Ensuring Liveness Properties of Distributed Systems
(A Research Agenda)

Rob van Glabbeek
NICTA*, Sydney, Australia
School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
rvg@cs.stanford.edu

Often fairness assumptions need to be made in order to establish liveness properties of distributed systems, but in many situations these lead to false conclusions.

This document presents a research agenda aiming at laying the foundations of a theory of concurrency that is equipped to ensure liveness properties of distributed systems without making fairness assumptions. This theory will encompass process algebra, temporal logic and semantic models, as well as treatments of real-time. The agenda also includes developing a methodology that allows successful application of this theory to the specification, analysis and verification of realistic distributed systems, including routing protocols for wireless networks.

Contemporary process algebras and temporal logics fail to make distinctions between systems of which one has a crucial liveness property and the other does not, at least when assuming *justness*, a strong progress property, but not assuming fairness. Setting up an alternative framework involves giving up on identifying strongly bisimilar systems, inventing new induction principles, developing new axiomatic bases for process algebras and new congruence formats for operational semantics, and creating new treatments of time and probability.

Even simple systems like fair schedulers or mutual exclusion protocols cannot be accurately specified in standard process algebras (or Petri nets) in the absence of fairness assumptions. Hence the work involves the study of adequate language or model extensions, and their expressive power.

1 State-of-the-art and objectives

Specification, analysis and verification of distributed systems

At an increasing rate, humanity is creating distributed systems through hardware and software—systems consisting of multiple components that interact with each other through message passing or other synchronisation mechanisms. Examples are distributed databases, communication networks, operating systems, industrial control systems, etc. Many of these systems are hard to understand, yet vitally important. Therefore, significant effort needs to be made to ensure their correct working.

Formal methods are an indispensable tool towards that end. They consist of specification formalisms to unambiguously capture the intended requirements and behaviour of a system under consideration, tools and analysis methods to study and reason about vital properties of the system, and mathematically rigorous methods to verify that (a) a system specification ensures the required properties, and (b) an implementation meets the specification.

The standard alternative to formal specification formalisms are descriptions in English, or other natural languages, that try to specify the requirements and intended workings of a system. History has shown, almost without exception, that such descriptions are riddled with ambiguities, contradictions and

*NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

POSITION PAPER March 3, 2016

under-specification. Formalisation of such a description—regardless in which formalism—is the key to elimination of these holes.

A formal specification of a distributed system typically comes in (at least) two parts.

One part formulates the *requirements* imposed on the system as a list of properties the system should have. Amongst the formalisms to specify such requirements are temporal logics like Linear-time Temporal Logic (LTL) [Pnu77] or Computation Tree Logic (CTL) [EC82]. Amongst others, they can specify *safety properties*, saying that something bad will never happen, and *liveness properties*, saying that something good will happen eventually [Lam77].

The other part is a formal description of how the system ought to work on an *operational* (= step by step) basis, but abstracting from implementation details. For distributed systems such accounts typically consist of descriptions of each of the parallel components, as well as of the communication interfaces that specify how different components interact with each other. Languages for giving such formal descriptions are *system description languages*. When a system description language features constants to specify elementary system activities, and operators (like parallel or sequential composition) to create more complex systems out of simpler ones, it is sometimes called a *process algebra*. Alternatively, operational system descriptions can be rendered in a model of concurrency, such as Petri nets or labelled transition systems. Such models are also used to describe the meaning of system description languages.

Once such a two-tiered formalisation of a system has been provided, there are two obvious tasks to ensure the correct working of implementations: (a) guaranteeing that the operational system description meets the requirements imposed on the system, and (b) ensuring that an implementation satisfies the specification. The latter task additionally requires a definition of what it means for an implementation to satisfy a specification, and this definition should ensure that any relevant correctness properties that are shown to hold for the specification also hold for the implementation.

A third type of task is the study of other properties of the implementation, not implied by the specification. Examples are measuring its execution times, when these are not part of the specification, or its success rate, for operations for which success cannot be guaranteed and only a best effort is made. Potentially, these tasks call for applications of probability theory.

Traditional approaches to ensure the correct working of distributed systems are simulation and test-bed experiments. While these are important and valid methods for system evaluation, in particular for quantitative performance evaluation, they have limitations in regards to the evaluation of basic correctness properties. Experimental evaluation is resource-intensive and time-consuming, and, even after a very long time of evaluation, only a finite set of operational scenarios can be considered—no general guarantee can be given about correct system behaviour for a wide range of unpredictable deployment scenarios. I believe that formal methods help in this regard; they complement simulation and test-bed experiments as methods for system evaluation and verification, and provide stronger and more general assurances about system properties and behaviour.

Achievements of process algebra and related formalisms

Process algebra is a family of approaches to the specification, analysis and verification of distributed systems. Its tools encompass algebraic languages for the specification of processes (mentioned above), algebraic laws to reason about processes, and induction principles to derive behaviours of infinite systems from those of their finite approximations.

Many industrial size distributed systems have been successfully specified, analysed and verified in frameworks based on process algebra. Examples can be found though the following links. Major toolsets primarily based on process algebra include FDR [GABR14], CADP [GLMS11], mCRL2 [GM14] and

the Psi-Calculi Workbench [BGRV15, BJPV11]. Most of these toolsets also use model checking or other mathematical techniques that explore the state spaces of distributed systems. Similar toolsets primarily based on the latter techniques include SPIN [Hol04], UPPAAL [BDL04], PRISM [KNP10] and TLA [Lam02].

Verification of routing protocols for wireless networks

In [FGH+12b, FGH+13, BGH16] my colleagues and I have developed a process algebra that is tailored to wireless networks—it features novel treatments of local broadcast, conditional unicast, and data handling—and used it to study AODV [PBD03]—a popular routing protocol for wireless networks, standardised by the IETF. Established methods turned out to be sufficient for proving safety properties of this protocol, in particular *loop freedom* [GHPT16]. However, standard techniques turned out to be fundamentally inadequate to establish *route discovery* or *packet delivery* [FGH+13], two fundamental liveness properties of routing protocols. Part of my research agenda is to lay the theoretical foundations to accurately formulate and prove such properties, and use them to continue my work on network protocols. I will use this case study to illustrate a few points made below.

Liveness, fairness assumptions, and their dangers

One of the crucial tasks in the analysis of distributed systems is the verification of liveness properties, saying that something good will happen eventually. A typical example is the verification of a communication protocol—such as the *alternating bit protocol* [Lyn68, BSW69]—that ensures that a stream of messages is relayed correctly, without loss or reordering, from a sender to a receiver, while using an unreliable communication channel. The protocol works by means of acknowledgements, and resending of messages for which no acknowledgement is received.

Naturally, no protocol is able to ensure such a thing, unless we assume that attempts to transmit a message over the unreliable channel will not fail in perpetuity. Such an assumption, essentially saying that if one keeps trying something forever it will eventually succeed, is often called a *fairness* assumption.

Making a fairness assumption is indispensable when verifying the alternating bit protocol. If one refuses to make such an assumption, no such protocol can be found correct, and one misses a change to check the protocol logic against possible flaws that have nothing to do with a perpetual failure of the communication channel.

For this reason, fairness assumptions are made in many process-algebraic verification methods, and are deeply ingrained in their methodology [BBK87]. This applies, amongst others, to the default incarnations of the process algebras CCS [Mil89], ACP [BW90, Fok00a, BBR10], the π-calculus [Mil99, SW01] and mCRL2 [GM14].

Using a fairness assumption, however, needs to be done with care. Making a fairness assumption can lead to patently false results. This applies for instance to the communication protocol in cases where one of the possible behaviours of the unreliable channel is to perpetually loose all messages. A *global* fairness assumption, as commonly used in process algebra, leads to false conclusions on packet delivery for routing protocols.

In defence of making a fairness assumption it is sometimes argued that whenever at some point the probability of success is 0, the success possibility should not be part of our model of reality. When infinitely many choices remain that allow success with a fixed positive probability, with probability 1 success will be achieved eventually. This argument rests on assumptions on relative probabilities of certain choices, but is applied to models that abstract from those probabilities. My counter-argument is

that (1) when abstracting from probabilities it is quite well possible that a success probability is always positive, yet quickly diminishing, so that the cumulative success probability is less than 1, and (2) that in many applications we do not know whether certain behaviours have a chance of occurring or not, but they are included in the model nevertheless.

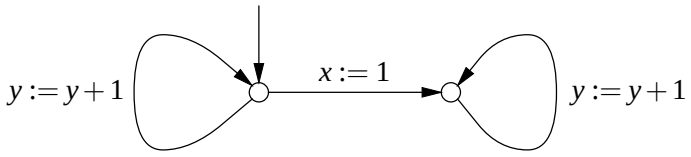
Process algebras without fairness assumptions, and their limitations

In certain process algebras, such as CSP [Hoa85, Ros97], it is not common to make any fairness assumptions. There also are variants of CCS, ACP, etc. that do not make them [Wal90, GLT09].

Here I will argue that virtually all these approaches *cannot* make sufficiently strong progress assumptions to establish meaningful liveness properties in realistic applications. Namely, I will show two programs, P and Q, that are equated in virtually all process-algebraic approaches to date. In technical terms, they are *strongly bisimilar* [Mil89, Gla11a]. Yet, there is a crucial liveness property that holds for P but not for Q, when not assuming fairness. So the process algebra must either claim that both programs have the liveness property, which in case of Q could be an unwarranted conclusion, possibly leading to the design of systems with dangerous or catastrophic behaviour, or it falls short in asserting the liveness property of P.

$$x := 1 \quad \| \quad \textbf{repeat } y := y+1 \textbf{ forever} \qquad (P)$$

Program P is the parallel composition of two non-interacting processes, one of which sets the variable x to 1, and the other repeatedly increments a variable y. I assume that both variables x and y are initialised to 0. In program Q the *case*-statement is interpreted such that if the conditions of multiple cases hold, a non-deterministic choice is made which one to execute. The programs P and Q are strongly bisimilar; both can be represented by means of the following labelled transition system:




```
repeat
  case
    if True then y:=y+1 fi
    if x = 0 then x:=1 fi
  end
forever                                  (Q)
```

As a warm-up exercise, one may ask whether the variable y in P or Q will ever reach the value 7—a liveness property. A priori, I cannot give a positive answer, for one can imagine that after incrementing y three times, the program for no apparent reason stops making progress and does not get around to any further activity. In most applications, however, it is safe to assume that this scenario will not occur. To accurately describe the intended behaviour of P or Q, or any other program, one makes a *progress assumption*, saying that if a program is in a state where further activity is possible (and this activity is not contingent on input from the environment that might fail to occur) some activity will in fact happen. This assumption is sufficient to ensure that in P or Q the variable y will at some point reach the value 7.

Progress assumptions are commonplace in process algebra and many other formalisms. They are explicitly or implicitly made in CCS, ACP, the π-calculus, CSP, etc., whenever such formalisms are employed to establish liveness properties. Temporal logics, such as LTL [Pnu77] and CTL [EC82], have progress assumptions built in, and can formalise the statement that y will in fact reach the value 7.

A more interesting question is whether x will ever reach the value 1. This liveness property is *not* guaranteed by progress assumptions as made in any of the standard process algebras or temporal logics. The problem is that all these formalisms rest on a model of concurrency where parallel composition is

modelled as arbitrary interleaving. The programs P and Q have computations like

$$\begin{array}{llllll} x:=1; & y:=y+1; & y:=y+1; & y:=y+1; & y:=y+1; & y:=y+1; \quad \ldots \\ y:=y+1; & x:=1; & y:=y+1; & y:=y+1; & y:=y+1; & y:=y+1; \quad \ldots \\ y:=y+1; & y:=y+1; & y:=y+1; & y:=y+1; & x:=1; & y:=y+1; \quad \ldots \end{array}$$

where the action $x := 1$ can be scheduled arbitrary far in the sequence of y-incrementations, but also a computation

$$y:=y+1; \quad y:=y+1; \quad y:=y+1; \quad y:=y+1; \quad y:=y+1; \quad y:=y+1; \quad \ldots \qquad (C^\infty)$$

in which $x := 1$ never happens, because $y := y+1$ is always scheduled instead. For this reason, temporal logic as well as process algebra—when not making fairness assumptions—say that x is not guaranteed to reach the value 1, regardless whether talking about P or Q.

When assuming that parallel composition is implemented by means of a scheduler that arbitrarily interleaves actions from both processes, this conclusion for P appears plausible. However, when $\|$ denotes a true parallel composition, where the program P consists of two completely independent processes, it appears more reasonable to use a stronger progress assumption that does guarantee that x will reach the value 1. Such a strong progress assumption is formalised in the context of process algebra in my paper [GH15b], written jointly with Peter Höfner, and called *justness*. We also applied it in the study of routing protocols for wireless networks [FGH+13].

However, whereas my justness assumption disqualifies the computation C^∞ for P, it is entirely reasonable to allow it for Q. After all, the **case**-statement may be implemented in such a way as to always pick the first case that applies. A fairness assumption would (unjustly) eliminate this computation even for Q; here I argue for a theory of concurrency in which such a fairness assumption is not made.

Hence, virtually all existing process-algebraic approaches equate two programs of which one has the liveness property that eventually x will reach the value 1, and the other does not, at least not without assuming fairness. So those approaches that do not assume fairness lack the power to establish this property for P.

The same limitations of temporal logics

At first sight, justness is exactly the same as what is called *justice* by Lehmann, Pnueli & Stavi [LPS81]. In fact, their motivating example for the proposal of justice is similar to program P. They called a computation *just* "if it is finite or if every transition which is continuously enabled beyond a certain point is taken infinitely many times." This proposal disqualifies the computation without the assignment $x := 1$ for the program P.

To determine whether it also disqualifies this computation for the program Q it matters how one formalises the notion of a transition being "continuously enabled". The literature following [LPS81] has systematically interpreted this as meaning "in every state" ("beyond a certain point"). The computation C^∞ of Q that only has transitions $y := y+1$ has the property that in each of its states (that is between two such transitions) the transition $x := 1$ is enabled. For this reason, that computation would be disqualified by the notion of justice attributed to [LPS81]. This formalisation of the concept of justice from [LPS81] is commonly known as *weak fairness*.

An alternative formalisation of a transition being "continuously enabled", based on the principle of "noninstantaneous readiness" of [AFK88], postulates that the enabledness of the transition $x := 1$ is interrupted when execution the conflicting transition $y := y+1$, even though it is again enabled in

the state following this transition. In [GH15b] we have formalised this interpretation of enabledness of transitions, and shown that the resulting notion of justice coincides with our own notion of justness.

Yet, since most work on temporal logic identifies the notion of justice from [LPS81] with a concept of fairness that makes no difference between the programs P and Q, it appears prudent to use a subtly different name for the concept of justness. The conclusion is that in standard approaches to temporal logic, just like in process algebra, either a concept of fairness is used that makes dangerous predictions—i.e. that in Q the variable x will eventually reach the value 1—or no such concept of fairness is used, resulting in the inability to derive important liveness properties—such as that in P the variable x will eventually reach the value 1. The notions of route discovery and packet delivery for routing protocols are among those liveness properties.

Goal

This brings me to my research agenda in this matter: the development of a theory of concurrency that is equipped to ensure liveness properties of distributed systems, incorporating justness assumptions as explained above, but without making fairness assumptions. This theory will encompass process algebra, temporal logic, Petri nets and other semantic models, as well as treatments of real-time, and of the interaction of probabilistic and nondeterministic choice.

Since this involves distinguishing programs that are strongly bisimilar, it requires a complete overhaul of all the basic machinery that has been built in the last few decennia. It requires new equivalence relations between processes, new axiomatisations, new induction principles to reason about infinite processes, new congruence formats for operational semantics ensuring compositionality of operators, and new extensions with time and probabilities.

As in the absence of fairness assumptions some crucial systems like fair schedulers or mutual exclusion protocols cannot be accurately specified in Petri nets or standard process algebras [GH15a], it also involves the study of adequate model or language extensions, and their expressive power.

My agenda furthermore aims at developing a methodology that allows successful application of the envisioned theory of concurrency to the specification, analysis and verification of realistic distributed systems, focusing on cases where the new balance in establishing liveness properties bears fruit. In particular, I advocate the application of this work to the analysis of routing protocols in wireless networks; using the new theory of concurrency I would like to extend our prior work [FGH$^+$13, FGH$^+$12a, HGT$^+$12, GHTP13, GHPT16, BGH16], amongst others by formally proving route discovery and package delivery properties of suitable protocols.

This research agenda involves the following tasks:

1. To formally define what it means for a process algebra to encompass justness but not fairness.
2. To investigate and classify semantic equivalences (necessarily incomparable with strong bisimilarity) that respect liveness when assuming justness.
3. To study liveness and justness properties in non-interleaved semantic models like Petri nets, event structures and higher dimensional automata.
4. To find complete axiomatisations and adequate induction principles for process algebras with justness.
5. To find syntactic requirements that guarantee that the relevant justness-preserving equivalences are congruences for operators specified conform those requirements.
6. To study the necessary extensions to process algebras or Petri nets to model simple processes like fair schedulers, and investigate the relative expressiveness of process algebras with and without them.

7. To re-evaluate the possibility and impossibility results for encoding synchrony in asynchrony when insisting that justness properties are preserved.
8. To extend relevant justness preserving formalisms with treatments of real-time.
9. To adapt the existing testing theory for nondeterministic probabilistic processes to a setting where justness is preserved.
10. To apply the obtained formalisms to the further study of routing protocols for wireless networks, paying special attention to liveness properties like package delivery.

2 Detailed research agenda

Although stable foundations of process algebra and related verification methods have been laid in the last 30 years, surprisingly little of it can be reused as foundation for a theory, as needed here, that encompassed justness but not fairness. Much of the existing work is predicated on the belief that at least strongly bisimilar processes need not be distinguished, and as it is necessary to give this up, a totally new way of thinking on the relevant foundations is in order.

Below I describe the main tasks that I see as part of this process.

Task 1: Process algebra and temporal logic

The first task is to formulate a precise definition of what it means for a process-algebraic specification and verification formalism to encompass justness but not fairness. Moreover, although I want to avoid a *global* fairness assumption, applying to all choices of represented systems, there needs to be a way to specify *local* fairness assumptions, ones that can be justified for particular scheduling problems in the systems under consideration, and for those only.

My proposals in this matter have already been formulated in [GH15b]—combining process algebra and temporal logic to tackle local fairness properties—but that paper is specific to the process algebra CCS and its extension with broadcast communication. In [FGH$^+$13] we have indicated how the same concepts apply to the process algebra AWN, tailored to wireless networks.

What is lacking so far is a general theory saying how to do this for arbitrary process algebras, possibly involving a CSP-style communication mechanism [Hoa85, Ros97], priorities [CLN01], or name-binding [Mil99, SW01].

Task 2: A classification of semantic equivalences and preorders

A crucial prerequisite for verifying that an implementation meets a specification is a definition of what this means. Such a definition can be given in the form of an equivalence relation or preorder on a space of models that can be used to describe both specifications and implementations. For sequential systems, an overview of suitable preorders and equivalence relations defined on labelled transition systems is given in [Gla01, Gla93]. Preorders and equivalences specifically tailored to preserve safety and liveness properties are explored in [Gla10]. Equivalences for non-sequential systems are discussed, e.g., in [GG01]. They include *interleaving equivalences*, in which parallelism is equated with arbitrary interleaving, as well as equivalence notions that take, to some degree, concurrency explicitly into account. In [Gla15] I show that none of these equivalences respect *inevitability* [MOP89]: when assuming justness but not fairness, they all equate systems of which only one has the property that all its runs reach a specific success state. Hence, none of these equivalences are suitable for a process-algebraic framework destined to establish liveness properties under the justness assumption advocated above.

As shown in [Gla10], safety and liveness properties are intimately linked with the notions of may- and must-testing of De Nicola & Hennessy [DNH84]. However, [Gla10] also treats *conditional liveness* properties that surpass the power of must-testing. In [Gla09] I proposed a notion of *reward testing*, and claimed that it exactly matches with conditional liveness properties.

My first goal in this task is to substantiate the above claim. Once that is done, the same testing framework can be applied to derive preorders for concurrent processes that respects (conditional) liveness properties in the presence of the justness assumption. This may yield a result similar to the fair failure preorder of [Vog02]. Additionally, variants of most existing preorders and equivalences may be found that take respect liveness under justness assumptions.

Possibly, just forcing an existing preorder to respect liveness by adding appropriate clauses to its definition gives a result that is less suitable for verification tasks. It will for instance most likely fail to satisfy the *recursive specification principle* of [BK86b, BBK87], saying that guarded recursive specifications have unique solutions. That problem might be addressed by formulating more discriminating preorders and equivalences that do not feature explicit conditions on infinite runs, yet respect liveness properties. A first proposal for such an equivalence is the *structure preserving bisimilarity* of [Gla15]. That equivalence is probably too discriminating for many verification tasks, so more research is called for.

Task 3: Petri nets and other semantic models

The standard semantics of process algebras is in terms of labelled transition systems. However, for accurately capturing causalities between event occurrences, models like Petri nets, event structures or higher dimensional automata are sometimes preferable. As shown above, unaugmented labelled transition systems are not sufficient to capture liveness properties when assuming justness but not fairness. On the other hand, Petri nets offer a structural characterisation of justness: if a transition is enabled, and none of the tokens enabling it are ever consumed by a competing transition, then it will eventually fire. For this reason, interpretations of process algebras in terms of these models [GM84, Win84, GV87, Win87, Old91, GV03, Gla06] need to be studied in relation to formulations of the relevant semantic equivalences in terms of these models.

Task 4: Complete axiomatisations and induction principles

Many process-algebraic verifications [Bae90] employ principles like the above-mentioned *recursive specification principle* and the *approximation induction principle* [BK86b, BBK87], allowing to derive properties of infinite systems through analysis of their finite approximations. It is likely that these principle do not hold in straightforward variants of existing semantic equivalences that respect liveness when assuming justness. The two ways to cope with that are (1) searching for finer equivalences that do not have this shortcoming, or (2) searching for alternative induction principles that hold and are useful in verification. As this time I cannot say which of these directions is the most promising; more research is in order here.

Algebraic laws have also shown their use in verification, and the isolation of a complete collection of such laws is often the starting point of both a good verification toolset and a better understanding of the semantic concepts involved. For these reasons, finding complete axiomatisations of suitable equivalences to deal with justness and liveness is an important task.

Task 5: Congruence formats for structural operational semantics

In process-algebraic verification it is essential that composition operators on processes, such as the parallel composition, are *compositional* w.r.t. the semantic equivalence employed. This means that the composition of two processes, each given as an equivalence class of, say, labelled transition systems, is independent on the choice of representatives within these equivalence classes. Compositionality of an operator w.r.t. an equivalence is the same as the equivalence being a congruence w.r.t. the operator.

Starting with [Sim85, BIM95, GV92], the most elegant and efficient way to establish compositionality results in process algebra is by means of *congruence formats*, sets of syntactic restrictions on the operational specification of the behaviour of composition operators that ensure compositionality. This line of research is continued in [Gro93, BG96, Uli92, Ver95, Gla04, FG96, Blo95, Uli00, UP02, UY00, Fok00b, BFG04, FGW06, Gla11b, GMR06, FGW12].

One of the tasks on my agenda is the search for congruence formats tailored to the equivalences produced by Task 2.

Task 6: Expressiveness

In the absence of fairness assumptions even simple systems like fair schedulers or mutual exclusion protocols cannot be accurately specified in standard process algebras or Petri nets. This is shown in [Vog02, KW97, GH15a]. For this reason, my research agenda involves the study of adequate extensions to standard process algebras, as well as to models of concurrency like Petri nets, event structures and higher dimensional automata. A plausible extension to process algebras to fill this gap are signals [Ber88]; alternatives are broadcast communication [GH15b] or priorities [GH15a]. A plausible extension to Petri are read arcs [Vog02]. The relative expressiveness of these models and languages needs to be studied, along with other arguments for one model or language over another.

The resulting study on the relative expressiveness of process algebras ought to be placed within the formal frameworks developed for comparisons of expressive power provided in [Bou85, Gor10, Gla12].

In [Sim85, Gla94] results are obtained saying that all languages with a structural operational semantics of a certain form can be expressed in versions of the process algebras MEIJE [AB84] and ACP [BK86a], respectively. Ideally, such results are to be obtained also for the language extensions contemplated above. This may involve a standardisable mechanism akin to structural operational semantics for specifying justness aspects of process combinators, in addition to their step-by-step behaviour.

Task 7: Asynchronous interaction in distributed systems

During the last few years I have been involved in joint research with Ursula Goltz (TU Braunschweig) and Uwe Nestmann (TU Berlin) aiming to determine to what extent synchronous communication can be simulated by asynchronous communication. This research was done in models like Petri nets as well as in terms of process algebras with mobility primitives, like the π-calculus. Building on earlier work [Nes00, NP00, Pal03, CCP07], we established separation results, saying that in some conditions synchronous communication cannot be mimicked by asynchronous communication, and encodability results, showing how in other situations it could be done [GGS09, GGS08, SPG11, PSN11, PN12, GGS13, PNG13, MSG14]. All those results are dependent on the choice of a semantic equivalence, to judge whether a proposed encoding of synchronous into asynchronous communication has been successful.

Hence these results need to be revisited when using semantic equivalences with the appropriate respect for justness. It would be interesting to see if this changes the relations found so far, and if so, to what extent.

Task 8: Real-time

Extensions of process algebras and semantic models with treatments of time are vital for many applications. Although many timed process algebras already exists [RR88, HR95, Wan90, MT90, MT91, BB91, NS94, BM02, OS06], it is unclear whether any of them could form a natural extension of an untimed process algebra with justness as envisioned in this project.

As an example of a process that is hard to encode in some of the proposals above, consider the parallel composition of two components that each await a communication from the environment and then synchronise with each other. The problematic part is that although, at the synchronisation point, either component can wait arbitrary long on the other, the synchronisation between the two will happen as soon as both components are ready for it, and does not admit further lingering after that point.

The goal of this task is to find a suitable process algebra that provides a timed extension for a suitable untimed proposal with justness, and to do a formal comparison of the expressiveness of the existing process algebras for timed systems. The latter can be done by mapping them all into a formalism that includes the expressive power of each of the process algebras to be compared.

In the setting of wireless network protocols, a step towards the first goal is [BGH16].

Task 9: Extensions with probabilistic choice

In collaboration with Yuxin Deng, Matthew Hennessy, Carroll Morgan and Chenyi Zhang [DGH$^+$07, DGMZ07, DGHM08, DGHM09, DGHM14], building on earlier work by Wang Yi, Kim Larsen, Bengt Jonsson, Chris Ho-Stuart and Roberto Segala [WL92, JHW94, Seg96], I have developed a theory of testing for processes with nondeterminism and probabilistic choice. This theory determines the coarsest semantic equivalences for probabilistic processes that respect safety and liveness properties. A natural task is to extend this work to a parallel processes with justness, thereby extending the non-probabilistic work in Task 2.

Task 10: Network protocols

The final task on my agenda is the application of the obtained framework to the specification, analysis and verification of routing protocols. This involves:

- enriching our process algebra of wireless networks (AWN [FGH$^+$12b]) with time, and use it to specify a timed version of the AODV routing protocol (based on the existing untimed formalisation of AODV [FGH$^+$13, GHPT16]); this is work to appear as [BGH16];
- formally specifying a number of similar protocols that are used in industry and standardisation bodies, including AODVv2 [PRD$^+$16], HWMP [IEE11], OLSR [CJ03], OLSRv2 [CDJH14] and BATMAN [IET08]—the latter three heavily depend on time and need a timed version of AWN;
- establishing safety properties like loop freedom for some of these protocols, or show why such properties do not hold;
- establishing liveness properties like route discovery or packet delivery for some of these protocols, or show why such properties do not hold;
- proposing modifications of these protocols that have the required safety and liveness properties, or design a new protocol based on similar ideas if the latter is hopeless;
- extending the above work to the link layer of the protocol stack, by formally specifying the CMSA protocol [IEE03] and studying its relevant properties.

Besides this, other applications of the obtained framework to the specification, analysis and verification of realistic distributed systems are worthy of study as well.

3 Conclusion

I have presented a research agenda aiming at laying the foundations of a theory of concurrency that is equipped to ensure liveness properties of distributed systems without making fairness assumptions. The agenda also includes the application of this theory to the specification, analysis and verification of realistic distributed systems. I have divided the detailed agenda into 10 tasks, each of which in some sense involves solving one or more open problems.

It is not my intension to address all tasks listed here myself—there are simply too many tasks for that to work out. I hope that this document stimulates others to do some work in this area.

References

[AB84] D. Austry & G. Boudol (1984): *Algèbre de processus et synchronisations*. *Theoretical Computer Science* 30(1), pp. 91–131, doi:10.1016/0304-3975(84)90067-7.

[AFK88] K.R. Apt, N. Francez & S. Katz (1988): *Appraising Fairness in Languages for Distributed Programming*. *Distributed Computing* 2(4), pp. 226–241, doi:10.1007/BF01872848.

[Bae90] J.C.M. Baeten, editor (1990): *Applications of Process Algebra*. Cambridge Tracts in Theoretical Computer Science 17, Cambridge University Press.

[BB91] J.C.M. Baeten & J.A. Bergstra (1991): *Real Time Process Algebra*. *Formal Asp. Comput.* 3(2), pp. 142–188, doi:10.1007/BF01898401.

[BBK87] J.C.M. Baeten, J.A. Bergstra & J.W. Klop (1987): *On the Consistency of Koomen's Fair Abstraction Rule*. *Theoretical Computer Science* 51(1/2), pp. 129–176, doi:10.1016/0304-3975(87)90052-1.

[BBR10] J.C.M. Baeten, T. Basten & M.A. Reniers (2010): *Process Algebra: Equational Theories of Communicating Processes*. Cambridge University Press.

[BDL04] G. Behrmann, A. David & K.G. Larsen (2004): *A Tutorial on Uppaal*. In M. Bernardo & F. Corradini, editors: Revised Lectures on *Formal Methods for the Design of Real-Time Systems*, LNCS 3185, Springer, pp. 200–236, doi:10.1007/978-3-540-30080-9_7.

[Ber88] J.A. Bergstra (1988): *ACP with signals*. In J. Grabowski, P. Lescanne & W. Wechler, editors: *Algebraic and Logic Programming*, LNCS 343, Springer, pp. 11–20, doi:10.1007/3-540-50667-5_53.

[BFG04] B. Bloom, W.J. Fokkink & R.J. van Glabbeek (2004): *Precongruence Formats for Decorated Trace Semantics*. *Transactions on Computational Logic* 5(1), pp. 26–78, doi:10.1145/963927.963929.

[BG96] R.N. Bol & J.F. Groote (1996): *The meaning of negative premises in transition system specifications*. *Journal of the ACM* 43(5), pp. 863–914, doi:10.1145/234752.234756.

[BGH16] E. Bres, R.J. van Glabbeek & P. Höfner (2016): *A Timed Process Algebra for Wireless Networks with an Application in Routing*. Technical Report 9145, NICTA. Available at http://nicta.com.au/pub?id=9145. Extended abstract to appear in P. Thiemann, editor: *Programming Languages and Systems*: Proceedings 25th *European Symposium on Programming* (ESOP'16), Eindhoven, The Netherlands, LNCS 9632, Springer, doi:10.1007/978-3-662-49498-1_5.

[BGRV15] J. Borgström, R. Gutkovas, I. Rodhe & B. Victor (2015): *The Psi-Calculi Workbench: A Generic Tool for Applied Process Calculi*. *ACM Trans. Embedded Comput. Syst.* 14(1), pp. 9:1–9:25, doi:10.1145/2682570.

[BIM95] B. Bloom, S. Istrail & A.R. Meyer (1995): *Bisimulation Can't be Traced*. *Journal of the ACM* 42(1), pp. 232–268, doi:10.1145/200836.200876.

[BJPV11] J. Bengtson, M. Johansson, J. Parrow & B. Victor (2011): *Psi-calculi: a framework for mobile processes with nominal data and logic*. *Logical Methods in Computer Science* 7(1), doi:10.2168/LMCS-7(1:11)2011.

[BK86a] J.A. Bergstra & J.W. Klop (1986): *Algebra of communicating processes*. In J.W.d. Bakker, M. Hazewinkel & J.K. Lenstra, editors: *Mathematics and Computer Science*, CWI Monograph 1, North-Holland, Amsterdam, pp. 89–138.

[BK86b] J.A. Bergstra & J.W. Klop (1986): *Verification of an alternating bit protocol by means of process algebra*. In W. Bibel & K.P. Jantke, editors: *Mathematical Methods of Specification and Synthesis of Software Systems '85*, LNCS 215, Springer, pp. 9–23, doi:10.1007/3-540-16444-8_1.

[Blo95] B. Bloom (1995): *Structural operational semantics for weak bisimulations*. *Theoretical Computer Science* 146, pp. 25–68, doi:10.1016/0304-3975(94)00152-9.

[BM02] J.C.M. Baeten & C.A. Middelburg (2002): *Process Algebra with Timing*. Springer, doi:10.1007/978-3-662-04995-2.

[Bou85] G. Boudol (1985): *Notes on algebraic calculi of processes*. In K. Apt, editor: *Logics and Models of Concurrent Systems*, Springer, pp. 261–303, doi:10.1007/978-3-642-82453-1_9. NATO ASI Series F13.

[BSW69] K.A. Bartlett, R.A. Scantlebury & P.T. Wilkinson (1969): *A note on reliable full-duplex transmission over half-duplex links*. *CACM* 12, pp. 260–261, doi:10.1145/362946.362970.

[BW90] J.C.M. Baeten & W.P. Weijland (1990): *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, doi:10.1017/CBO9780511624193.

[CCP07] D. Cacciagrano, F. Corradini & C. Palamidessi (2007): *Separation of synchronous and asynchronous communication via testing*. *Theoretical Computer Science* 386(3), pp. 218–235, doi:10.1016/j.tcs.2007.07.009.

[CDJH14] T. Clausen, C. Dearlove, P. Jacquet & U. Herberg (2014): *The Optimized Link State Routing Protocol Version 2*. RFC 7181, Internet Engineering Task Force (IETF). Available at http://www.ietf.org/rfc/rfc7181.txt.

[CJ03] T. Clausen & P. Jacquet (2003): *Optimized Link State Routing Protocol (OLSR)*. RFC 3626 (Experimental), Network Working Group. Available at http://www.ietf.org/rfc/rfc3626.txt.

[CLN01] R. Cleaveland, G. Lüttgen & V. Natarajan (2001): *Priority in Process Algebra*. In J.A. Bergstra, A. Ponse & S.A. Smolka, editors: *Handbook of Process Algebra*, chapter 12, Elsevier, pp. 711–765, doi:10.1016/B978-044482830-9/50030-8.

[DGH+07] Y. Deng, R.J. van Glabbeek, M. Hennessy, C.C. Morgan & C. Zhang (2007): *Remarks on Testing Probabilistic Processes*. In L. Cardelli, M. Fiore & G. Winskel, editors: *Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin*, *Electronic Notes in Theoretical Computer Science* 172, Elsevier, pp. 359–397, doi:10.1016/j.entcs.2007.02.013.

[DGHM08] Y. Deng, R.J. van Glabbeek, M. Hennessy & C.C. Morgan (2008): *Characterising Testing Preorders for Finite Probabilistic Processes*. *Logical Methods in Computer Science* 4(4):4, doi:10.2168/LMCS-4(4:4)2008.

[DGHM09] Y. Deng, R.J. van Glabbeek, M. Hennessy & C.C. Morgan (2009): *Testing Finitary Probabilistic Processes (extended abstract)*. In M. Bravetti & G. Zavattaro, editors: Proceedings 20th International Conference on *Concurrency Theory* (CONCUR'09), Bologna, Italy, September 1-4, 2009, LNCS 5710, Springer, pp. 274–288, doi:10.1007/978-3-642-04081-8_19.

[DGHM14] Y. Deng, R.J. van Glabbeek, M. Hennessy & C.C. Morgan (2014): *Real-Reward Testing for Probabilistic Processes*. *Theoretical Computer Science* 538, pp. 16–36, doi:10.1016/j.tcs.2013.07.016.

[DGMZ07] Y. Deng, R.J. van Glabbeek, C.C. Morgan & C. Zhang (2007): *Scalar Outcomes Suffice for Finitary Probabilistic Testing*. In R. De Nicola, editor: Proceedings 16th *European Symposium on Program-*

ming (ESOP'07), Braga, Portugal, 24 March - 1 April, 2007, LNCS 4421, Springer, pp. 363–378, doi:10.1007/978-3-540-71316-6_25.

[DNH84] R. De Nicola & M. Hennessy (1984): *Testing equivalences for processes*. *Theoretical Computer Science* 34, pp. 83–133, doi:10.1016/0304-3975(84)90113-0.

[EC82] E.A. Emerson & E.M. Clarke (1982): *Using Branching Time Temporal Logic to Synthesize Synchronization Skeletons*. *Science of Computer Programming* 2(3), pp. 241–266, doi:10.1016/0167-6423(83)90017-5.

[FG96] W.J. Fokkink & R.J. van Glabbeek (1996): *Ntyft/ntyxt rules reduce to ntree rules*. *Information and Computation* 126(1), pp. 1–10, doi:10.1006/inco.1996.0030.

[FGH+12a] A. Fehnker, R.J. van Glabbeek, P. Höfner, A.K. McIver, M. Portmann & W.L. Tan (2012): *Automated Analysis of AODV Using UPPAAL*. In C. Flanagan & B. König, editors: Proceedings 18th International Conference on *Tools and Algorithms for the Construction and Analysis of Systems* (TACAS'12), Tallinn, Estonia, March/April 2012, LNCS 7214, Springer, pp. 173–187, doi:10.1007/978-3-642-28756-5_13.

[FGH+12b] A. Fehnker, R.J. van Glabbeek, P. Höfner, A.K. McIver, M. Portmann & W. Tan (2012): *A Process Algebra for Wireless Mesh Networks*. In H. Seidl, editor: *Programming Languages and Systems:* Proceedings 21st *European Symposium on Programming* (ESOP'12), Tallinn, Estonia, March/April 2012, LNCS 7211, Springer, pp. 295–315, doi:10.1007/978-3-642-28869-2_15.

[FGH+13] A. Fehnker, R.J. van Glabbeek, P. Höfner, A.K. McIver, M. Portmann & W.L. Tan (2013): *A Process Algebra for Wireless Mesh Networks used for Modelling, Verifying and Analysing AODV*. Technical Report 5513, NICTA. Available at http://arxiv.org/abs/1312.7645.

[FGW06] W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2006): *Compositionality of Hennessy-Milner Logic by Structural Operational Semantics*. *Theoretical Computer Science* 354(3), pp. 421–440, doi:10.1016/j.tcs.2005.11.035.

[FGW12] W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2012): *Divide and congruence: From decomposition of modal formulas to preservation of branching and η-bisimilarity*. *Information and Computation* 214, pp. 59–85, doi:10.1016/j.ic.2011.10.011.

[Fok00a] W.J. Fokkink (2000): *Introduction to Process Algebra*. Texts in Theoretical Computer Science, An EATCS Series, Springer, doi:10.1007/978-3-662-04293-9.

[Fok00b] W.J. Fokkink (2000): *Rooted branching bisimulation as a congruence*. *Journal of Computer and System Sciences* 60(1), pp. 11–27, doi:10.1006/jcss.1999.1663.

[GABR14] T. Gibson-Robinson, P.J. Armstrong, A. Boulgakov & A.W. Roscoe (2014): *FDR3 - A Modern Refinement Checker for CSP*. In E. Ábrahám & K. Havelund, editors: Proceedings *Tools and Algorithms for the Construction and Analysis of Systems* - 20th International Conference (TACAS'14), Grenoble, France, April 2014, LNCS 8413, Springer, pp. 187–201, doi:10.1007/978-3-642-54862-8_13.

[GG01] R.J. van Glabbeek & U. Goltz (2001): *Refinement of Actions and Equivalence Notions for Concurrent Systems*. *Acta Informatica* 37, pp. 229–327, doi:10.1007/s002360000041.

[GGS08] R.J. van Glabbeek, U. Goltz & J.-W. Schicke (2008): *On Synchronous and Asynchronous Interaction in Distributed Systems*. In E. Ochmański & J. Tyszkiewicz, editors: Proceedings 33rd International Symposium on *Mathematical Foundations of Computer Science* (MFCS'08), Toruń, Poland, August 2008, LNCS 5162, Springer, pp. 16–35, doi:10.1007/978-3-540-85238-4_2.

[GGS09] R.J. van Glabbeek, U. Goltz & J.-W. Schicke (2009): *Symmetric and Asymmetric Asynchronous Interaction*. In F. Bonchi, D. Grohmann, P. Spoletini, A. Troina & E. Tuosto, editors: Proceedings of the *First Interaction and Concurrency Experience* (ICE'08), *Electronic Notes in Theoretical Computer Science* 229(3), Elsevier, pp. 77–95, doi:10.1016/j.entcs.2009.06.040.

[GGS13] R.J. van Glabbeek, U. Goltz & J.-W. Schicke-Uffmann (2013): *On Characterising Distributability*. *Logical Methods in Computer Science* 9(3):17, doi:10.2168/LMCS-9(3:17)2013.

[GH15a] R.J. van Glabbeek & P. Höfner (2015): *CCS: It's not fair! - Fair schedulers cannot be implemented in CCS-like languages even under progress and certain fairness assumptions*. *Acta Informatica* 52(2-3), pp. 175–205, doi:10.1007/s00236-015-0221-6.

[GH15b] R.J. van Glabbeek & P. Höfner (2015): *Progress, Fairness and Justness in Process Algebra*. Technical Report 8501, NICTA, Sydney, Australia. Available at http://arxiv.org/abs/1501.03268.

[GHPT16] R.J. van Glabbeek, P. Höfner, M. Portmann & W.L. Tan (2016): *Modelling and Verifying the AODV Routing Protocol*. To appear in *Distributed Computing*, doi:10.1007/s00446-015-0262-7. Available at http://arxiv.org/abs/1512.08867.

[GHTP13] R.J. van Glabbeek, P. Höfner, W.L. Tan & M. Portmann (2013): *Sequence Numbers Do Not Guarantee Loop Freedom —AODV Can Yield Routing Loops—*. In: Proceedings 16th ACM International Conference on *Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWiM'13), Barcelona, Spain, November 2013, ACM, pp. 91–100, doi:10.1145/2507924.2507943.

[Gla93] R.J. van Glabbeek (1993): *The Linear Time – Branching Time Spectrum II; The semantics of sequential systems with silent moves (extended abstract)*. In E. Best, editor: Proceedings *CONCUR'93,* 4^{th} International Conference on *Concurrency Theory,* Hildesheim, Germany, August 1993, LNCS 715, Springer, pp. 66–81, doi:10.1007/3-540-57208-2_6.

[Gla94] R.J. van Glabbeek (1994): *On the expressiveness of ACP (extended abstract)*. In A. Ponse, C. Verhoef & S.F.M. van Vlijmen, editors: Proceedings First Workshop on the *Algebra of Communicating Processes,* ACP94, Utrecht, The Netherlands, May 1994, Workshops in Computing, Springer, pp. 188–217.

[Gla01] R.J. van Glabbeek (2001): *The Linear Time – Branching Time Spectrum I; The Semantics of Concrete, Sequential Processes*. In J.A. Bergstra, A. Ponse & S.A. Smolka, editors: *Handbook of Process Algebra*, chapter 1, Elsevier, pp. 3–99, doi:10.1016/B978-044482830-9/50019-9.

[Gla04] R.J. van Glabbeek (2004): *The Meaning of Negative Premises in Transition System Specifications II*. *Journal of Logic and Algebraic Programming* 60–61, pp. 229–258, doi:10.1016/j.jlap.2004.03.007.

[Gla06] R.J. van Glabbeek (2006): *On the Expressiveness of Higher Dimensional Automata*. *Theoretical Computer Science* 368(1-2), pp. 169–194, doi:10.1016/j.tcs.2006.06.024.

[Gla09] R.J. van Glabbeek (2009): *The Linear Time Branching Time Spectrum after 20 years* or *Full abstraction for safety and liveness properties*. Copies of slides. Invited talk for IFIP WG 1.8 at CONCUR'09 in Bologna. Available at http://theory.stanford.edu/~rvg/abstracts.html#20years.

[Gla10] R.J. van Glabbeek (2010): *The Coarsest Precongruences Respecting Safety and Liveness Properties*. In C.S. Calude & V. Sassone, editors: Proceedings 6th IFIP TC 1/WG 2.2 International Conference on *Theoretical Computer Science* (TCS 2010), Held as Part of the *World Computer Congress* 2010, Brisbane, Australia, September 20-23, 2010, *IFIP* 323, Springer, pp. 32–52, doi:10.1007/978-3-642-15240-5_3. Available at http://arxiv.org/abs/1007.5491.

[Gla11a] R.J. van Glabbeek (2011): *Bisimulation*. In D. Padua, editor: *Encyclopedia of Parallel Computing*, Springer, pp. 136–139, doi:10.1007/978-0-387-09766-4_149.

[Gla11b] R.J. van Glabbeek (2011): *On Cool Congruence Formats for Weak Bisimulations*. *Theoretical Computer Science* 412(28), pp. 3283–3302, doi:10.1016/j.tcs.2011.02.036.

[Gla12] R.J. van Glabbeek (2012): *Musings on Encodings and Expressiveness*. In B. Luttik & M.A. Reniers, editors: Proceedings Combined 19th International Workshop on *Expressiveness in Concurrency* and 9th Workshop on *Structured Operational Semantics,* Newcastle upon Tyne, UK, September 3, 2012, *Electronic Proceedings in Theoretical Computer Science* 89, Open Publishing Association, pp. 81–98, doi:10.4204/EPTCS.89.7.

[Gla15] R.J. van Glabbeek (2015): *Structure Preserving Bisimilarity, Supporting an Operational Petri Net Semantics of CCSP*. In R. Meyer, A. Platzer & H. Wehrheim, editors: Proceedings *Correct System Design - Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birth-*

day, Oldenburg, Germany, September 8-9, 2015, LNCS 9360, Springer, pp. 99–130, doi:10.1007/978-3-319-23506-6_9. Available at http://arxiv.org/abs/1509.05842.

[GLMS11] H. Garavel, F. Lang, R. Mateescu & W. Serwe (2011): *CADP 2010: A Toolbox for the Construction and Analysis of Distributed Processes*. In P.A. Abdulla & K.R.M. Leino, editors: Proceedings *Tools and Algorithms for the Construction and Analysis of Systems* - 17th International Conference (TACAS 2011), Saarbrücken, Germany, March 26-April 3, 2011, LNCS 6605, Springer, pp. 372–387, doi:10.1007/978-3-642-19835-9_33.

[GLT09] R.J. van Glabbeek, B. Luttik & N. Trčka (2009): *Branching Bisimilarity with Explicit Divergence*. *Fundamenta Informaticae* 93(4), pp. 371–392.

[GM84] U. Goltz & A. Mycroft (1984): *On the relationship of CCS and Petri nets*. In J. Paredaens, editor: *Proceedings 11th ICALP,* Antwerpen, LNCS 172, Springer, pp. 196–208, doi:10.1007/3-540-13345-3_18.

[GM14] J.F. Groote & M.R. Mousavi (2014): *Modeling and Analysis of Communicating Systems*. MIT Press.

[GMR06] J.F. Groote, M.R. Mousavi & M.A. Reniers (2006): *A Hierarchy of SOS Rule Formats*. *Electronic Notes in Theoretical Computer Science* 156(1), pp. 3–25, doi:10.1016/j.entcs.2005.11.077.

[Gor10] D. Gorla (2010): *Towards a unified approach to encodability and separation results for process calculi*. *Information and Computation* 208(9), pp. 1031–1053, doi:10.1016/j.ic.2010.05.002.

[Gro93] J.F. Groote (1993): *Transition System Specifications with Negative Premises*. *Theoretical Computer Science* 118(2), pp. 263–299, doi:10.1016/0304-3975(93)90111-6.

[GV87] R.J. van Glabbeek & F.W. Vaandrager (1987): *Petri net models for algebraic theories of concurrency (extended abstract)*. In J.W. de Bakker, A.J. Nijman & P.C. Treleaven, editors: Proceedings *PARLE, Parallel Architectures and Languages Europe,* Eindhoven, The Netherlands, June 1987, Vol. II: Parallel Languages, LNCS 259, Springer, pp. 224–242, doi:10.1007/3-540-17945-3_13.

[GV92] J.F. Groote & F.W. Vaandrager (1992): *Structured Operational Semantics and Bisimulation as a Congruence*. *Information and Computation* 100(2), pp. 202–260, doi:10.1016/0890-5401(92)90013-6.

[GV03] R.J. van Glabbeek & F.W. Vaandrager (2003): *Bundle Event Structures and CCSP*. In R. Amadio & D. Lugiez, editors: Proceedings *CONCUR'03,* 14th International Conference on *Concurrency Theory,* Marseille, France, September 2003, LNCS 2761, Springer, pp. 57–71, doi:10.1007/978-3-540-45187-7_4.

[HGT+12] P. Höfner, R.J. van Glabbeek, W.L. Tan, M. Portmann, A.K. McIver & A. Fehnker (2012): *A Rigorous Analysis of AODV and its Variants*. In A.Y. Zomaya, B. Landfeldt & R. Prakash, editors: Proceedings 15th ACM International Conference on *Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWiM'12), Paphos, Cyprus, October 2012, ACM, pp. 203–212, doi:10.1145/2387238.2387274.

[Hoa85] C.A.R. Hoare (1985): *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs.

[Hol04] G.J. Holzmann (2004): *The SPIN Model Checker - primer and reference manual*. Addison-Wesley.

[HR95] M. Hennessy & T. Regan (1995): *A Process Algebra for Timed Systems*. *Information and Computation* 117(2), pp. 221–239, doi:10.1006/inco.1995.1041.

[IEE03] IEEE (2003): *IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*, pp. i–513, doi:10.1109/IEEESTD.2003.95617.

[IEE11] IEEE (2011): *IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 10: Mesh Networking*. Available at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6018236.

[IET08] IETF (2008): *Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.) draft-openmesh-b-a-t-m-a-n-00*. Available at `https://tools.ietf.org/html/draft-openmesh-b-a-t-m-a-n-00`.

[JHW94] B. Jonsson, C. Ho-Stuart & Wang Yi (1994): *Testing and Refinement for Nondeterministic and Probabilistic Processes*. In: Proceedings of the 3rd International Symposium on *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, Springer, pp. 418–430, doi:10.1007/3-540-58468-4_176.

[KNP10] M. Kwiatkowska, G. Norman & D. Parker (2010): *Advances and Challenges of Probabilistic Model Checking*. In: *Proc. 48th Annual Allerton Conference on Communication, Control and Computing*, IEEE Press, pp. 1691–1698, doi:10.1109/ALLERTON.2010.5707120.

[KW97] E. Kindler & R. Walter (1997): *Mutex Needs Fairness*. *Information Processing Letters* 62(1), pp. 31–39, doi:10.1016/S0020-0190(97)00033-1.

[Lam77] L. Lamport (1977): *Proving the correctness of multiprocess programs*. *IEEE Transactions on Software Engineering* 3(2), pp. 125–143, doi:10.1109/TSE.1977.229904.

[Lam02] L. Lamport (2002): *Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley.

[LPS81] D.J. Lehmann, A. Pnueli & J. Stavi (1981): *Impartiality, Justice and Fairness: The Ethics of Concurrent Termination*. In S. Even & O. Kariv, editors: *Automata, Languages and Programming (ICALP)*, LNCS 115, Springer, pp. 264–277, doi:10.1007/3-540-10843-2_22.

[Lyn68] W.C. Lynch (1968): *Reliable full-duplex file transmission over half-duplex telephone line*. *CACM* 11(6), pp. 407–410, doi:10.1145/363347.363366.

[Mil89] R. Milner (1989): *Communication and concurrency*. PHI Series in computer science, Prentice Hall.

[Mil99] R. Milner (1999): *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press.

[MOP89] A.W. Mazurkiewicz, E. Ochmanski & W. Penczek (1989): *Concurrent Systems and Inevitability*. *Theoretical Computer Science* 64(3), pp. 281–304, doi:10.1016/0304-3975(89)90052-2.

[MSG14] S. Mennicke, J.-W. Schicke-Uffmann & U. Goltz (2014): *On the Step Branching Time Closure of Free-Choice Petri Nets*. In E. Ábrahám & C. Palamidessi, editors: Proceedings 34th IFIP WG 6.1 International Conference on *Formal Techniques for Distributed Objects, Components, and Systems* (FORTE'14), Berlin, Germany, June 3-5, 2014, LNCS 8461, Springer, pp. 232–248, doi:10.1007/978-3-662-43613-4_15.

[MT90] F. Moller & C.M.N. Tofts (1990): *A Temporal Calculus of Communicating Systems*. In J.C.M. Baeten & J.W. Klop, editors: *Theories of Concurrency: Unification and Extension* (CONCUR'90), LNCS 458, Springer, pp. 401–415, doi:10.1007/BFb0039073.

[MT91] F. Moller & C.M.N. Tofts (1991): *Relating Processes With Respect to Speed*. In J.C.M. Baeten & J.F. Groote, editors: Proceedings 2nd International Conference on *Concurrency Theory* (CONCUR'91), Amsterdam, The Netherlands, August 1991, LNCS 527, Springer, pp. 424–438, doi:10.1007/3-540-54430-5_104.

[Nes00] U. Nestmann (2000): *What is a "Good" Encoding of Guarded Choice?* *Information and Computation* 156(1-2), pp. 287–319, doi:10.1006/inco.1999.2822.

[NP00] U. Nestmann & B. Pierce (2000): *Decoding Choice Encodings*. *Information and Computation* 163(1), pp. 1–59, doi:10.1006/inco.2000.2868.

[NS94] X. Nicollin & J. Sifakis (1994): *The Algebra of Timed Processes, ATP: Theory and Application*. *Information and Computation* 114(1), pp. 131–178, doi:10.1006/inco.1994.1083.

[Old91] E.-R. Olderog (1991): *Nets, Terms and Formulas: Three Views of Concurrent Processes and their Relationship*. *Cambridge Tracts in Theoretical Computer Science* 23, Cambridge University Press, doi:10.1017/CBO9780511526589.

[OS06] J. Ouaknine & S. Schneider (2006): *Timed CSP: A Retrospective*. *Electronic Notes in Theoretical Computer Science* 162, pp. 273–276, doi:10.1016/j.entcs.2005.12.093.

[Pal03] C. Palamidessi (2003): *Comparing The Expressive Power Of The Synchronous And Asynchronous Pi-Calculi*. *Mathematical Structures in Computer Science* 13(5), pp. 685–719, doi:10.1017/S0960129503004043.

[PBD03] C.E. Perkins, E.M. Belding-Royer & S. Das (2003): *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561, Network Working Group. Available at http://www.ietf.org/rfc/rfc3561.txt.

[PN12] K. Peters & U. Nestmann (2012): *Is It a "Good" Encoding of Mixed Choice?* In L. Birkedal, editor: Proceedings 15th International Conference on *Foundations of Software Science and Computational Structures* (FOSSACS'12), Tallinn, Estonia, March 24 - April 1, 2012, LNCS 7213, Springer, pp. 210–224, doi:10.1007/978-3-642-28729-9_14.

[PNG13] K. Peters, U. Nestmann & U. Goltz (2013): *On Distributability in Process Calculi*. In M. Felleisen & P. Gardner, editors: *Programming Languages and Systems:* Proceedings 22nd *European Symposium on Programming* (ESOP'13), Rome, Italy, March 2013, LNCS 7792, Springer, pp. 310–329, doi:10.1007/978-3-642-37036-6_18.

[Pnu77] A. Pnueli (1977): *The Temporal Logic of Programs*. In: *Foundations of Computer Science (FOCS'77)*, IEEE, pp. 46–57, doi:10.1109/SFCS.1977.32.

[PRD+16] C.E. Perkins, S. Ratliff, J. Dowdell, L. Steenbrink & V. Mercieca (2016): *Ad Hoc On-demand Distance Vector Version 2 (AODVv2) Routing*. Internet Draft (Standards Track), Mobile Ad hoc Networks Working Group. Available at http://tools.ietf.org/html/draft-ietf-manet-aodvv2-13.

[PSN11] K. Peters, J.-W. Schicke & U. Nestmann (2011): *Synchrony vs Causality in the Asynchronous Pi-Calculus*. In B. Luttik & F. Valencia, editors: Proceedings 18th International Workshop on *Expressiveness in Concurrency,* Aachen, Germany, September 2011, *Electronic Proceedings in Theoretical Computer Science* 64, Open Publishing Association, pp. 89–103, doi:10.4204/EPTCS.64.7.

[Ros97] A.W. Roscoe (1997): *The Theory and Practice of Concurrency*. Prentice-Hall. Available at http://www.comlab.ox.ac.uk/bill.roscoe/publications/68b.pdf.

[RR88] G.M. Reed & A.W. Roscoe (1988): *A Timed Model for Communicating Sequential Processes*. *Theoretical Computer Science* 58, pp. 249–261, doi:10.1016/0304-3975(88)90030-8.

[Seg96] R. Segala (1996): *Testing Probabilistic Automata*. In: Proceedings of the 7th International Conference on *Concurrency Theory* (CONCUR'96), Pisa, Italy, August 1996, LNCS 1119, Springer, pp. 299–314, doi:10.1007/3-540-61604-7_62.

[Sim85] R. de Simone (1985): *Higher-level synchronising devices in* MEIJE*-SCCS*. *Theoretical Computer Science* 37, pp. 245–267, doi:10.1016/0304-3975(85)90093-3.

[SPG11] J.-W. Schicke, K. Peters & U. Goltz (2011): *Synchrony vs. Causality in Asynchronous Petri Nets*. In B. Luttik & F. Valencia, editors: Proceedings 18th International Workshop on *Expressiveness in Concurrency,* Aachen, Germany, September 2011, *Electronic Proceedings in Theoretical Computer Science* 64, Open Publishing Association, pp. 119–131, doi:10.4204/EPTCS.64.9.

[SW01] D. Sangiorgi & D. Walker (2001): *The Pi-Calculus - a theory of mobile processes*. Cambridge University Press.

[Uli92] I. Ulidowski (1992): *Equivalences on Observable Processes*. In: Proceedings of the Seventh Annual Symposium on *Logic in Computer Science (LICS'92), Santa Cruz, California, USA, June 1992*, IEEE Computer Society, pp. 148–159, doi:10.1109/LICS.1992.185529.

[Uli00] I. Ulidowski (2000): *Finite axiom systems for testing preorder and De Simone process languages*. *Theoretical Computer Science* 239(1), pp. 97–139, doi:10.1016/S0304-3975(99)00214-5.

[UP02] I. Ulidowski & I.C.C. Phillips (2002): *Ordered SOS Process Languages for Branching and Eager Bisimulations*. *Information and Computation* 178(1), pp. 180–213, doi:10.1006/inco.2002.3161.

[UY00] I. Ulidowski & S. Yuen (2000): *Process Languages for Rooted Eager Bisimulation*. In C. Palamidessi, editor: Proceedings 11th International Conference on *Concurrency Theory* (CON-

CUR'00), University Park, PA, USA, August 2000, LNCS 1877, Springer, pp. 275–289, doi:10.1007/3-540-44618-4_21.

[Ver95] C. Verhoef (1995): *A Congruence Theorem for Structured Operational Semantics with Predicates and Negative Premises*. *Nord. J. Comput.* 2(2), pp. 274–302.

[Vog02] W. Vogler (2002): *Efficiency of asynchronous systems, read arcs, and the MUTEX-problem*. *Theoretical Computer Science* 275(1-2), pp. 589–631, doi:10.1016/S0304-3975(01)00300-0.

[Wal90] D.J. Walker (1990): *Bisimulation and divergence*. *Information and Computation* 85(2), pp. 202–241, doi:10.1016/0890-5401(90)90048-M.

[Wan90] Wang Yi (1990): *Real-Time Behaviour of Asynchronous Agents*. In J.C.M. Baeten & J.W. Klop, editors: Proceedings *CONCUR'90, Theories of Concurrency: Unification and Extension,* Amsterdam, August 1990, LNCS 458, Springer, pp. 502–520, doi:10.1007/BFb0039080.

[Win84] G. Winskel (1984): *A new definition of morphism on Petri nets*. In M. Fontet & K. Mehlhorn, editors: *Proceedings STACS 84*, LNCS 166, Springer, pp. 140–150, doi:10.1007/3-540-12920-0_13.

[Win87] G. Winskel (1987): *Event structures*. In W. Brauer, W. Reisig & G. Rozenberg, editors: *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course,* Good Honnef, September 1986, LNCS 255, Springer, pp. 325–392, doi:10.1007/3-540-17906-2_31.

[WL92] Wang Yi & K.G. Larsen (1992): *Testing Probabilistic and Nondeterministic Processes*. In: Proceedings of the IFIP TC6/WG6.1 Twelfth International Symposium on *Protocol Specification, Testing and Verification, IFIP Transactions* C-8, North-Holland, pp. 47–61.