Quantitative Aspects of Programming Languages and Systems over the past 2^4 years and beyond

Alessandro Aldini
University of Urbino
Urbino, Italy
`alessandro.aldini@uniurb.it`

Quantitative aspects of computation are related to the use of both physical and mathematical quantities, including time, performance metrics, probability, and measures for reliability and security. They are essential in characterizing the behaviour of many critical systems and in estimating their properties. Hence, they need to be integrated both at the level of system modeling and within the verification methodologies and tools. Along the last two decades a variety of theoretical achievements and automated techniques have contributed to make quantitative modeling and verification mainstream in the research community. In the same period, they represented the central theme of the series of workshops entitled Quantitative Aspects of Programming Languages and Systems (QAPL) and born in 2001. The aim of this survey is to revisit such achievements and results from the standpoint of QAPL and its community.

1 Introduction

Quantitative aspects of computation refer to the use of physical quantities, like, e.g., time and bandwidth, as well as mathematical quantities, like, e.g., probabilities, for the characterisation of the behaviour and for determining the properties of systems. These quantities contribute to define both the model of systems from different perspectives (architecture, language design, semantics) and the methodologies and tools for the analysis and verification of system properties. In the last two decades several tool-supported developments, novel frameworks, and theoretical results in this area have been achieved. This is also witnessed by the amount of domains, ranging from security to biology and from quantum computing to hardware analysis, to which quantitative models and formal verification approaches have been applied.

The objective of this survey is to provide an overview of such achievements, their effects on the research trends, and the open issues they have left, through the lens of QAPL, the International Workshop on Quantitative Aspects of Programming Languages and Systems, born in 2001 as one of the first events in theoretical computer science to focus on the design of models and techniques capturing the various quantitative aspects of computation. Nowadays, the scientific research on such quantitative aspects is mainstream for a majority of communities and leading conferences on the foundations of programming languages and systems.

Through the analysis and classification of the contributions presented at QAPL, we revisit some research topics concerned with the quantitative modeling and verification of programming languages and systems, by offering a survey with respect to the following categories: languages and models, logics, information flow analysis and equivalence checking, model checking, static analysis, and related verification tools.

A. Aldini and H. Wiklicky (Eds.): Quantitative Aspects
of Programming Languages and Systems 2019 (QAPL'19).
EPTCS 312, 2020, pp. 1–19, doi:10.4204/EPTCS.312.1

2 Trends

The early 2000s, which cover the period in which the proceedings of the workshops have been published as volumes in Electronic Notes in Theoretical Computer Science (ENTCS), concentrate on several formal frameworks for the quantitative modeling and analysis of systems, the main representatives of which can be categorized as follows: concurrent constraint programming (6 papers), process algebraic languages (12 papers), modal logics (11 papers) and model checking (6 papers), static analysis based on abstract interpretation (6 papers), behavioural equivalences and their approximations (8 papers), information flow analysis (9 papers). Classical techniques, like numerical analysis and performance evaluation, received attention from time to time. During this period, 72 papers were published, collecting 924 citations, CiteScore equal to 12.83 and h-index equal to 15[1].

In the second decade of 2000s, QAPL proceedings appeared as volumes of the Electronic Proceedings in Theoretical Computer Science (EPTCS). By respecting the same classification emerged above, the research on process algebra (8 papers) and other enhanced formal paradigms (9 papers), behavioural equivalences and information flow analysis (15 papers), has continued its upward trend. Topics like concurrent constraint programming almost disappeared, while it is worth observing that the focus moved significantly towards the study and development of a variety of efficient analysis techniques, supported by automated tools, in the setting of static program analysis and abstractions (8 papers), model checking (9 papers), game theory (7 papers), numerical and approximate analysis of Markovian and hybrid systems (8 papers). During this period, 58 published papers produced 261 citations, corresponding to CiteScore equal to 4.5 and h-index equal to 9[2].

2.1 Languages and Models

Expressiveness of modeling languages is an important and intriguing issue going beyond the pure functional aspects of sequential and concurrent programming. In many settings, the purpose is not only to model the functionalities of a system, but also to quantify its behaviour in order, e.g., to enable performance analysis.

By enhancing programming language semantics with probabilities, the properties of programs are not ensured with certainty but up to some probability. A commonly used framework for such a kind of extension is given by the probabilistic guarded command language pGCL and its related quantitative logic. In such a setting, properties are expressed through a logic of real-valued functions, called expectations, and several techniques are used to verify them efficiently [85], including abstractions [114].

Concurrent constraint logic programming is a paradigm for reasoning about concurrency tied to logic and for computing with constraints [124], which represent partial information about variables that describe every state of the program execution. In the setting of constraint logic programming language, a general complexity meta-theorem for the worst-case time complexity of programs is known [69]. Timed extensions of concurrent constraint programming language introduce temporal aspects enabling the model checking analysis of timing properties of concurrent systems (see, e.g., [29], where both discrete and continuous time are modeled). As usual in such a kind of extensions, abstract techniques are defined to deal with the state space explosion problem [8] or to enhance the analysis capabilities, e.g. through semantics based on ordinary differential equations [35, 30]. Constraint logic programming is also combined with constraint-semiring structures, which consist of a domain (representing the values

---

[1] Source: Scopus in May 2019.

[2] The 2011 proceedings collecting 11 papers are not indexed and, therefore, not included in the statistics. Source: Scopus in May 2019.

related to constraints), an additive operation (for projecting constraints), and a multiplicative operation (for combining constraints), so that to enable modeling of quantitative aspects such as costs [27].

A lot of work has been done to endow process algebraic approaches with quantitative elements describing temporal and/or probabilistic behaviours. Such studies go back to the early 1990s and reached their full maturity in the 2000s, where it is easy to find an amount of extensions and variants of both qualitative and quantitative languages. In the setting of QAPL, we can mention several such examples: probabilistic variant of the process-algebraic μCRL language [90], discrete time variant of distributed π-calculus [48], probabilistic extension of π-calculus [117], interleaving semantics and true concurrent semantics for the probabilistic variant of π-calculus [142], stochastic broadcast π-calculus [131], stochastic version of Mobile Ambient [143], stochastic extension of the hybrid process algebra HYPE [32, 33, 68], stochastic extension of the Software Component Ensemble Language for modeling ensemble based autonomous systems [101], Linda-like coordination calculus extended with quantitative information [38], and finally a mixture of concurrent and probabilistic Kleene algebras enriched with probabilistic choices [110]. Further examples include process calculi for performance evaluation, like LYSA [28], proposed for the context of cryptographic protocols, CARMA [31], specifically defined for collective adaptive systems, MELA [107], for modeling in ecology with location attributes, and PEPA Queues [13], introduced for the modeling of queueing networks with mobility features. Moreover, PADS [116] is a process algebraic framework, inspired by real-time process algebra, for reasoning compositionally in a component-based fashion about resource demand and supply.

A specific research topic that is worth mentioning separately is concerned with the formal description of the behaviour of biological systems. Since the first seminal works on the modeling and analysis of biological complex systems through process algebra [119], several amenable extensions of process calculi have been investigated with the aim of faithfully specifying biological quantitative phenomena. In most cases, they are defined in a stochastic framework supporting numerical analysis and classical simulation techniques [73]. These studies include stochastic versions of BioAmbients for the modeling and simulation of biochemical reactions [17] and of Beta-binders for the modeling and quantitative analysis of biological systems [55], another variant of BioAmbients with context-dependent rates and ambient volumes [36], and the stochastic calculus of wrapped compartments [51].

To facilitate performance modeling of systems by avoiding the technicalities of process algebra, both abstraction methods (e.g., from C source code to PEPA models [129]) and mappings between high-level formalisms (e.g., from algorithmic skeletons to PEPA models [147], and from the nanoκ formalism for the modeling of biochemical systems to the Stochastic Pi Machine, a simulator for the stochastic π calculus [96]) may help the validation task performed by the software designer/programmer. Alternative languages are proposed as a bridge between light-weight and rigorous formalisms, like the rewrite-based specification language PMAUDE [2], encompassing both formal basis relying on probabilistic rewrite theories and characteristics of high-level programming languages, and supporting the specification of probabilistic concurrent systems, discrete-event simulation, and statistical analysis.

The relations among languages and models from the expressiveness standpoint, as well as the definition of unifying theories, have been widely investigated in the purely nondeterministic setting, while they still represent ongoing work in many quantitative scenarios. To cite few examples taken from the QAPL literature, the expressiveness comparison among languages, as proposed by Shapiro by defining embeddings among concurrent programming languages [127], can be extended to introduce quantitative estimates of the expressiveness similarities [39]. This is done through a notion of linear embedding by taking linear spaces as semantic domains and, therefore, by employing a linear semantics which associates a linear operator to each program, based on which it is possible to construct a partial order over the languages (in the specific case, a family of Linda-like languages). As another example, adding time to both membrane systems and Petri nets with localities does not increase the expressiveness of the corresponding untimed models, and a operational correspondence between these timed formalisms can be established [10]. An interesting general formal framework is given by a modal specification theory for combined probabilistic timed systems, called abstract probabilistic timed automata, which generalizes existing formalisms [80]. Finally, with the aim of setting the base for general frameworks behind labelled transition systems with quantitative aspects, a unifying theory for nondeterministic processes with quantitative aspects, based on a general GSOS specification format, is proposed with a related notion of bisimulation, which induces labelled transition systems according to the general model of ULTraS [111].

2.2 Information flow analysis and equivalence checking

Information flow analysis and the problem of checking the leakage of sensitive data for programs received great attention, especially since the development of automatic techniques that, in the early 90s, allowed for the systematic verification of noninterference properties of security protocols and programs. Methods to determine bounds on the amount of information leaked represented a natural extension of the classical nondeterministic approaches, and the first sophisticated, formal attempts in the setting of operational models of computation go back to Gray [76], even if it is necessary to wait another decade to see approaches dealing with quantitative analyses at the level of the syntax of programming languages, thus posing the base for automatic analysis of programs [144, 123, 60, 49]. Most of these approaches can be categorized either as entropy-based information theoretic or as inspired by probability theory. In particular, the seminal work by Clark et al. [49] uses information theory, in Shannon's sense, to analyse bounds on the actual leakage of confidential information and, nowadays, it is still the most referenced paper in the history of QAPL, with 102 citations[3]. The same approach is then extended two years later to deal with loops and unbounded iterations [50], and more recently to improve scalability [93] and to trade for the relation existing between scalability issues of large programs and exactness of the bounds of leakage estimated [113]. The basic idea is to face the state space explosion through probabilistic abstract semantics and then to estimate the effects of the abstraction on the information flow measurement, by finding related leakage upper bounds. Analogously, the same quantitative information flow approach is used in the setting of multi-threaded programs [115, 92], where (probabilistic) scheduling policies come into place, and to compute upper and lower bounds in the more general setting of hyperproperties [148]. Entropy-based approaches are commonly used also in the setting of security protocol verification, when moving from the classical symbolic approach of the Dolev-Yao model to the computational model of security [112].

The noninterference approach to information flow analysis has been subject to a wide amount of work in the literature of quantitative aspects of systems, and the classical quantitative extension of such an approach derives from the analysis of probabilistic [7], timing [19, 128, 46], or a mixture of both [98] behaviours. The various approaches differ for the modeling framework, ranging from imperative programming languages to automata and process calculi, and for the notions of equivalence or similarity used for analysis purposes, ranging from ad-hoc state-of-the-memory notions to bisimulation relations.

One of the first approaches to time-sensitive noninterference for object-oriented programming languages dealing with both exceptions and method calls is based on an equality notion for memories and a program transformation eliminating timing leaks [19], which is still of inspiration for several studies in time-sensitive secure information flow analysis [89] and in more practical settings, like Java [88], cloud and edge computing [20].

---

[3] Source: Scopus, May 2019.

In the probabilistic setting, the amount of information leakage is determined in terms of the probability of observing a security violation and, in the case the system execution is (partially) under the control of an adversary (which is the typical scenario whenever the system exposes a mixture of external nondeterministic behaviours and probabilistic behaviours), the goal becomes to estimate the maximum leakage for the most powerful adversary. Such an objective can be achieved, for instance, in the framework of probabilistic formal paradigms, like process algebra [7], and exploiting quantitative notions of equivalences, either exact or approximate.

The noninterference-based security analysis can be viewed as an instance of a more general problem related to the quantitative comparison of the similarities among systems. In fact, in the specific setting of information flow, the models under comparison represent different versions of the same system with and without the interference of events, behaviours, and/or agents that may cause the undesired leakage. In general, the study of equivalence relations and of alternative notions approximating the exactness conditions of such relations played a fundamental role, and it is still mainstream, in the field of quantitative analysis.

As far as quantitative notions of equivalence relations are concerned, in the last 20 years several different semantics have been considered, as also witnessed in the context of QAPL, which in such a sense proposes interesting representatives of the lines of research in this field. Such studies include linear-time equivalences, like trace equivalence for continuous-time Markov chains [145] and for interactive Markov chains [146]; probabilistic barbed congruence [57], which coincides with observational equivalence for a version of CCS including a probabilistic guarded choice operator, branching bisimulation congruence for probabilistic transition systems obeying a general alternating model of probabilistic and nondeterministic states [11] and for a more general probabilistic transition system specification format [104], weak bisimulation for continuous-time Markov chains [23] and for Markov automata [6]; testing equivalence for reactive probabilistic processes [71] and for nondeterministic, probabilistic, and Markovian processes [22], reward-based testing preorders for probabilistic labeled transition systems [59]; finally, a spectrum of different probabilistic equivalences, including trace, bisimulation, and testing semantics, in the setting of nondeterministic and probabilistic processes [24], a generalized notion of bisimulation for state-to-function transition systems that is comparable to many other quantitative notions of bisimulation [103], and undecidability results of bisimulation on Petri nets under durational semantics [100].

When dealing with quantities, a natural extension of equivalence checking is to relax the exactness condition based on which all the equivalence relations are defined and to consider similarities among processes differing for negligible details. In this setting, a typical approach borrowed from pure mathematics relies on the use of metrics, or pseudo-metrics (see, e.g., [58] for a survey related to the Kantorovich metric), which provide a measure of the distance between non-equivalent processes. In particular, starting from the first developments [72], several definitions converging to bisimilarity have been proposed for various models encompassing in different ways probabilities and nondeterminism [56, 149, 108], sometimes enriched with characterizing logics [136, 43], while others rely on alternative semantics, as in the case of linear/branching distances [67, 4, 66, 44], as well as game-based simulation preorders [45].

2.3 Logics, with time and probability

The need for assessing quantitative characteristics of programs and systems has fostered not only the development of formal modeling methodologies for behavioural specification, but also the definition of companion modeling notations for property specification.

Several studies concern extensions of classical linear-time and branching-time logics. In the case of Computation Tree Logic (CTL), we mention relational abstraction techniques [86] for hybrid CTL and hybrid Kripke structures[4], and for the probabilistic extension of the logic, PCTL, and infinite-state labelled Markov chains [87]. We then have exogenous probabilistic CTL [18], which is characterized by an exogenous semantics in which the models of state formulas are probability distributions of models of a propositional logic. Since probabilistic CTL is not expressive enough to reason about probabilities of sequences of formulas (like, e.g., stating that the probability that p will hold for some time is greater than the probability that q will hold for the same amount of time), more expressive branching probabilistic logics have been proposed, like PPL [140], which is also enriched with a sound and complete axiomatization.

In the case of Linear Temporal Logic (LTL), one possible quantitative extension is equipped with verification algorithms over quantitative versions of Kripke structures and of Markov chains [65].

The μ-calculus represents another important framework for quantitative extensions, and in [109] it is shown that stochastic games can be defined in the setting of the quantitative version of the logic, with applications to the economic domain. In order to generalize the μ-calculus and CTL to consider probabilistic aspects, it is also possible to define extensions over constraint-semirings [106]. The evaluation of logic formulas is then given by values of constraint-semirings, thus enabling in the same formalism the analysis of qualitative and quantitative aspects. Such an approach can be applied to model quantitative spatial aspects [47], security conditions [108], and Quality of Service (QoS) properties [84]. QoS, together with mobility and distribution awareness, are the main subject of the logic MoSL [54], a temporal logic for the stochastic variant of the process algebra KLAIM, a large fragment of which can be translated to the Continuous Stochastic Logic (CSL).

Usability represents a fundamental issue also in the setting of property specification, as often the gap between the practitioner and the model checking tool prevents the former from using the latter successfully. Solutions in the quantitative context include pattern systems [77] for real-time properties expressed in the formalism of timed automata, and a logic for the component-oriented specification of reward-based stochastic-time measures [5], that builds on a simple first-order logic by means of which rewards are attached to the states and the transitions of the continuous-time Markov chains (CTMCs) underlying component-oriented system models.

2.4 Analysis techniques and tools: program analysis, model checking, et al.

Formal verification techniques relying on precise mathematical models provide the base for automated analysis, which is a fundamental achievement to bridge the gap from theory to real-world applications.

The goal of program analysis is to establish the properties of a program without executing it. One classical approach to semantics-based program analysis is through abstract interpretation, which has been extended in several different ways, including to take into account probabilistic behaviours. The probabilistic extension leads to a semantics where programs are modeled as linear operators represented by stochastic matrices [63] and, as shown for the probabilistic λ-calculus, turns out to extend in a natural way the classical framework [81]. Among the applications of such an approach we mention the verification of resource consumption properties in the setting of semirings [132], data flow analysis [61] and precision analysis [62] in the setting of the probabilistic While imperative language. An analogous approach very close in principle is adopted to analyze quantitative properties with regard to the use of resources such as time and memory [41]. An alternative approach, aiming to improve scalability, is based on the abstract interpretation of a probabilistic automaton semantics for a simple imperative language [130]. Alternative (numerical) program analysis techniques include robustness analysis of program output with

---

[4] We recall that hybrid approaches enrich temporal logics and their models with the ability to name and track model states.

regard to input variations [70], probabilistic output analysis based on the input distributions [121], lifting analysis of resource consumption from compiled to source code [137], timing analysis of programs with real-time constraints [105], and cost analysis of object-oriented bytecode programs [3], supported by the software tool COSTA.

In contrast to static analysis, model checking is intended to verify whether a program satisfies a property by exploring in a systematic way the state space associated with the program. Quantitative model checking is concerned with quantities, in most cases probabilities. The resulting algorithmic approach is accompanied by several automated software tools, among which the probabilistic model checker PRISM [95] is one of the representatives used in a wide range of application domains, including security [97], safety [78] and reliability of communication protocols [150]. Moreover, PRISM has been extended to deal with large, complex systems, by using, e.g., game-based abstractions [91] and efficient algorithms based on linear programming for the analysis of Markov decision processes (MDPs) [74].

In general, facing the state space explosion problem is an issue addressed in the context of both purely functional and quantitative systems, and with respect to methods like model checking of linear time and branching time properties. Among the proposed techniques, we mention confluence reductions and partial order reductions for branching semantics [16, 82, 135], and algorithms to compute the progress of linear time model checking [52]. Alternative abstraction methods are proposed in the setting of timed automata [9] and implemented for extensions of the real-time verification tool UPPAAL. In this framework, statistical model checking [40] is another approach used to trade between testing and formal verification.

In the last years, scalability of analysis has become even more relevant because of the application of tool-supported, formal verification techniques to domains characterized by big data evolving in time and space, like computational biology and collective adaptive systems [102]. A widely used formalism is that of Markov population models (MPMs), for which ad-hoc algorithms for computing transient distributions [12] and to model check CSL properties [133, 34] have been implemented. Enhancing scalability is the main improvement of alternative methods of performance analysis for very large systems, e.g., based on fluid approximation techniques [83, 138] and on solving systems of ordinary differential equations (ODEs) [35], which is an approach allowing for producing approximated transient measures for models of 10^{100} states and beyond, even thanks to (approximate) reductions [141] and aggregations [139] of ODE systems. These methods are supported by automated analysers, like, e.g., GPA [134], the specification language of which is inspired by a version of the process algebra PEPA, and ERODE [42, 141]. Approximation methods [15] and scalable reachability analysis techniques [125] are proposed also for mixed models like probabilistic hybrid automata.

In addition to efficiency, usability is another important issue of model checking, especially with respect to the problem of interpreting the reasons for unsatisfiability, which has been faced in the setting of the temporal resolution prover TRP++ for linear time properties [126].

Other interesting extensions of model checking are concerned with both analysis and modeling issues. For instance, in the former case it is worth mentioning the analysis of infinite-state models (see, e.g., the case of CTMCs and CSL [120], and the local abstraction refinement approach developed for MDPs and reachability properties, and even implemented in PRISM [64]) and of distributed probabilistic input/output automata [75], while in the latter case, some developments emerged in the last decade are given by the analysis of security [1], mobility [53], and of temporal and epistemic properties of quantum systems, the automated verification of which is proposed in the setting of the symbolic model checker MCMAS [21].

Finally, studies on quantitative game theory and related algorithms for determining the optimal strategies [26, 118, 122, 25, 14, 37, 99], conducted for various formalisms, like, e.g., automata, are particularly relevant, as such games can be used as models to define the interaction between a system and its environment on the base of quantitative objectives and behaviours inducing costs, rewards, and resource consumption. The relation with model checking is strict, as also demonstrated by the most recent advances in tools like, e.g., PRISM and MCMAS, implementing model checking algorithms for MDPs founded on game semantics.

Among the other analysis techniques, supported by automated tools, we mention the case of language equivalence for weighted automata and of bisimulation for conditional transition systems, both of which are accompanied with decision algorithms implemented in PAWS [94].

3 Conclusion, challenges, and open issues

The field of quantitative modeling and analysis is still very active both on the foundational aspects and on the side of software tool development. As a representative example concerning analysis techniques, a large amount of work has been done about notions of quantitative bisimilarity for processes encompassing concurrency and nondeterminism, and robust extensions have been devised in the form of behavioural pseudometrics, the kernel of which is a bisimilarity, while in all other cases they assign a distance measuring the degree to which two states are *quasi* bisimilar. Such metrics are defined in terms of several alternative formulations equipped with different (in terms of complexity) algorithms to estimate the distances. From the applicability standpoint, it would be useful to have a comprehensive, comparative analysis with the aim of establishing which notions can be really advantageous in practice. On the theoretical side, the picture of the relations among metrics relying on different theories is not complete yet, for instance from the viewpoint of the logical characterizations, possibly accompanied by characteristic formulae, for such metrics. Moreover, the same effort has not been done yet in the setting of semantics that are alternative to bisimulation, like, e.g., testing.

Static information flow analysis is effective provided that all code is available, while dynamic analysis considers concrete executions rather than abstract code, even if tracking data throughout the execution process may be difficult and time-consuming. As far as dynamic, quantitative analysis is concerned, a large amount of modeling languages and mathematical formalisms are nowadays accompanied by efficient verification algorithms and, therefore, automated tools supporting their use for real-world case studies. A sign of the reached maturity in this setting is given by the first competition among quantitative verification tools, QComp [79], promoted and presented in the context of TACAS during the same edition of ETAPS in which the last QAPL has taken place. The outcome emphasizes the versatility and the expressiveness of the considered tools, which cover an increasing amount of specification languages, mathematical formalisms, quantitative properties, and verification techniques. We think that such a kind of initiatives favours the adoption of specification standards, e.g., by promoting the use of the JANI model exchange format, and performance standards, in order to establish reasonable and expected tradeoffs among expressiveness, tool efficiency, and accuracy of the results obtained.

Finally, it is worth observing that many of the verification techniques discussed in this survey are close and related to other domains, as it is the case, e.g., for probabilistic model checking, with respect to, e.g., probabilistic programming and (Bayesian approaches to) machine learning. Thus, cross-fertilisation of knowledge among the related communities may lead to promising results.

Acknowledgements: It is worth expressing sincere thanks to all the people who organized and put considerable effort into QAPL over the years and, in particular: Alessandra, Herbert, Antonio, Alessandro, Frank, Christel, Gethin, Mieke, Luca, Nathalie, Mirco, and Erik.

References

[1] Pedro Adão, Paulo Mateus, Tiago Reis & Luca Viganò (2006): *Towards a Quantitative Analysis of Security Protocols*. Electronic Notes in Theoretical Computer Science 164(3), pp. 3–25, doi:10.1016/j.entcs.2006.07.009. Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages (QAPL 2006).

[2] Gul Agha, José Meseguer & Koushik Sen (2006): *PMaude: Rewrite-based Specification Language for Probabilistic Object Systems*. Electronic Notes in Theoretical Computer Science 153(2), pp. 213–239, doi:10.1016/j.entcs.2005.10.040. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[3] Elvira Albert, Puri Arenas, Samir Genaim, German Puebla & Damiano Zanardini (2012): *Cost analysis of object-oriented bytecode programs*. Theoretical Computer Science 413(1), pp. 142–159, doi:10.1016/j.tcs.2011.07.009. Quantitative Aspects of Programming Languages (QAPL 2010).

[4] Alessandro Aldini (2012): *Approximating Markovian testing equivalence*. Theoretical Computer Science 413(1), pp. 73–86, doi:10.1016/j.tcs.2011.07.019. Quantitative Aspects of Programming Languages (QAPL 2010).

[5] Alessandro Aldini & Marco Bernardo (2007): *Mixing logics and rewards for the component-oriented specification of performance measures*. Theor. Comput. Sci. 382(1), pp. 3–23, doi:10.1016/j.tcs.2007.05.006. Quantitative Aspects of Programming Languages.

[6] Alessandro Aldini & Marco Bernardo (2015): *Expected-Delay-Summing Weak Bisimilarity for Markov Automata*. In: Proceedings Thirteenth Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 194, pp. 1–15, doi:10.4204/EPTCS.194.1.

[7] Alessandro Aldini & Alessandra Di Pierro (2005): *On Quantitative Analysis of Probabilistic Protocols*. Electronic Notes in Theoretical Computer Science 112, pp. 131–148, doi:10.1016/j.entcs.2004.01.019. Proceedings of the Second Workshop on Quantitative Aspects of Programming Languages (QAPL 2004).

[8] María Alpuente, María-del-Mar Gallardo, Ernesto Pimentel & Alicia Villanueva (2005): *A semantic framework for the abstract model checking of tccp programs*. Theor. Comput. Sci. 346(1), pp. 58–95, doi:10.1016/j.tcs.2005.08.009. Quantitative Aspects of Programming Languages.

[9] Karine Altisen, Yanhong Liu & Matthieu Moy (2010): *Performance Evaluation of Components Using a Granularity-based Interface Between Real-Time Calculus and Timed Automata*. In: Proceedings Eighth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 28, pp. 16–33, doi:10.4204/EPTCS.28.2.

[10] Bogdan Aman & Gabriel Ciobanu (2011): *Time Delays in Membrane Systems and Petri Nets*. In: Proceedings Ninth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 57, pp. 47–60, doi:10.4204/EPTCS.57.4.

[11] Suzana Andova, Sonja Georgievska & Nikola Trčka (2012): *Branching bisimulation congruence for probabilistic systems*. Theoretical Computer Science 413(1), pp. 58–72, doi:10.1016/j.tcs.2011.07.020. Quantitative Aspects of Programming Languages (QAPL 2010).

[12] Aleksandr Andreychenko, Pepijn Crouzen, Linar Mikeev & Verena Wolf (2011): *On-the-fly Uniformization of Time-Inhomogeneous Infinite Markov Population Models*. In: Proceedings Ninth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 57, pp. 1–15, doi:10.4204/EPTCS.57.1.

[13] Ashok Argent-Katwala & Jeremy T. Bradley (2007): *PEPA Queues: Capturing Customer Behaviour in Queueing Networks*. Electronic Notes in Theoretical Computer Science 190(3), pp. 3–25, doi:10.1016/j.entcs.2007.07.002. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[14] Arjun Arul & Julien Reichert (2013): *The Complexity of Robot Games on the Integer Line*. In: Proceedings 11th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 117, pp. 132–148, doi:10.4204/EPTCS.117.9.

[15] Joseph Assouramou & Josée Desharnais (2011): *Analysis of Non-Linear Probabilistic Hybrid Systems*. In: Proceedings Ninth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 57, pp. 104–119, doi:10.4204/EPTCS.57.8.

[16] Christel Baier, Pedro D'Argenio & Marcus Groesser (2006): *Partial Order Reduction for Probabilistic Branching Time*. Electronic Notes in Theoretical Computer Science 153(2), pp. 97–116, doi:10.1016/j.entcs.2005.10.034. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[17] Steffen van Bakel, Ikra Khan, Maria Grazia Vigliotti & John K. Heath (2008): *Modelling Intracellular Fate of FGF Receptors With BioAmbients*. Electronic Notes in Theoretical Computer Science 220(3), pp. 181–197, doi:10.1016/j.entcs.2008.11.026. Proceedings of the Sixth Workshop on Quantitative Aspects of Programming Languages (QAPL 2008).

[18] Pedro Baltazar, Paulo Mateus, Rajagopal Nagarajan & Nikolaos Papanikolaou (2007): *Exogenous Probabilistic Computation Tree Logic*. Electronic Notes in Theoretical Computer Science 190(3), pp. 95–110, doi:10.1016/j.entcs.2007.07.007. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[19] Gilles Barthe, Tamara Rezk & Martijn Warnier (2006): *Preventing Timing Leaks Through Transactional Branching Instructions*. Electronic Notes in Theoretical Computer Science 153(2), pp. 33–55, doi:10.1016/j.entcs.2005.10.031. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[20] Mohammad-Mahdi Bazm, Marc Lacoste, Mario Südholt & Jean-Marc Menaud (2019): *Isolation in cloud computing infrastructures: new security challenges*. Annales des Telecommunications/Annals of Telecommunications 74(3-4), pp. 197–209, doi:10.1007/s12243-019-00703-z.

[21] Francesco Belardinelli, Pavel Gonzalez & Alessio Lomuscio (2012): *Automated Verification of Quantum Protocols using MCMAS*. In: Proceedings 10th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 85, pp. 48–62, doi:10.4204/EPTCS.85.4.

[22] Marco Bernardo (2009): *Uniform Logical Characterizations of Testing Equivalences for Nondeterministic, Probabilistic and Markovian Processes*. Electronic Notes in Theoretical Computer Science 253(3), pp. 3–23, doi:10.1016/j.entcs.2009.10.002. Proceedings of Seventh Workshop on Quantitative Aspects of Programming Languages (QAPL 2009).

[23] Marco Bernardo (2012): *Weak Markovian Bisimulation Congruences and Exact CTMC-Level Aggregations for Concurrent Processes*. In: Proceedings 10th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 85, pp. 122–136, doi:10.4204/EPTCS.85.9.

[24] Marco Bernardo, Rocco De Nicola & Michele Loreti (2013): *The Spectrum of Strong Behavioral Equivalences for Nondeterministic and Probabilistic Processes*. In: Proceedings 11th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 117, pp. 81–96, doi:10.4204/EPTCS.117.6.

[25] Nathalie Bertrand & Philippe Schnoebelen (2013): *Solving Stochastic Büchi Games on Infinite Arenas with a Finite Attractor*. In: Proceedings 11th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 117, pp. 116–131, doi:10.4204/EPTCS.117.8.

[26] Alessandro Bianco, Marco Faella, Fabio Mogavero & Aniello Murano (2012): *Quantitatively fair scheduling*. Theoretical Computer Science 413(1), pp. 160–175, doi:10.1016/j.tcs.2011.06.029. Quantitative Aspects of Programming Languages (QAPL 2010).

[27] Stefano Bistarelli, Ugo Montanari, Francesca Rossi & Francesco Santini (2007): *Modelling Multicast QoS Routing by using Best-Tree Search in And-or Graphs and Soft Constraint Logic Programming*. *Electronic Notes in Theoretical Computer Science* 190(3), pp. 111–127, doi:10.1016/j.entcs.2007.07.008. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[28] Chiara Bodei, Michele Curti, Pierpaolo Degano, Mikael Buchholtz, Flemming Nielson, Hanne Riis Nielson & Corrado Priami (2005): *Performance Evaluation of Security Protocols Specified in LySa*. *Electronic Notes in Theoretical Computer Science* 112, pp. 167–189, doi:10.1016/j.entcs.2004.01.017. Proceedings of the Second Workshop on Quantitative Aspects of Programming Languages (QAPL 2004).

[29] Luca Bortolussi (2006): *Stochastic Concurrent Constraint Programming*. *Electronic Notes in Theoretical Computer Science* 164(3), pp. 65–80, doi:10.1016/j.entcs.2006.07.012. Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages (QAPL 2006).

[30] Luca Bortolussi (2008): *On the Approximation of Stochastic Concurrent Constraint Programming by Master Equation*. *Electronic Notes in Theoretical Computer Science* 220(3), pp. 163–180, doi:10.1016/j.entcs.2008.11.025. Proceedings of the Sixth Workshop on Quantitative Aspects of Programming Languages (QAPL 2008).

[31] Luca Bortolussi, Rocco De Nicola, Vashti Galpin, Stephen Gilmore, Jane Hillston, Diego Latella, Michele Loreti & Mieke Massink (2015): *CARMA: Collective Adaptive Resource-sharing Markovian Agents*. In: Proceedings Thirteenth Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 194, pp. 16–31, doi:10.4204/EPTCS.194.2.

[32] Luca Bortolussi, Vashti Galpin & Jane Hillston (2011): *HYPE with stochastic events*. In: Proceedings Ninth Workshop on *Quantitative Aspects of Programming Languages*, *Electronic Proceedings in Theoretical Computer Science* 57, pp. 120–133, doi:10.4204/EPTCS.57.9.

[33] Luca Bortolussi, Vashti Galpin & Jane Hillston (2012): *Hybrid performance modelling of opportunistic networks*. In: Proceedings 10th Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 85, pp. 106–121, doi:10.4204/EPTCS.85.8.

[34] Luca Bortolussi & Jane Hillston (2015): *Efficient Checking of Individual Rewards Properties in Markov Population Models*. In: Proceedings Thirteenth Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 194, pp. 32–47, doi:10.4204/EPTCS.194.3.

[35] Luca Bortolussi & Alberto Policriti (2007): *Stochastic Concurrent Constraint Programming and Differential Equations*. *Electronic Notes in Theoretical Computer Science* 190(3), pp. 27–42, doi:10.1016/j.entcs.2007.07.003. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[36] Luca Bortolussi & Maria Grazia Vigliotti (2009): *CoBiC: Context-dependent Bioambient Calculus*. *Electronic Notes in Theoretical Computer Science* 253(3), pp. 187–201, doi:10.1016/j.entcs.2009.10.012. Proceedings of Seventh Workshop on Quantitative Aspects of Programming Languages (QAPL 2009).

[37] Bettina Braitling, Luis María Ferrer Fioriti, Hassan Hatefi, Ralf Wimmer, Bernd Becker & Holger Hermanns (2014): *MeGARA: Menu-based Game Abstraction and Abstraction Refinement of Markov Automata*. In: Proceedings Twelfth Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 154, pp. 48–63, doi:10.4204/EPTCS.154.4.

[38] Mario Bravetti, Roberto Gorrieri, Roberto Lucchi & Gianluigi Zavattaro (2005): *Quantitative information in the tuple space coordination model*. *Theor. Comput. Sci.* 346(1), pp. 28–57, doi:10.1016/j.tcs.2005.08.004.

[39] Antonio Brogi, Alessandra Di Pierro & Herbert Wiklicky (2002): *Linear Embedding for a Quantitative Comparison of Language Expressiveness*. *Electronic Notes in Theoretical Computer Science* 59(3), pp. 207–237, doi:10.1016/S1571-0661(04)00289-0. QAPL'01, Quantitative Aspects of Programming Languages.

[40] Peter Bulychev, Alexandre David, Kim Gulstrand Larsen, Marius Mikučionis, Danny Bøgsted Poulsen, Axel Legay & Zheng Wang (2012): *UPPAAL-SMC: Statistical Model Checking for Priced Timed Au-*

tomata. In: Proceedings 10th Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 85, pp. 1–16, doi:10.4204/EPTCS.85.1.

[41] David Cachera & Arnaud Jobin (2010): *Injecting Abstract Interpretations into Linear Cost Models*. In: Proceedings Eighth Workshop on *Quantitative Aspects of Programming Languages*, *Electronic Proceedings in Theoretical Computer Science* 28, pp. 64–81, doi:10.4204/EPTCS.28.5.

[42] Luca Cardelli, Mirco Tribastone, Max Tschaikowski & Andrea Vandin (2017): *ERODE: A Tool for the Evaluation and Reduction of Ordinary Differential Equations*. In Axel Legay & Tiziana Margaria, editors: *Tools and Algorithms for the Construction and Analysis of Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 310–328, doi:10.1007/978-3-662-54580-5_19.

[43] Valentina Castiglioni, Daniel Gebler & Simone Tini (2016): *Logical Characterization of Bisimulation Metrics*. In: Proceedings 14th Workshop *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 227, pp. 44–62, doi:10.4204/EPTCS.227.4.

[44] Valentina Castiglioni & Simone Tini (2017): *Logical Characterization of Trace Metrics*. In: Proceedings 15th Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 250, pp. 39–74, doi:10.4204/EPTCS.250.4.

[45] Pavol Černý, Thomas A. Henzinger & Arjun Radhakrishna (2012): *Simulation distances*. *Theoretical Computer Science* 413(1), pp. 21–35, doi:10.1016/j.tcs.2011.08.002. Quantitative Aspects of Programming Languages (QAPL 2010).

[46] Tom Chothia, Jun Pang & Mohammad Torabi Dashti (2007): *Keeping Secrets in Resource Aware Components*. *Electronic Notes in Theoretical Computer Science* 190(3), pp. 79–94, doi:10.1016/j.entcs.2007.07.006. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[47] Vincenzo Ciancia & Gian Luigi Ferrari (2007): *Co-Algebraic Models for Quantitative Spatial Logics*. *Electronic Notes in Theoretical Computer Science* 190(3), pp. 43–58, doi:10.1016/j.entcs.2007.07.004. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[48] Gabriel Ciobanu & Cristian Prisacariu (2006): *Timers for Distributed Systems*. *Electronic Notes in Theoretical Computer Science* 164(3), pp. 81–99, doi:10.1016/j.entcs.2006.07.013. Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages (QAPL 2006).

[49] David Clark, Sebastian Hunt & Pasquale Malacaria (2002): *Quantitative Analysis of the Leakage of Confidential Data*. *Electronic Notes in Theoretical Computer Science* 59(3), pp. 238–251, doi:10.1016/S1571-0661(04)00290-7. QAPL'01, Quantitative Aspects of Programming Languages.

[50] David Clark, Sebastian Hunt & Pasquale Malacaria (2005): *Quantified Interference for a While Language*. *Electronic Notes in Theoretical Computer Science* 112, pp. 149–166, doi:10.1016/j.entcs.2004.01.018. Proceedings of the Second Workshop on Quantitative Aspects of Programming Languages (QAPL 2004).

[51] Mario Coppo, Ferruccio Damiani, Maurizio Drocco, Elena Grassi & Angelo Troina (2010): *Stochastic Calculus of Wrapped Compartments*. In: Proceedings Eighth Workshop on *Quantitative Aspects of Programming Languages*, *Electronic Proceedings in Theoretical Computer Science* 28, pp. 82–98, doi:10.4204/EPTCS.28.6.

[52] Elise Cormie-Bowins & Franck van Breugel (2012): *Measuring Progress of Probabilistic LTL Model Checking*. In: Proceedings 10th Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 85, pp. 33–47, doi:10.4204/EPTCS.85.3.

[53] Rocco De Nicola, Joost-Pieter Katoen, Diego Latella, Michele Loreti & Mieke Massink (2007): *Model checking mobile stochastic logic*. *Theoretical Computer Science* 382(1), pp. 42–70, doi:10.1016/j.tcs.2007.05.008. Quantitative Aspects of Programming Languages.

[54] Rocco De Nicola, Joost-Pieter Katoen, Diego Latella & Mieke Massink (2006): *Towards a Logic for Performance and Mobility*. *Electronic Notes in Theoretical Computer Science* 153(2), pp. 161–175, doi:10.1016/j.entcs.2005.10.037. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[55] Pierpaolo Degano, Davide Prandi, Corrado Priami & Paola Quaglia (2006): *Beta-binders for Biological Quantitative Experiments*. Electronic Notes in Theoretical Computer Science 164(3), pp. 101–117, doi:10.1016/j.entcs.2006.07.014. Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages (QAPL 2006).

[56] Yuxin Deng, Tom Chothia, Catuscia Palamidessi & Jun Pang (2006): *Metrics for Action-labelled Quantitative Transition Systems*. Electronic Notes in Theoretical Computer Science 153(2), pp. 79–96, doi:10.1016/j.entcs.2005.10.033. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[57] Yuxin Deng & Wenjie Du (2007): *Probabilistic Barbed Congruence*. Electronic Notes in Theoretical Computer Science 190(3), pp. 185–203, doi:10.1016/j.entcs.2007.07.011. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[58] Yuxin Deng & Wenjie Du (2009): *The Kantorovich Metric in Computer Science: A Brief Survey*. Electronic Notes in Theoretical Computer Science 253(3), pp. 73–82, doi:10.1016/j.entcs.2009.10.006. Proceedings of Seventh Workshop on Quantitative Aspects of Programming Languages (QAPL 2009).

[59] Yuxin Deng, Rob van Glabbeek, Matthew Hennessy & Carroll Morgan (2014): *Real-reward testing for probabilistic processes*. Theoretical Computer Science 538, pp. 16–36, doi:10.1016/j.tcs.2013.07.016. Quantitative Aspects of Programming Languages and Systems (2011-12).

[60] Alessandra Di Pierro, Chris Hankin & Herbert Wiklicky (2002): *Approximate Non-Interference*. In: 15th IEEE Workshop on Computer Security Foundations, pp. 3–17, doi:10.1109/CSFW.2002.1021803.

[61] Alessandra Di Pierro, Chris Hankin & Herbert Wiklicky (2007): *On Probabilistic Techniques for Data Flow Analysis*. Electronic Notes in Theoretical Computer Science 190(3), pp. 59–77, doi:10.1016/j.entcs.2007.07.005. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[62] Alessandra Di Pierro, Pascal Sotin & Herbert Wiklicky (2008): *Relational Analysis and Precision via Probabilistic Abstract Interpretation*. Electronic Notes in Theoretical Computer Science 220(3), pp. 23–42, doi:10.1016/j.entcs.2008.11.017. Proceedings of the Sixth Workshop on Quantitative Aspects of Programming Languages (QAPL 2008).

[63] Alessandra Di Pierro & Herbert Wiklicky (2000): *Concurrent constraint programming: towards probabilistic abstract interpretation*. In: Proceedings of the 2nd ACM SIGPLAN conference on on Principles and Practice of Declarative Programming, pp. 127–138, doi:10.1145/351268.351284.

[64] Klaus Dräger, Marta Kwiatkowska, David Parker & Hongyang Qu (2014): *Local abstraction refinement for probabilistic timed programs*. Theoretical Computer Science 538, pp. 37–53, doi:10.1016/j.tcs.2013.07.013. Quantitative Aspects of Programming Languages and Systems (2011-12).

[65] Marco Faella, Axel Legay & Marielle Stoelinga (2008): *Model Checking Quantitative Linear Time Logic*. Electronic Notes in Theoretical Computer Science 220(3), pp. 61–77, doi:10.1016/j.entcs.2008.11.019. Proceedings of the Sixth Workshop on Quantitative Aspects of Programming Languages (QAPL 2008).

[66] Uli Fahrenberg & Axel Legay (2014): *The quantitative linear-time–branching-time spectrum*. Theoretical Computer Science 538, pp. 54–69, doi:10.1016/j.tcs.2013.07.030. Quantitative Aspects of Programming Languages and Systems (2011-12).

[67] Uli Fahrenberg, Claus Thrane & Kim G. Larsen (2011): *Distances for Weighted Transition Systems: Games and Properties*. In: Proceedings Ninth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 57, pp. 134–147, doi:10.4204/EPTCS.57.10.

[68] Cheng Feng (2014): *Patch-based Hybrid Modelling of Spatially Distributed Systems by Using Stochastic HYPE - ZebraNet as an Example*. In: Proceedings Twelfth Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 154, pp. 64–77, doi:10.4204/EPTCS.154.5.

[69] Thom Frühwirth (2002): *As Time Goes By II: More Automatic Complexity Analysis of Concurrent Rule Programs*. Electronic Notes in Theoretical Computer Science 59(3), pp. 185–206, doi:10.1016/S1571-0661(04)00288-9. QAPL'01, Quantitative Aspects of Programming Languages.

[70] Ivan Gazeau, Dale Miller & Catuscia Palamidessi (2012): *A non-local method for robustness analysis of floating point programs*. In: Proceedings 10th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 85, pp. 63–76, doi:10.4204/EPTCS.85.5.

[71] Sonja Georgievska & Suzana Andova (2010): *Testing Reactive Probabilistic Processes*. In: Proceedings Eighth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 28, pp. 99–113, doi:10.4204/EPTCS.28.7.

[72] Alessandro Giacalone, Chi-Chang Jou & Scott A. Smolka (1990): *Algebraic Reasoning for Probabilistic Concurrent Systems*. In: Programming concepts and methods: Proceedings of the IFIP Working Group 2.2, 2.3 Working Conference on Programming Concepts and Methods, pp. 443–458. Available at `https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.3664&rank=1`.

[73] Daniel T. Gillespie (1977): *Exact stochastic simulation of coupled chemical reactions*. The Journal of Physical Chemistry 81(25), pp. 2340–2361, doi:10.1021/j100540a008.

[74] Sergio Giro (2014): *Optimal schedulers vs optimal bases: An approach for efficient exact solving of Markov decision processes*. Theoretical Computer Science 538, pp. 70–83, doi:10.1016/j.tcs.2013.08.020. Quantitative Aspects of Programming Languages and Systems (2011-12).

[75] Sergio Giro, Pedro R. D'Argenio & Luis María Ferrer Fioriti (2014): *Distributed probabilistic input/output automata: Expressiveness, (un)decidability and algorithms*. Theor. Comput. Sci. 538, pp. 84–102, doi:10.1016/j.tcs.2013.07.017.

[76] James W. Gray (1991): *Toward a mathematical foundation for information flow security*. In: Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 21–34, doi:10.1109/RISP.1991.130769.

[77] Volker Gruhn & Ralf Laue (2006): *Patterns for Timed Property Specifications*. Electronic Notes in Theoretical Computer Science 153(2), pp. 117–133, doi:10.1016/j.entcs.2005.10.035. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[78] Matthias Güdemann & Frank Ortmeier (2010): *Probabilistic Model-Based Safety Analysis*. In: Proceedings Eighth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 28, pp. 114–128, doi:10.4204/EPTCS.28.8.

[79] Ernst Moritz Hahn, Arnd Hartmanns, Christian Hensel, Michaela Klauck, Joachim Klein, Jan Křetínský, David Parker, Tim Quatmann, Enno Ruijters & Marcel Steinmetz (2019): *The 2019 Comparison of Tools for the Analysis of Quantitative Formal Models*. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), LNCS 11429, Springer, pp. 69–92, doi:10.1007/978-3-030-17502-3_5.

[80] Tingting Han, Christian Krause, Marta Kwiatkowska & Holger Giese (2013): *Modal Specifications for Probabilistic Timed Systems*. In: Proceedings 11th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 117, pp. 66–80, doi:10.4204/EPTCS.117.5.

[81] Chris Hankin & Herbert Wiklicky (2005): *Lambda-calculus and Quantitative Program Analysis: (Extended Abstract)*. Electronic Notes in Theoretical Computer Science 112, pp. 5–18, doi:10.1016/j.entcs.2004.01.016. Proceedings of the Second Workshop on Quantitative Aspects of Programming Languages (QAPL 2004).

[82] Henri Hansen & Mark Timmer (2014): *A comparison of confluence and ample sets in probabilistic and non-probabilistic branching time*. Theoretical Computer Science 538, pp. 103–123, doi:10.1016/j.tcs.2013.07.014. Quantitative Aspects of Programming Languages and Systems (2011-12).

[83] Richard A. Hayden, Anton Stefanek & Jeremy T. Bradley (2012): *Fluid computation of passage-time distributions in large Markov models*. Theoretical Computer Science 413(1), pp. 106–141, doi:10.1016/j.tcs.2011.07.017. Quantitative Aspects of Programming Languages (QAPL 2010).

[84] Dan Hirsch, Alberto Lluch-Lafuente & Emilio Tuosto (2006): *A Logic for Application Level QoS*. Electronic Notes in Theoretical Computer Science 153(2), pp. 135–159, doi:10.1016/j.entcs.2005.10.036. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[85] Joe Hurd, Annabelle McIver & Carroll Morgan (2005): *Probabilistic guarded commands mechanized in* HOL. Theor. Comput. Sci. 346(1), pp. 96–112, doi:10.1016/j.tcs.2005.08.005. Quantitative Aspects of Programming Languages.

[86] Michael Huth (2005): *Abstraction and Probabilities for Hybrid Logics*. Electronic Notes in Theoretical Computer Science 112, pp. 61–76, doi:10.1016/j.entcs.2004.01.023. Proceedings of the Second Workshop on Quantitative Aspects of Programming Languages (QAPL 2004).

[87] Michael Huth (2005): *On finite-state approximants for probabilistic computation tree logic*. Theor. Comput. Sci. 346(1), pp. 113–134, doi:10.1016/j.tcs.2005.08.008. Quantitative Aspects of Programming Languages.

[88] Zeinab Iranmanesh & Mehran S. Fallah (2016): *Specification and static enforcement of scheduler-independent noninterference in a middleweight Java*. Computer Languages, Systems and Structures 46, pp. 20–43, doi:10.1016/j.cl.2016.05.003.

[89] Vineeth Kashyap, Ben Wiedermann & Ben Hardekopf (2011): *Timing- and termination-sensitive secure information flow: Exploring a new approach*. In: Proceedings - IEEE Symposium on Security and Privacy, pp. 413–428, doi:10.1109/SP.2011.19.

[90] Joost-Pieter Katoen, Jaco van de Pol, Marille Stoelinga & Mark Timmer (2012): *A linear process-algebraic format with data for probabilistic automata*. Theoretical Computer Science 413(1), pp. 36–57, doi:10.1016/j.tcs.2011.07.021. Quantitative Aspects of Programming Languages (QAPL 2010).

[91] Mark Kattenbelt, Marta Kwiatkowska, Gethin Norman & David Parker (2008): *Game-Based Probabilistic Predicate Abstraction in PRISM*. Electronic Notes in Theoretical Computer Science 220(3), pp. 5–21, doi:10.1016/j.entcs.2008.11.016. Proceedings of the Sixth Workshop on Quantitative Aspects of Programming Languages (QAPL 2008).

[92] Yusuke Kawamoto & Thomas Given-Wilson (2015): *Quantitative Information Flow for Scheduler-Dependent Systems*. In: Proceedings Thirteenth Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 194, pp. 48–62, doi:10.4204/EPTCS.194.4.

[93] Vladimir Klebanov (2014): *Precise quantitative information flow analysis a symbolic approach*. Theoretical Computer Science 538, pp. 124–139, doi:10.1016/j.tcs.2014.04.022. Quantitative Aspects of Programming Languages and Systems (2011-12).

[94] Barbara König, Sebastian Küpper & Christina Mika (2017): *PAWS: A Tool for the Analysis of Weighted Systems*. In: Proceedings 15th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 250, pp. 75–91, doi:10.4204/EPTCS.250.5.

[95] Marta Kwiatkowska, Gethin Norman & David Parker (2006): *Quantitative Analysis With the Probabilistic Model Checker PRISM*. Electronic Notes in Theoretical Computer Science 153(2), pp. 5–31, doi:10.1016/j.entcs.2005.10.030. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[96] Cosimo Laneve, Sylvain Pradalier & Gianluigi Zavattaro (2009): *From Biochemistry to Stochastic Processes*. Electronic Notes in Theoretical Computer Science 253(3), pp. 167–185, doi:10.1016/j.entcs.2009.10.011. Proceedings of Seventh Workshop on Quantitative Aspects of Programming Languages (QAPL 2009).

[97] Ruggero Lanotte, Andrea Maggiolo-Schettini & Angelo Troina (2005): *Automatic Analysis of a Non-Repudiation Protocol*. Electronic Notes in Theoretical Computer Science 112(SPEC. ISS.), pp. 113–129,

doi:10.1016/j.entcs.2004.01.020. Proceedings of the Second Workshop on Quantitative Aspects of Programming Languages (QAPL 2004).

[98] Ruggero Lanotte, Andrea Maggiolo-Schettini & Angelo Troina (2006): *A Classification of Time and/or Probability Dependent Security Properties*. *Electronic Notes in Theoretical Computer Science* 153(2), pp. 177–193, doi:10.1016/j.entcs.2005.10.038. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[99] Kim G. Larsen, Simon Laursen & Martin Zimmermann (2016): *Limit Your Consumption! Finding Bounds in Average-energy Games*. In: Proceedings 14th Workshop *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 227, pp. 1–14, doi:10.4204/EPTCS.227.1.

[100] Slawomir Lasota & Marcin Poturalski (2016): *Undecidability of performance equivalence of Petri nets*. *Theor. Comput. Sci.* 655, pp. 109–119, doi:10.1016/j.tcs.2016.01.034.

[101] Diego Latella, Michele Loreti, Mieke Massink & Valerio Senni (2014): *Stochastically timed predicate-based communication primitives for autonomic computing*. In: Proceedings Twelfth Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 154, pp. 1–16, doi:10.4204/EPTCS.154.1.

[102] Diego Latella & Mieke Massink (2017): *Design and Optimisation of the FlyFast Front-end for Attribute-based Coordination*. In: Proceedings 15th Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 250, pp. 92–110, doi:10.4204/EPTCS.250.6.

[103] Diego Latella, Mieke Massink & Erik de Vink (2015): *A Definition Scheme for Quantitative Bisimulation*. In: Proceedings Thirteenth Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 194, pp. 63–78, doi:10.4204/EPTCS.194.5.

[104] Matias D. Lee & Erik P. de Vink (2015): *Rooted branching bisimulation as a congruence for probabilistic transition systems*. In: Proceedings Thirteenth Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 194, pp. 79–94, doi:10.4204/EPTCS.194.6.

[105] Karl Lermer, Colin J. Fidge & Ian J. Hayes (2005): *A theory for execution-time derivation in real-time programs*. *Theoretical Computer Science* 346(1), pp. 3–27, doi:10.1016/j.tcs.2005.08.003. Quantitative Aspects of Programming Languages (QAPL 2004).

[106] Alberto Lluch-Lafuente & Ugo Montanari (2005): *Quantitative mu-calculus and CTL defined over constraint semirings*. *Theor. Comput. Sci.* 346(1), pp. 135–160, doi:10.1016/j.tcs.2005.08.006. Quantitative Aspects of Programming Languages.

[107] Ludovica Luisa Vissat, Jane Hillston, Glenn Marion & Matthew J. Smith (2016): *MELA: Modelling in Ecology with Location Attributes*. In: Proceedings 14th Workshop *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 227, pp. 82–97, doi:10.4204/EPTCS.227.6.

[108] Fabio Martinelli, Ilaria Matteucci & Francesco Santini (2015): *Semiring-based Specification Approaches for Quantitative Security*. In: Proceedings Thirteenth Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 194, pp. 95–109, doi:10.4204/EPTCS.194.7.

[109] Annabelle McIver & Carroll Morgan (2006): *A Novel Stochastic Game Via the Quantitative -calculus*. *Electronic Notes in Theoretical Computer Science* 153(2), pp. 195–212, doi:10.1016/j.entcs.2005.10.039. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[110] Annabelle McIver, Tahiry Rabehaja & Georg Struth (2013): *Probabilistic Concurrent Kleene Algebra*. In: Proceedings 11th Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 117, pp. 97–115, doi:10.4204/EPTCS.117.7.

[111] Marino Miculan & Marco Peressotti (2016): *Structural operational semantics for nondeterministic processes with quantitative aspects*. Theoretical Computer Science 655, pp. 135–154, doi:10.1016/j.tcs.2016.01.012. Quantitative Aspects of Programming Languages and Systems (2013-14).

[112] Bruno Montalto & Carlos Caleiro (2009): *Modeling and Reasoning about an Attacker with Cryptanalytical Capabilities*. Electronic Notes in Theoretical Computer Science 253(3), pp. 143–165, doi:10.1016/j.entcs.2009.10.010. Proceedings of Seventh Workshop on Quantitative Aspects of Programming Languages (QAPL 2009).

[113] Chunyan Mu & David Clark (2009): *An Interval-based Abstraction for Quantifying Information Flow*. Electronic Notes in Theoretical Computer Science 253(3), pp. 119–141, doi:10.1016/j.entcs.2009.10.009. Proceedings of Seventh Workshop on Quantitative Aspects of Programming Languages (QAPL 2009).

[114] Ukachukwu Ndukwu & Annabelle McIver (2010): *An expectation transformer approach to predicate abstraction and data independence for probabilistic programs*. In: Proceedings Eighth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 28, pp. 129–143, doi:10.4204/EPTCS.28.9.

[115] Tri Minh Ngo & Marieke Huisman (2013): *Quantitative Security Analysis for Multi-threaded Programs*. In: Proceedings 11th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 117, pp. 34–48, doi:10.4204/EPTCS.117.3.

[116] Anna Philippou, Insup Lee & Oleg Sokolsky (2012): *PADS: An approach to modeling resource demand and supply for the formal analysis of hierarchical scheduling*. Theoretical Computer Science 413(1), pp. 2–20, doi:10.1016/j.tcs.2011.08.025. Quantitative Aspects of Programming Languages (QAPL 2010).

[117] Sylvain Pradalier & Catuscia Palamidessi (2006): *Expressiveness of Probabilistic pi-calculus*. Electronic Notes in Theoretical Computer Science 164(3), pp. 119–136, doi:10.1016/j.entcs.2006.07.015. Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages (QAPL 2006).

[118] Markus Rabe & Sven Schewe (2010): *Optimal Time-Abstract Schedulers for CTMDPs and Markov Games*. In: Proceedings Eighth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 28, pp. 144–158, doi:10.4204/EPTCS.28.10.

[119] Aviv Regev, Ekaterina M. Panina, William Silverman, Luca Cardelli & Ehud Shapiro (2004): *BioAmbients: an abstraction for biological compartments*. Theor. Comput. Sci. 325(1), pp. 141–167, doi:10.1016/j.tcs.2004.03.061.

[120] Anne Remke, Boudewijn R. Haverkort & Lucia Cloth (2007): *CSL model checking algorithms for QBDs*. Theoretical Computer Science 382(1), pp. 24–41, doi:10.1016/j.tcs.2007.05.007. Quantitative Aspects of Programming Languages.

[121] Mads Rosendahl & Maja H. Kirkeby (2015): *Probabilistic Output Analysis by Program Manipulation*. In: Proceedings Thirteenth Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 194, pp. 110–124, doi:10.4204/EPTCS.194.8.

[122] Michal Rutkowski (2011): *Two-Player Reachability-Price Games on Single-Clock Timed Automata*. In: Proceedings Ninth Workshop on Quantitative Aspects of Programming Languages, Electronic Proceedings in Theoretical Computer Science 57, pp. 31–46, doi:10.4204/EPTCS.57.3.

[123] Andrei Sabelfeld & David Sands (2000): *Probabilistic Noninterference for Multi-Threaded Programs*. In: 13th IEEE Workshop on Computer Security Foundations, IEEE Computer Society, pp. 200–214, doi:10.1109/CSFW.2000.856937.

[124] Vijay A. Saraswat & Martin Rinard (1990): *Concurrent Constraint Programming*. In: Proc. of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'90, ACM, pp. 232–245, doi:10.1145/96709.96733.

[125] Stefan Schupp, Johanna Nellen & Erika Ábrahám (2017): *Divide and Conquer: Variable Set Separation in Hybrid Systems Reachability Analysis*. In: Proceedings 15th Workshop on Quantitative Aspects of Programming Languages and Systems, Electronic Proceedings in Theoretical Computer Science 250, pp. 1–14, doi:10.4204/EPTCS.250.1.

[126] Viktor Schuppan (2016): *Enhancing unsatisfiable cores for LTL with information on temporal relevance*. *Theoretical Computer Science* 655, pp. 155–192, doi:10.1016/j.tcs.2016.01.014. Quantitative Aspects of Programming Languages and Systems (2013-14).

[127] Ehud Shapiro (1992): *Embeddings among concurrent programming languages*. In W.R. Cleaveland, editor: *CONCUR'92*, *LNCS* 630, Springer, pp. 486–503, doi:10.1007/BFb0084811.

[128] Igor Siveroni (2006): *Filling Out the Gaps: A Padding Algorithm for Transforming Out Timing Leaks*. *Electronic Notes in Theoretical Computer Science* 153(2), pp. 241–257, doi:10.1016/j.entcs.2005.10.041. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[129] Michael J.A. Smith (2007): *Stochastic Modelling of Communication Protocols from Source Code*. *Electronic Notes in Theoretical Computer Science* 190(3), pp. 129–145, doi:10.1016/j.entcs.2007.08.013. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[130] Michael J.A. Smith (2008): *Probabilistic Abstract Interpretation of Imperative Programs using Truncated Normal Distributions*. *Electronic Notes in Theoretical Computer Science* 220(3), pp. 43–59, doi:10.1016/j.entcs.2008.11.018. Proceedings of the Sixth Workshop on Quantitative Aspects of Programming Languages (QAPL 2008).

[131] Lei Song, Flemming Nielson & Bo Friis Nielsen (2011): *A Stochastic Broadcast Pi-Calculus*. In: Proceedings Ninth Workshop on *Quantitative Aspects of Programming Languages*, *Electronic Proceedings in Theoretical Computer Science* 57, pp. 74–88, doi:10.4204/EPTCS.57.6.

[132] Pascal Sotin, David Cachera & Thomas Jensen (2006): *Quantitative Static Analysis Over Semirings: Analysing Cache Behaviour for Java Card*. *Electronic Notes in Theoretical Computer Science* 164(3), pp. 153–167, doi:10.1016/j.entcs.2006.07.017. Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages (QAPL 2006).

[133] David Spieler, Ernst Moritz Hahn & Lijun Zhang (2014): *Model Checking CSL for Markov Population Models*. In: Proceedings Twelfth Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 154, pp. 93–107, doi:10.4204/EPTCS.154.7.

[134] Anton Stefanek, Richard Hayden & Jeremy Bradley (2010): *A new tool for the performance analysis of massively parallel computer systems*. In: Proceedings Eighth Workshop on *Quantitative Aspects of Programming Languages*, *Electronic Proceedings in Theoretical Computer Science* 28, pp. 159–181, doi:10.4204/EPTCS.28.11.

[135] Mark Timmer, Joost-Pieter Katoen, Jaco van de Pol & Marielle Stoelinga (2016): *Confluence reduction for Markov automata*. *Theoretical Computer Science* 655, pp. 193–219, doi:10.1016/j.tcs.2016.01.017. Quantitative Aspects of Programming Languages and Systems (2013-14).

[136] Mathieu Tracol, Josée Desharnais & Abir Zhioua (2011): *Computing Distances between Probabilistic Automata*. In: Proceedings Ninth Workshop on *Quantitative Aspects of Programming Languages*, *Electronic Proceedings in Theoretical Computer Science* 57, pp. 148–162, doi:10.4204/EPTCS.57.11.

[137] Paolo Tranquilli (2013): *Indexed Labels for Loop Iteration Dependent Costs*. In: Proceedings 11th Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 117, pp. 19–33, doi:10.4204/EPTCS.117.2.

[138] Max Tschaikowski & Mirco Tribastone (2014): *Exact fluid lumpability in Markovian process algebra*. *Theoretical Computer Science* 538, pp. 140–166, doi:10.1016/j.tcs.2013.07.029. Quantitative Aspects of Programming Languages and Systems (2011-12).

[139] Max Tschaikowski & Mirco Tribastone (2014): *Extended Differential Aggregations in Process Algebra for Performance and Biology*. In: Proceedings Twelfth Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 154, pp. 34–47, doi:10.4204/EPTCS.154.3.

[140] Evan Tzanis & Robin Hirsch (2008): *Probabilistic Logic over Paths*. *Electronic Notes in Theoretical Computer Science* 220(3), pp. 79–96, doi:10.1016/j.entcs.2008.11.020. Proceedings of the Sixth Workshop on Quantitative Aspects of Programming Languages (QAPL 2008).

[141] Andrea Vandin (2017): *Language-based Abstractions for Dynamical Systems*. In: Proceedings 15th Workshop on *Quantitative Aspects of Programming Languages and Systems*, *Electronic Proceedings in Theoretical Computer Science* 250, pp. 15–24, doi:10.4204/EPTCS.250.2.

[142] Daniele Varacca & Nobuko Yoshida (2007): *Probabilistic Pi-Calculus and Event Structures*. *Electronic Notes in Theoretical Computer Science* 190(3), pp. 147–166, doi:10.1016/j.entcs.2007.07.009. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[143] Maria Grazia Vigliotti & Peter G. Harrison (2006): *Stochastic Ambient Calculus*. *Electronic Notes in Theoretical Computer Science* 164(3), pp. 169–186, doi:10.1016/j.entcs.2006.07.018. Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages (QAPL 2006).

[144] Dennis M. Volpano & Geoffrey Smith (1999): *Probabilistic Noninterference in a Concurrent Language*. *Journal of Computer Security* 7(1), pp. 231–253, doi:10.3233/JCS-1999-72-305.

[145] Verena Wolf, Christel Baier & Mila Majster-Cederbaum (2006): *Trace Machines for Observing Continuous-Time Markov Chains*. *Electronic Notes in Theoretical Computer Science* 153(2), pp. 259–277, doi:10.1016/j.entcs.2005.10.042. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005).

[146] Verena Wolf, Christel Baier & Mila Majster-Cederbaum (2006): *Trace Semantics for Stochastic Systems with Nondeterminism*. *Electronic Notes in Theoretical Computer Science* 164(3), pp. 187–204, doi:10.1016/j.entcs.2006.07.019. Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages (QAPL 2006).

[147] Gagarine Yaikhom, Murray Cole, Stephen Gilmore & Jane Hillston (2007): *A Structural Approach for Modelling Performance of Systems Using Skeletons*. *Electronic Notes in Theoretical Computer Science* 190(3), pp. 167–183, doi:10.1016/j.entcs.2007.07.010. Proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007).

[148] Hirotoshi Yasuoka & Tachio Terauchi (2014): *Quantitative information flow as safety and liveness hyperproperties*. *Theoretical Computer Science* 538, pp. 167–182, doi:10.1016/j.tcs.2013.07.031. Quantitative Aspects of Programming Languages and Systems (2011-12).

[149] Jinjin Zhang & Zhaohui Zhu (2008): *A Behavioural Pseudometric based on Bisimilarity*. *Electronic Notes in Theoretical Computer Science* 220(3), pp. 115–127, doi:10.1016/j.entcs.2008.11.022. Proceedings of the Sixth Workshop on Quantitative Aspects of Programming Languages (QAPL 2008).

[150] Miaomiao Zhang & Dang Van Hung (2006): *Formal Analysis of Streaming Downloading Protocol for System Upgrading*. *Electronic Notes in Theoretical Computer Science* 164(3), pp. 205–224, doi:10.1016/j.entcs.2006.07.020. Proceedings of the 4th Workshop on Quantitative Aspects of Programming Languages (QAPL 2006).