkiewicz trace languages and Zielonka automata fare in the world of process algebra? This article has no original results, and the attempt is to raise questions rather than answer them.[2]

1 Formal languages

Formal language theory begins with the monoid of words $(\Sigma^*, \cdot, 1)$ over a finite alphabet Σ. A language is a set of words, and the algebraic structure of a set can be added to form an idempotent semiring $(\wp(\Sigma^*), \cdot, 1, +, 0)$. The identification of the semiring as a relevant algebraic structure is due to Conway [14] and Eilenberg [18].

Definition 1 *A **semiring** is a set S with an associative, commutative binary operation $+$ on S with identity 0; an associative binary operation $\cdot$ on S with identity 1 and absorbing element 0; and $\cdot$ distributing over $+$. The semiring is said to be **idempotent** if $+$ is idempotent.*

If we restrict ourselves to a regular language, recognized by a finite automaton, this amounts to saying that some equations hold in addition to those derived from the axioms of an idempotent semiring. Myhill and Nerode showed that recognizable languages, those saturated by finite-index congruences over the word monoid, are exactly the regular languages.

Kleene showed that the regular languages can be modelled by rational expressions, formed by adding to the signature an additional unary star operation forming the (Kleene) starred

[1]This article is based on the talk "Looking back at process algebra" given at the AFL '05 conference in Dobogókő. I take this opportunity to thank the organizers of the conference, Zoltán Ésik and Zoltán Fülöp, for their invitation and hospitality. I also thank Zoltán Ésik for his encouragement over the years.

[2]For some related questions in the world of process calculi, see [2].

(idempotent) semiring $(\wp(\Sigma^*), \cdot, 1, +, 0, {}^*)$. We will henceforth assume idempotence of $+$ in our algebraic structures. As is usual, we will omit $\cdot$ when writing expressions.

Chomsky's type 3 grammars are another formalism to describe regular languages, where one works with a system of tail-recursive equations over the semiring $S[V]$ with a set of variables V. The equations can be put in Greibach form and solved using Arden's rule [3] which says that, with the proviso $a \neq 1 + a$, the equation $x = ax + b$ has the solution $\mu x.(ax + b) = a^*b$, where $\mu : V \times S[V] \to S$ is a partial function giving a unique solution $\mu x.e$ to the equation $x = e$ when it exists. Formally we are in a (Chomsky) μ-semiring [20] $(\wp(\Sigma^*)[V], \cdot, 1, +, 0, \mu)$.

This solution procedure is the basis of the axiomatization of equality of rational expressions by Aanderaa [1] and Salomaa [45], using the "no empty word property" (NEWP), a syntactically checkable condition equivalent to $a \neq 1 + a$ over the semiring of regular languages. Here is Salomaa's axiomatization:

Axiom system S for starred semirings

(Assoc)	$(a + b) + c = a + (b + c); \quad (ab)c = a(bc)$
(Ident)	$a + 0 = a; \quad a1 = 1a = a$
(Comm)	$a + b = b + a$
(Idem)	$a + a = a$
(Absorp)	$a0 = 0a = 0$
(Distr)	$(a + b)c = ac + bc; \quad a(b + c) = ab + ac$
(Guard)	$a^* = (1 + a)^*$
(Fixpt)	$a^* = 1 + aa^*; \quad a^* = 1 + a^*a$
(GuardInd)	$\dfrac{x = ax + b}{x = a^*b}; \quad \dfrac{x = xa + b}{x = ba^*}$ (provided a has NEWP)

Kozen gives an equational treatment using axioms and inference rules [28] and identifies Kleene algebras (which we will not describe here) as the basic structure. The main property used, inspired by Conway [14], is that matrices over a Kleene algebra form a Kleene algebra. These matrices can be used to encode automata and constructions over them. Completeness is proved by reducing to isomorphism over the minimal deterministic finite automaton.

1.1 Concurrency

Definition 2 (Petri [44]) *A **Petri net** is a bipartite directed graph $\mathcal{N} = (P, T, F)$ where P and T are disjoint finite sets of places and transitions, and $F \subseteq (P \times T) \cup (T \times P)$ a flow relation. For a place or transition y, its pre-set $\{x \mid xFy\}$ is conventionally denoted ${}^\bullet y$ and its post-set $\{z \mid yFz\}$ is denoted $y^\bullet$. F satisfies the condition that for each transition t, ${}^\bullet t$ and $t^\bullet$ are nonempty, and for each place p, either ${}^\bullet p$ or $p^\bullet$ is nonempty.*

*A **marking** is a multiset of places. A transition t is **enabled** at marking M if ${}^\bullet t \subseteq M$. A transition t enabled at M "fires" taking M to $(M - {}^\bullet t) + t^\bullet$. Given an initial marking M_0, the net system $(\mathcal{N}, M_0)$ is said to be **1-safe** if every reachable marking is a* set *(hence multisets are not required).*

The "firing sequences" of nets (words over the alphabet T) have been investigated thoroughly from the formal language viewpoint. For instance, since we have not introduced any notion of a final marking, the language accepted by a net system is prefix-closed. In the firing sequence view, nets are seen as no more than a representation of automata which have concurrent behaviour. The marking graph of a 1-safe net system, with vertices the reachable markings and edges representing the firing relation, is in fact a finite automaton. Concurrency is modelled as the shuffle or interleaving of two languages, for which rational expressions are sufficient since rationality is preserved by the shuffle.

But rational expressions are certainly not succinct for concurrent behaviour. The shuffle expression $a||b||c$ has equivalent rational expression $abc + acb + bac + bca + cab + cba$ (this is an instance of Milner's expansion axiom from CCS [33]), which shows that a shuffle can be exponentially succinct. A net for this language is exponentially succinct compared to the corresponding automaton.

The operating systems community continually had to deal with concurrent behaviour and were alive to this problem. They developed `cobegin-coend` [17], path expressions [13], and the languages COSY and CSP (fully described in the later books [27, 26]). The signature of rational expressions was expanded by binary shuffle operations $\{||_C \mid C \subseteq \Sigma\}$, with intersection over the letters in C. The intersection comes in handy to represent synchronization between concurrent processes.

Definition 3 (Grabowski [23]) *The **series-rational expressions** over an alphabet Σ consist of the atomic actions $a \in \Sigma$ and the constants 1 and 0, closed under the binary operations $\cdot, +, ||$ and the unary operation *.*

The shuffle operator $e_1||e_2$ is now redefined to additionally act as intersection whenever an action is shared between e_1 and e_2. In the term algebra generated from Σ, these expressions still describe languages over starred semirings, since a Milner-like expansion axiom can be used to eliminate the shuffle operations.

There is a translation from series-rational expressions to 1-safe net systems which preserves succinctness. The later work of Garg and Ragunath [21], when restricted to 1-safe nets, provides a method of going from net systems to these expressions (with the notable addition of renaming functions) when a distribution of places of a net is provided.

Grabowski [23] provided an interpretation of series-rational expressions over labelled posets (or "pomsets" as Pratt called them). A net can be seen as accepting a poset language, and Grabowski provided a two-way translation between 1-safe nets and series-rational expressions with renaming (including crucially renaming to the empty poset), representing regular poset languages. But posets are difficult to put into an algebraic framework. A popular representation of these posets which is closer to usual formal language theory is as Mazurkiewicz traces, to which we now turn.

2 Trace languages

Let I be an irreflexive symmetric relation over Σ, called independence, and let its reflexive transitive closure be $\sim_I$, called trace congruence. For instance, if aIb then $wabx \sim_I wbax$ (a and b commute). Sometimes it is convenient to consider the complementary symmetric dependence relation instead of independence.

Definition 4 (Mazurkiewicz [31]) *A **trace** over the concurrency alphabet (Σ, I) is a word over the partially commutative monoid $(\Sigma^*/\sim_I, \cdot, 1)$. Trace concatenation $\cdot$ works on the congruence classes. A trace language is a set of traces.*

Trace languages form the trace semiring $(\wp(\Sigma^*/\sim_I), \cdot, 1, +, 0)$ where the commutativity equations $ab = ba$ are added for every pair a, b in the independence relation. Hence only one representative of a trace needs to be described, the others being inferred, and we regain succinctness. We need not restrict ourselves to the term algebra, and the shuffle operations are not needed.

A 1-safe Petri net has a natural independence relation on its transitions: they are independent if their neighbourhoods are disjoint. This is a necessary condition for concurrent behaviour but not sufficient. The firing traces of a finite 1-safe net system are defined by quotienting the firing sequences with this independence relation. The set of firing traces form a recognizable trace language; that is, it is saturated by a finite-index congruence over the partially commutative monoid defined by (Σ, I). Again, because of the lack of final markings, the language will be prefix-closed.

Extend the independence relation to words: for nonzero w and x, let wIx iff every letter in w is independent with every letter in x. w and x are said to be connected if they are not independent. This syntactically checkable condition can be inductively lifted to rational expressions. Assuming that a and b are independent, we can derive $ab = ba = ab + ba$ by using idempotence. $a^*b^* = 1 + aa^*b^* + a^*bb^* = 1 + (a+b)a^*b^* = (a+b)^*$. The axiom system S is used in the first step and again in the last step, which is an application of the (GuardInd) rule.

Ochmański realized that it is sufficient to take the trace closure $[e^*]$ of the usual Kleene e^* over connected expressions e—that is, e and every starred subterm of e is connected [42]. If e does not satisfy this condition, Ochmański defined the concurrent star as described by the axiom below.

Now the trace languages (not just the prefix-closed ones) form an (Ochmański) starred trace (idempotent) semiring $(\wp(\Sigma^*/\sim_I), \cdot, 1, +, 0, ^*)$. However, no axiomatization for equality of recognizable trace languages is known.

Axioms TS for starred trace semirings

(S)	All valid equalities for starred semirings
(Comm)	$ab = ba$, provided a and b are independent
(CStar)	$(ab + c)^* \stackrel{\text{def}}{=} (a + b + c)^*$, if a and b are independent

Assume that a, b and c are independent. By iterating the derivation $((a+b)^*c+d)^* = (a^*b^*c+d)^* \stackrel{\text{def}}{=} (a^*+b^*+c+d)^* = (a+b+c+d)^*$, where the first step was derived above and the last step uses the S system, the Ochmański star can be reduced to the Kleene star over connected expressions.

Question 1 *Is equality of trace languages over a given concurrency alphabet, described by rational expressions, recursively enumerable?*

Question 2 *Is there a complete axiomatization for rational trace languages over a concurrency alphabet?*

Here is a proof attempt which gets stuck.

Fix a total order over the letters of the alphabet and extend it lexicographically to words. Each trace can be represented by its lexicographically minimal word. Let Lex be the set of lexicographically minimal words. For a rational trace language TL, $Lex(TL) = Lex \cap (\bigcup TL)$ is a rational word language.

Suppose expressions a and b denote the same rational trace language $TL(a) = TL(b)$. By another theorem of Ochmański [16], there are connected rational expressions e and f whose word languages $WL(e) = Lex(TL(a))$ and $WL(f) = Lex(TL(b))$ are the same, and the trace closures are $[WL(e)] = TL(a)$ and $[WL(f)] = TL(b)$. By completeness of Salomaa's axiomatization, the equality $e = f$ is provable in S, and hence in TS. If we could show for a connected rational expression e that if e describes $Lex(TL(a))$, then $e = a$ is provable in TS, we could prove $a = b$ in TS and obtain its completeness. We do not have such an argument.

2.1 Distributed automata

A suitable automaton model which matches recognizability was defined by Zielonka [47]. Let Loc be a finite set of "locations", and $loc : \Sigma \to \wp(Loc)$ map each action to the locations required for executing it. Thus the alphabet Σ is distributed across the locations; if an action requires more than one location, we think of it as a synchronization between the distributed locations. A word language is said to be Loc-consistent if it is closed under commutation, where the actions a and b commute ($wabx \sim_{Loc} wbax$) if they are not shared by more than one location in Loc. Trace languages and Loc-consistent word languages are essentially the same thing.

Definition 5 (Zielonka [47]) *Let Q be a set of states distributed by the function $dist : Q \to Loc$. For $L \subseteq Loc$, let $\Pi_L Q$ be the functions $f : L \to Q$ such that $dist(f(i)) = i$. A **Zielonka automaton** over the distributed alphabet (Σ, Loc) is given by $(Q, dist, q_0, \to, F)$, where $q_0 \in \Pi_{Loc} Q$ is a distributed initial state and $F \subseteq \Pi_{Loc} Q$ a set of distributed final states, and $\to = \bigcup_{a \in \Sigma} \{\to_a \subseteq \Pi_{loc(a)} Q \times \Pi_{loc(a)} Q\}$ is a transition relation.*

Zielonka automata are automata distributed over locations. The states are local, the transitions act on exactly those locations which an action is declared to require, and the final states are global. A run of a Zielonka automaton is defined over global states, every action transforming the states of the locations it affects, the other states remaining fixed.

Zielonka [47] showed that the regular trace languages, those accepted by his automata, match the recognizable trace languages. Our notation for the automata follows Mukund and Sohoni [38], who provided an alternate proof of Zielonka's theorem by defining a gossip framework which explicitly represents state information shared across locations.

Thus trace theory [16] neatly generalizes formal language theory with regular trace languages playing a pivotal role. Mohalik and Ramanujam [37] provide a framework for *Loc*-consistent regular languages and a variant of series-rational expressions using special labelling functions, which provide a local presentation of distributed automata.

Question 3 *Is there an equational treatment of distributed automata in a Kleene algebra-like framework?*

3 Process calculi

We now turn to what Pnueli called the viewpoint of "reactive" systems: viewing automata in a concurrent environment not just as language generators but as processes. The classic vending machine example [26] shows that processes describe branching behaviour, and hence the left-distributivity axiom $a(b + c) = ab + ac$ for language equivalence fails. Some of the early models include failure sets, testing equivalences, synchronization trees and bisimulation [12, 15, 33, 43].

Definition 6 (Benson and Tiuryn [6]) *A* ***grove*** *is a set G with an associative, commutative binary operation $+$ with identity 0, an associative binary operation $\cdot$ with 0 a left zero, and where $\cdot$ right-distributes over $+$, that is, $(a+b)\cdot c = a\cdot c+b\cdot c$. A grove is* ***idempotent*** *if $+$ is idempotent. A* ***μ-grove*** *$(G[V], \cdot, +, 0, \mu)$ with a set of variables V has a partial solution function $\mu : V \times G[V] \to G$ analogous to a μ-semiring [20].*

A grove is defined by dropping the monoid identity, the right-absorption of 0 for multiplication and the left-distributivity of multiplication over addition from the axioms of a semiring. We will use μ-groves $G_\Sigma[V]$ generated from an alphabet Σ and a set of variables V as our basic model. (As before, we assume idempotence of $+$ in our structures.) Idempotent μ-groves are closely related to the axiomatization of bisimulation equivalence. Bloom and Ésik's monograph [10] provides a detailed description.

The first process calculus, Robin Milner's CCS, was published in 1980 [33]. Of course, CCS was based on a lot of earlier work, and Milner himself had been developing the idea for a few years, but *LNCS* 92 is the first fully developed treatment.

Milner proved a striking early result in process algebra [34], showing that tail-recursive equations (or guarded μ-expressions in his terminology) interpreted over μ-groves are sufficient to describe branching behavior of finite automata, whereas rational expressions over Kleene starred groves are not.

Axiom system M for μ-groves

(Assoc)	$(a+b)+c = a+(b+c); \quad (a\cdot b)\cdot c = a\cdot(b\cdot c)$
(Comm)	$a+b = b+a$
(Idem)	$a+a = a$
(Ident)	$a+0 = a$
(LeftAbs)	$0\cdot a = 0$
(RightDistr)	$(a+b)\cdot c = (a\cdot c)+(b\cdot c)$
(Guard)	$\mu x.e = \mu x.(x+e)$
(Fixpt)	$\mu x.e = e[\mu x.e/x]$
(GuardInd)	$\dfrac{f = e[f/x]}{f = \mu x.e}$ (provided x guarded in e)

The existence of unique solutions over certain groves was proved by Bergstra and Klop [8]. They also extended the positive result to automata with silent transitions [9], which was later developed by Milner in [35]. Since a finite system of tail-recursive equations implicitly defines a finite-index congruence on a finitely generated free grove, the negative result led to various kinds of extended star operations to restore the syntactic treatment known for rational languages. They are described in the survey article [7] mentioned in the introduction.[3]

3.1 Concurrency

Representing concurrency as interleaving of atomic actions, the shuffle operators can be added on since the expansion axioms are sound over groves. This yields the framework of process calculi [5]—PA and ACP for shuffles without and with synchronization respectively. Within a term model the shuffles can again be eliminated.

Bravetti and Gorrieri [11] extended Milner's axiomatization of regular behaviour to strongly guarded μ-expressions over Σ with shuffle, that is, those which are in Greibach form and do not allow a shuffle operation inside a recursion. The following question is still open:

Question 4 *Is there a direct way of going from finite 1-safe Petri nets to strongly guarded μ-expressions with shuffle, without incurring an exponential blowup?*

One approach may be to work with a "concurrent" bisimulation, as for example in [40]. Van Glabbeek and Vaandrager [22] proposed to axiomatize such a bisimulation by dropping the expansion axiom while retaining some desirable properties of the shuffle such as commutativity and associativity. That is, they expand groves with a shuffle operator $(G_\Sigma[V], \cdot, +, 0, ||, \mu)$. The shuffle is not reducible to the other operators.

[3]The paper [4] provides a recent update on Milner's results and questions.

Axiom system SM for μ-groves with shuffle

(M)	All axioms of M
(Assoc$\|\|$)	$(a\|\|b)\|\|c = a\|\|(b\|\|c)$
(Comm$\|\|$)	$a\|\|b = b\|\|a$
(Ident$\|\|$)	$a\|\|0 = a$
(Distr$\|\|$)	$(a+b)\|\|c = (a\|\|c) + (b\|\|c)$
(StGuardInd)	$\dfrac{f = e[f/x]}{f = \mu x.e}$ (provided x strongly guarded in e)

Question 5 *Is there a complete axiomatization of concurrent bisimulation over finite state processes?*

3.2 Mobility

Process theory research seems to be moving more in the direction of value-passing [24] and mobile processes [19, 36], which are described by π-expressions upto a value-passing bisimulation, which comes in "early" and "late" variants to model eager and lazy forms of evaluation. We do not provide details of the syntax here.

Finite-control mobile systems model a state as an edge-labelled graph, where the nodes ("agents") have local storage to save some values and the edges ("links") communicate these values between the agents. Further, the values communicated are the link names themselves. Hence the atomic actions are of the form $c!v$ and $c?x$, sending a value v on a link c or receiving it in a variable x. To describe these systems, we allow tail-recursion in π-expressions, but disallow the replication operator which is sufficiently powerful to model general recursion. Effectively the syntax reduces to guarded μ-expressions *with parameters and a calling mechanism* built over an alphabet of atomic expressions with constants and variables (and with a shuffle, which is eliminable in a term algebra).

Milner's axiomatization has been extended to the value-passing bisimulations by Hennessy, Lin and Rathke [25] for finite-control systems described by tail-recursive π-expressions. However the underlying algebraic structure is far from clear. It appears to be some kind of combinatory grove, as illustrated by the communication axiom, which is based on the β-rule of λ-calculus:

$(c!v \cdot P)||(c?x \cdot Q) = P||(Q[v/x])$.

Question 6 *Can one describe the algebraic structure of mobile systems?*

3.3 Event structures

Definition 7 (Nielsen, Plotkin and Winskel [39]) *A **(Σ-labelled) event structure** $(E, \leq, \#, \ell)$ is a (Σ-labelled) poset $(E, \leq, \ell)$ with an irreflexive symmetric conflict relation $\#$ which is "inherited"; that is, if two events $e_1, e_2 \in E$ are in conflict, all events $e_1' \geq e_1$ and $e_2' \geq e_2$ above them are also in conflict. A configuration of an event structure is a downward-closed conflict-free set of events.*

Event structures are a generalization of traces or labelled posets to include branching behaviour. Events can be related by causality ($\leq$ or $\geq$), conflict ($\#$), or by neither causality nor conflict, in which case we say they are concurrent.

Configurations are a notion of "state" in an event structure. For the purposes of finite state behaviour, it is sufficient to restrict oneself to event structures which are finitary, where each event has a finite number of events below it, and have bounded enabling, that is, each configuration can be extended by a bounded number of immediately enabled successor events. In particular, this will mean that all configurations of interest are finite sets of events, and the conflict relation will be generated from an immediate conflict relation. We henceforth assume our event structures satisfy these properties.

We now lift some definitions from infinite trees.

Definition 8 (Thiagarajan [46]) *The residue of a configuration in an event structure is those events strictly above it. Two configurations are said to be right invariant if their residues are isomorphic as event structures. An event structure is recognizable if the right invariance relation on its configurations is of finite index.*

Although configurations are finite, residues can very well be infinite. The concurrent branching behaviour of a 1-safe Petri net can be defined by "unfolding" it; Thiagarajan proves that this yields a special kind of event structure.

Call an event structure deterministic if at any of its configurations, for any letter of the alphabet, at most one event labelled by that letter is enabled.

Definition 9 (Thiagarajan [46]) *A deterministic Σ-labelled event structure is said to be a **trace event structure** if there is an (irreflexive symmetric) independence relation over Σ such that the labels of concurrent events are independent, and the labels of neighbouring events (related by the immediate successor relation or immediate conflict relation) are dependent.*

Theorem 10 (Thiagarajan [46]) *An event structure is the unfolding of a 1-safe Petri net if and only if it is a recognizable trace event structure.*

The proof of the right-to-left direction goes via Zielonka's theorem.

Petri nets as we have defined them are not sufficiently abstract, since their behaviour is described in terms of the transitions T. Even a finite language like $\{a, aa\}$ is not representable. Hence one should start with a *labelled* 1-safe Petri net (P, T, F, ℓ), $\ell : T \to \Sigma$. Unfolding such a net certainly yields a recognizable labelled event structure, but it may no longer be deterministic.

Question 7 *Is the converse also true? Is a recognizable labelled event structure the unfolding of a labelled 1-safe Petri net?*

Thiagarajan [46] conjectured that the answer is yes. The conjecture has been proved for conflict-free event structures [41], where the conflict relation is empty; sequential event structures, which have no concurrency [41]; and deterministic event structures [29]. The general case is still open.

The reliance on determinism amounts, in the algebraic setting, to left-distributivity. So the basic algebraic structure is that of a semiring, or a trace semiring in the case of a trace event structure. Like posets, event structures are not well suited for algebra, and groves might be better to work with. Thiagarajan's conjecture leads one to ask the following:

Question 8 *Given a finite-index congruence over an idempotent grove with shuffle, is there a direct way of constructing a finite 1-safe Petri net which satisfies this particular behaviour?*

A categorical structure suitable for Petri nets has been proposed by Meseguer and Montanari [32]. A similar question can be raised in that setting.

References

[1] S. Aanderaa. On the algebra of regular expressions, in *Appl. Math. (course notes)*, Harvard, Jan 1965, 1–18.

[2] L. Aceto. Some of my favourite results in classic process algebra, *Bull. EATCS* 81, Oct 2003, 89–108.

[3] D.N. Arden. Delayed logic and finite state machines, in *Theory of computing machine design (course notes)*, U. Mich., Ann Arbor, 1960, 1–35.

[4] J.C.M. Baeten and F. Corradini. Regular expressions in process algebra, *Proc. LICS*, Chicago, IEEE, 2005, 12–19.

[5] J.C.M. Baeten and W.P. Weijland. *Process algebra*, CUP, 1990.

[6] D.B. Benson and J. Tiuryn. Fixed points in free process algebras I, *TCS* 63(3), 1989, 275–294.

[7] J. Bergstra, W. Fokkink and A. Ponse. Process algebra with recursive operations, in *Handbook of process algebra* (J. Bergstra, A. Ponse and S.A. Smolka, eds.), Elsevier, 2001, 333–389.

[8] J. Bergstra and J.W. Klop. Fixed point semantics in process algebra, *Report* IW 206/82, Centre for Mathematics and Computer Science, Amsterdam, 1982.

[9] J. Bergstra and J.W. Klop. A complete inference system for regular processes with silent moves, *Proc. Logic Colloquium*, Hull (F. Drake and J. Truss, eds.), North-Holland, 1986, 21–81.

[10] S. Bloom and Z.Ésik. *Iteration theories: the equational logic of iterative processes*, Springer, 1993.

[11] M. Bravetti and R. Gorrieri. Deciding and axiomatizing weak ST bisimulation for a process algebra with recursion and action refinement, *ACM TOCL* 3(4), 2002, 465–520.

[12] S.D. Brookes, C.A.R. Hoare and A.W. Roscoe. A theory of communicating sequential processes, *JACM* 31(3), 1984, 560–599.

[13] R.H. Campbell and A.N. Habermann. The specification of process synchronization by path expressions, in *Proc. Operating Systems conference* (E. Gelenbe and C. Kaiser, eds.), LNCS 16, 1974, 89–102.

[14] J.H. Conway. *Regular algebra and finite machines*, Chapman and Hall, 1971.

[15] R. De Nicola and M. Hennessy. Testing equivalences for processes, *TCS* 34, 1984, 83–133.

[16] V. Diekert and G. Rozenberg, eds. *The book of traces*, World Scientific, 1995.

[17] E.W. Dijkstra. Cooperating sequential processes, in *Programming languages* (F. Genuys, ed.), Academic Press, 1968.

[18] S. Eilenberg. *Automata, languages and machines* A, Academic Press, 1974.

[19] U.H. Engberg and M. Nielsen. A calculus of communicating systems with label-passing, Report DAIMI PB-208, Aarhus University, 1986.

[20] Z. Ésik and H. Leiß. Algebraically complete semirings and Greibach normal form, *Ann. Pure Appl. Logic* 133, 2005, 173–203.

[21] V.K. Garg and M.T. Ragunath. Concurrent regular expressions and their relationship to Petri nets, *TCS* 96(2), 1992, 285–304.

[22] R.J. van Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency, *Proc. PARLE* 2, Eindhoven (J.W. de Bakker, A.J. Nijman and P.C. Treleaven, eds.), LNCS 259, 1987, 224–242.

[23] J. Grabowski. On partial languages, *Fund. Inform.* IV(2), 1981, 427–498.

[24] M. Hennessy and A. Ingólfsdóttir. A theory of communicating processes with value-passing, *Inf. Comput.* 107(2), 1993, 202–236.

[25] M. Hennessy, H. Lin and J. Rathke. Unique fixpoint induction for message-passing process calculi, *Sci. Comput. Program.* 41(3), 2001, 241–275.

[26] C.A.R. Hoare. *Communicating sequential processes*, Prentice-Hall, 1985.

[27] R. Janicki and P.E. Lauer. *Specification and analysis of concurrent systems: the COSY approach*, Springer, 1992.

[28] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events, *Inf. Comput.* 110(2), 1994, 366–390.

[29] K. Lodaya. Petri nets, event structures and algebra, in *Formal models, languages and applications* (K.G. Subramanian, K. Rangarajan and M. Mukund, eds.), World Scientific, to appear, 2006.

[30] D. Lugiez and Ph. Schnoebelen. The regular viewpoint on PA-processes, *TCS* 274(1–2), 2002, 89–115.

[31] A. Mazurkiewicz. Concurrent program schemes and their interpretations, Report DAIMI PB-78, Aarhus University, 1977.

[32] J. Meseguer and U. Montanari. Petri nets are monoids, *Inf. Comput.* 88 (1990) 105–155.

[33] R. Milner. *A calculus of communicating systems*, LNCS 92, 1980.

[34] R. Milner. A complete inference system for a class of regular behaviours, *JCSS* 28(3), 1984, 439–466.

[35] R. Milner. A complete axiomatisation for observation congruence of finite-state behaviours, *Inf. Comput.* 81(2), 1989, 227–247.

[36] R. Milner, J. Parrow and D. Walker. A calculus of mobile processes I and II, *Inf. Comput.* 100(1), 1992, 1–77.

[37] S. Mohalik and R. Ramanujam. Distributed automata in an assumption-commitment framework, *Sādhanā* 27, Part 2, 2002, 209–250.

[38] M. Mukund and M. Sohoni. Keeping track of the latest gossip in a distributed system, *Distr. Comp.* 10(3), 1997, 117–127.

[39] M. Nielsen, G. Plotkin and G. Winskel. Petri nets, event structures and domains I, *TCS* 13 (1980) 86–108.

[40] M. Nielsen and P.S. Thiagarajan. Degrees of nondeterminism and concurrency: a Petri net view, *Proc. FSTTCS*, Bangalore (M. Joseph and R.K. Shyamasundar, eds.), LNCS 181, 1984, 89–117.

[41] M. Nielsen and P.S. Thiagarajan. Regular event structures and finite Petri nets: the conflict-free case, *Proc. ICATPN*, Adelaide (J. Esparza and C. Lakos, eds.), LNCS 2360, 2002, 335–351.

[42] E. Ochmański. Regular behaviour of concurrent systems, *Bull. EATCS* 27, 1985, 56–67.

[43] D. Park. Concurrency and automata on infinite sequences, *Proc. 5th GI conference*, Karlsruhe (P. Deussen, ed.), LNCS 104, 1981, 167–183.

[44] C.-A. Petri. Fundamentals of a theory of asynchronous information flow, *Proc. IFIP*, Munich (C.M. Popplewell, ed.), North-Holland, 1962, 386–390.

[45] A. Salomaa. Two complete axiom systems for the algebra of regular events, *JACM* 13(1), 1966, 158–169.

[46] P.S. Thiagarajan. Regular trace event structures, BRICS Research Abstracts RS-96-32, 1996.

[47] W. Zielonka. Notes on finite asynchronous automata, *RAIRO Inf. Th. Appl.* 21(2), 1987, 99–135.