The Concurrency Column

by

Nobuko Yoshida

Department of Computing
Imperial College London
180 Queen's Gate, London, SW7 2AZ
n.yoshida@imperial.ac.uk, http://mrg.doc.ic.ac.uk/

Interviews with the 2021 CONCUR Test-of-Time Award Recipients

Luca Aceto
ICE-TCS, Department of Computer Science,
Reykjavik University
Gran Sasso Science Institute, L'Aquila

Nathalie Bertrand
Univ Rennes, Inria, CNRS, IRISA

Nobuko Yoshida
Department of Computing, Imperial College London, UK

Abstract

Last year, the CONCUR conference series inaugurated its Test-of-Time Award, purpose of which is to recognise important achievements in Concurrency Theory that were published at the CONCUR conference and that have stood the test of time. This year, the following four papers were chosen to receive the CONCUR Test-of-Time Awards for the periods 1994–1997 and 1996–1999 by a jury consisting of Rob van Glabbeek (chair), Luca de Alfaro, Nathalie Bertrand, Catuscia Palamidessi, and Nobuko Yoshida:

- David Janin and Igor Walukiewicz. On the Expressive Completeness of the Propositional mu-Calculus with respect to Monadic Second Order Logic [3].
- Uwe Nestmann and Benjamin C. Pierce. Decoding Choice Encodings [4].
- Ahmed Bouajjani, Javier Esparza, and the late Oded Maler. Reachability Analysis of Pushdown Automata: Application to Model-checking [2].
- Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. Alternating Refinement Relations [1].

This year, the second paper was live-interviewed by Nobuko Yoshida; the third paper was interviewed by Nathalie Bertrand and the forth paper was interviewed by Luca Aceto. Adam Barwell and Francisco Ferreira helped making the article from the live interview by Yoshida.

1 Interview with David Janin and Igor Walukiewicz

Luca: You receive the CONCUR ToT Award 2021 for your paper 'On the Expressive Completeness of the Propositional mu-Calculus with respect to Monadic Second Order Logic', which appeared at CONCUR 1996. In that article, you showed what I consider to be a fundamental result, namely that the mu-calculus and the bisimulation-invariant fragment of monadic second-order logic (MSOL) have the same expressiveness over transition systems. Could you tell us how you came to study that question and why do you think that such a natural problem hadn't been answered before your paper?

David and Igor: At that time we were interested in automata characterizations of the expressive power of MSOL and the mu-calculus. In 1988 Damian Niwinski has shown that over binary trees the mu-calculus and MSOL are expressively equivalent. When it appeared, the result was quite surprising, even unexpected. The two logics are not equivalent on unranked trees where the number of children of a node is not fixed. In consequence, the logics are not equivalent over the class of all transition systems.

In 1983 van Benthem showed that modal logic is equivalent to the bisimulation invariant fragment of first-order logic. When we have learned about this result we have realized that our automata models can be used to prove a similar statement for the mu-calculus.

The method of van Benthem could not be applied to MSOL, the automata based method looks like the only way to prove the result.

Luca: I am interested in how research collaborations start, as I like to tell 'research life stories' to PhD students and young researchers. Could you tell us how you started your collaboration?

David: I came to meet Igor when he was visiting Bordeaux in fall 93, presenting his first completeness result on a proof system for the mu-calculus (LICS 93). I was myself starting a PhD considering modal logic for specifying (various forms of) nondeterminism in connection with (power)domain theory (FSTTCS 93).

We eventually met again in Auvergnes (centre of France) for the Logic Colloquium in summer 94. There, spending time in the park nearby the conference venue or walking around the volcanoes, we elaborated a notion of nondeterministic automata for the modal mu-calculus. This result can be seen as extending the notion of 'disjunctive' normal form from propositional logic to the modal mu-calculus. I remember that Igor later used this result for his second completeness result for Kozen's proposed proof system for the mu-calculus (LICS 95). Thanks to Igor, this was for me the occasion to learn in depth the link between mu-calculus and tree automata.

Incidentally, Johan van Benthem was also attending this Logic Colloquium and we both attended his lecture about the bisimulation-invariant fragment of FO. Although we did not yet realize we could generalize his result to MSO, this surely increased our own understanding of logic.

Our first result (Automata for the mu-calculus) was presented at MFCS in 95 in Prague, where Igor and I met again. It could have been there, or a little later, that we eventually postulated our bisimulation-invariance result. However, proof arguments were only found later while exchanging emails. It was a bit amazing for me that we could discuss that way across Europe: email started to be heavily used only in the late 80's. In my head, Poland was far, far away from France.

Yet our collaboration was eventually in the line of an ongoing collaboration between Warsaw and Bordeaux, involving researcher like Arnold (my supervisor) and Niwinski (Igor's mentor), both major specialists in the area of automata and logics. Somehow, as a followup, together with Aachen (Grädel and Thomas) and many other sites, the GAMES European network was later created, and, almost at the same time, Igor came to Bordeaux as a CNRS researcher.

Luca: As you mentioned earlier, van Bentham has shown that modal logic has the same expressive power as the bisimulation-invariant fragment of first-order logic. In some sense, one may consider your main result as the extension of van Bentham's theorem to a setting with fixed-points. Could you briefly describe at a high level the challenges that fixed-points created in your w ork? To your mind, what was the main technical achievement or technique in your paper?

David and Igor: Sure our result is similar to van Benthem's, and, as mentioned above, his own presentation in 1993 was very inspiring. However, his proof relies on compactness of first-order logic and cannot be adapted to monadic second-order logic. In our proof, we have used automata-theoretic techniques.

In our previous works we had developed automata models for MSOL and the mu-calculus on unranked trees. Every transition system is bisimulation invariant to a tree obtained by the unfolding of the transition system. Crucially for our result, it is also equivalent to a tree where every subtree is duplicated infinitely many times. A short pumping argument shows that on such trees the two automata models are equivalent.

Luca: Did any of your subsequent research build explicitly on the results and the techniques you developed in your award-winning paper? Is there any result obtained by other researchers that builds on your work and that you like in particular?

David and Igor: Around that time Marco Hollenberg and Giovana D'Agostino used our automata-theoretic methods to show the uniform interpolation prop-

erty for the mu-calculus.

In collaboration with Giacomo Lenzi (postdoc in Bordeaux from Pisa), David considered the bisimulation-invariant fragment of various levels of the monadic quantifier alternation hierarchy. It turns out that monadic Σ_1 corresponds to reachability properties and monadic Σ_2 corresponds to Büchi properties, respectively the first and second levels of the mu-calculus hierarchy.

It could be the case that all mu-calculus can be translated into monadic Σ_3 formulas – this is true when restricting to deterministic transition systems. However, with nondeterminism, such a result seems difficult to achieve.

Incidentally, Giacomo and David also proved that adding limited counting capacity to modalities yields a fixed-point calculus equivalent to the unravelling invariant fragment of MSO (LICS 2001).

In 1999, in collaboration with Erich Grädel, Igor has used similar techniques to define and study guarded fixed-point logic. Subsequently, several other fixed-point logics of this kind were proposed with the most expressive one probably being guarded negation logic with fixed points of Bárány, ten Cate, and Segoufin from 2015. These works all use automata-theoretic methods to some extent.

Luca: Your paper was written while Igor was at BRICS at the University of Aarhus. Igor, what was it like to be at BRICS@Aarhus at that time? What role do you think centres like BRICS played and can play in the development of theoretical computer science? Do you think that your stay at BRICS had a positive impact on your future career?

Igor: My stay at BRICS had definitively a beneficial impact on me . At that time BRICS was one of the most active centers world wide in theoretical computer science. Being able to see and talk to so many different people, being exposed to so many different ideas, was very enriching. BRICS was a meeting place allowing scientists to have a better and larger vision of our field. BRICS contributed to the development of many people involved, as well as to the excellence of Aarhus, and even Denmark as a whole, in our field.

Luca: I have been brought up in the concurrency-theory tradition and I feel that bisimulation-invariant properties are the interesting ones over transition systems. Do you think that we actually 'need' logics that allow one to specify properties of transition systems that are not bisimulation invariant?

David: That was the idea indeed and the mu-calculus is the good logic for that. From a mathematical perspective, bisimulation is a fairly natural definition. In some sense, bisimulation equivalence is a greatest co-congruence.

However, I always suspected that bisimulation is too fine grained as an equivalence for concurrency. When modelling realistic systems, nondeterminism arises

either as controllable (angelic) or uncontrollable (demonic) choice. In this case, the right equivalence to be considered is simulation equivalence.

The funny thing is that David Park, who 'invented' bisimulation, was actually studying equivalence of J.R. Büchi's deterministic string automata where, because of determinism, bisimulation turns out to be equivalent to simulation equivalence. It is only after Matthew Hennessy and Robin Milner's work in concurrency that bisimulation was applied to nondeterministic behaviors.

Igor: In the context of XML and semistructured data, bisimulation is not relevant. One of the prominent queries in XPATH is whether a value appears twice – that is clearly not a bisimulation-invariant property. So while XPATH looks very close to PDL or the mu-calculus, many techniques and questions are very different. The other context that comes to mind is controller synthesis, where we ask for a transition system of a specific shape, for example, with self-loops on certain actions. Such self loops represent invisibility of the action to the controller.

Luca: What are the research topics that you find most interesting right now? Is there any specific problem in your current field of interest that you'd like to see solved?

David: In Logic for Computer Science, there have always been two kinds of approaches: model theory that eventually led to model checking and proof theory that eventually led to typed programming languages.

Twenty years later, aiming at designing and implementing real concurrent systems, especially for interactive arts, I realized that the latter approach was (at least for me) a lot more effective. Building concurrent systems by synchronizing arbitrary sub-systems sounded for me like unstructured programming; it was essentially unmaintainable. Monads and linear types, among many other approaches in typed functional programming, surely offered interesting alternatives to process calculi approaches.

Igor: In the context of the paper we discuss here, I am surprised by developments around the model-checking problem for the mu-calculus. After some years of relative calm, Calude, Jain, Khoussainov, Li and Stephan have made an important breakthrough in 2017. Yet, despite big activity after this result, the research on the problem seems to have hit one more barrier. Another old prominent problem is decidability of the alternation hierarchy for the mu-calculus. The problem is: given a formula and a number of alternations between the least and the greatest fixed-points, decide if there is an equivalent formula with this number of alternations. Even when the number of alternations is fixed we do not know the answer. Among others, Thomas Colcombet and Christof Loeding have done very interesting work on this subject.

Luca: What advice would you give to a young researcher who is keen to start working on topics related to automata theory and logic in computer science?

David: From the distance, our result sounds to me as a combination of both technique and imagination. Technique for mastering known results and imagination for finding original open problems, especially between research fields that are not (yet) known to be deeply related. As a matter of fact, I felt lucky finding such a fresh open problem that was (probably) a lot easier to solve than many other well-known hard problems.

Technique comes from hard work. It is obviously essential but somehow easy to teach and evaluate in academia. Imagination comes from curiosity. It is still essential but a lot more difficult to teach. So young researchers must develop by themselves their own imagination and curiosity.

In the automata-theory branch of logic in computer science, the remaining open problems seem fairly hard, so I believe that it is the imagination of young researchers that will make the difference for setting up new interesting directions of research, especially those who are ready to look aside, towards other areas of logic in computer science and, because it can be a considerable source of motivation, funny applications.

Igor: Naturally, the field is much broader these days than it was 25 years ago. It is crucial to master some techniques. For this, working on variations of already solved problems is a good method. Yet, I think it is important to escape the cycle of constant modifications of existing problems and their solutions. I would suggest that, at some moment, one should find an important open problem that one is passionate about and should spend a considerable effort on it. I admit that this is a matter of a taste, personality, and having a sufficiently comfortable situation to afford such a risk. Another good option is to look at frontiers with other areas: distributed computing, semantics, control theory, security.

2 Interview with Uwe Nestmann and Benjamin Pierce

Nobuko: You receive the CONCUR 2021 Test-of-Time Award for your paper "Decoding Choice Encodings", which appeared at CONCUR 1996. Could you tell us briefly what lead you to embark on studying the expressiveness of choice in the asynchronous pi-calculus?

Uwe: I did my diploma thesis in '91. I was working on a topic that had to do with communicating functions. I built a lambda calculus with communication in it. It was a typed lambda calculus and it was strong enough to type the Y-combinator. I went to Edinburgh and presented that at the Concurrency Club, and

they asked me, "*who is your supervisor?*". I didn't really have one, because I was mostly self-driven those days. What they told me is that it's better to not work on this topic without a competent supervisor. The second piece of advice was to get a supervisor first and then look for a t opic. So I found B enjamin. Benjamin had this wonderful project at the time on trying to make a programming language out of the pi-calculus and, in that language, choice encodings (or at least choice operators) played a role. He invited me to visit him in Paris, where he was on six-months' leave, I think.

Benjamin: It was actually a "nested postdoc." I did three postdocs after finishing my PhD at CMU: one in Edinburgh, one in Paris, and one in Cambridge. The Paris one was nested inside the Edinburgh one.

Uwe: I was in Paris for one week, and Benjamin told me to try programming in his new programming language, Pict. I tried to write down the dining philosophers problem, in a way such that I could use an abstraction to pick up forks in either order, and I wanted to instantiate it with left and right, and right and left, but the Pict language didn't allow me to do so. Behind was the interplay between choice and abstraction (and instantiation) and that was the start of all of it from my point of view. Then I wrote up an exposé and I ended up actually working on just a third of that for my PhD thesis. And of course, there were technical reasons for Benjamin and Dave [Turner] at the time for being interested in choice constructs.

Benjamin: Of course, Dave Turner is the most important name that needs to be mentioned here, besides obviously Robin Milner. All of this was happening under the umbrella of Robin's wonderful work on pi-calculus and the amazing group that he had assembled at the time. He had this incredible set of students, including Dave Turner and David Sangiorgi and Peter Sewell, doing all sorts of things with pi-calculus. Dave, besides being a first-class hacker, was also a really good theoretician. He truly married the two. He and I started talking at some point about what kind of programming language would you get if you treated the pi-calculus like the Lisp people treated the lambda calculus. What that led to was a lot of different language designs based on different versions of the pi-calculus, but we kept wanting to make it simpler and simpler. Partly because we were thinking of it as possibly even a distributed language, not just a concurrent language, and as everybody knows, the choice operator – in the full-blown pi-calculus or CCS sense – is not a real thing in distributed systems: it's not implementable. So we were trying to make the underlying calculus simpler and simpler, and eventually wound up with this programming language with no choice at all. But as Uwe discovered, there are things that you might want to do where choice is the natural primitive, like dining philosophers, which raises the question of how much of it can you get just by programming on top of plain parallel composition plus messages on channels. We found that programming up a restricted form of choice was a little tricky but not *that* tricky. What *was* really tricky, though, was justifying that it was

correct. The reason why it turned into a whole dissertation for Uwe, was because the well-known notions of correctness that were lying around did not apply to this situation. I remember being totally astonished at the length and technicality of the final proof that Uwe ended up doing.

Nobuko: Did you imagine at the time that your award-winning paper would have so much impact on the area of expressiveness in concurrency theory, and how do you feel now?

Benjamin: Maybe Uwe did; I did not. I think we were just following our noses.

Uwe: Actually, I would say "yes" and "no". The "no" is when it came to the CONCUR acceptance, I got the impression that we just about made it because the competition was so tough and the pi-calculus was really hot at that time. There were six or seven pi-calculus papers in the conference in the end (I don't know how many were in the submission pool). The tiny "yes" that I would like to say is because Kohei [Honda] had foreseen it. When I gave the presentation at the Newton Institute just in autumn '95 – that was the workshop that Benjamin organised on concurrent high-level languages – Kohei came to me after the talk and said something like, "*maybe you don't know yet, but you will be known for this*". I can't remember the exact wording. I think he called it "*Nestmann's Theorem*" or something. Me, a PhD student, the first time in front of this crowd of expert people and then he tells me something like that. I didn't believe him. Of course not.

Benjamin: Kohei was ahead of his time in so many ways.

Nobuko: Could you tell us what the research environment was like in Edinburgh and the UK as a whole at that time and how it has influenced the rest of your career?

Benjamin: I came as a postdoc to Robin's group. I was the last postdoc of Robin's in Edinburgh, and then travelled with him to Cambridge, where Peter Sewell and I were his first postdocs. I would say that both Edinburgh and Cambridge at the time, and still, were just incredible. In Edinburgh you had Milner, you had [Gordon] Plotkin, you had Don Sannella, you had students around you like Martin Hofmann and Philippa Gardner and Marcelo Fiore, and the list goes on and on. You had other postdocs like Randy Pollack. It was just an incredible place. People talking about amazing, deep, mind-bending things all the time. It was particularly an amazing place for thinking about concurrency. There were a lot people breaking new ground.

Nobuko: Benjamin, how did that experience influence your current research?

Benjamin: For one thing, it solidified my interest in language design. The whole Pict experience was so fruitful. It was so much fun working with Dave on

implementing this interesting language, and both the design and the programming that we did in PICT gave rise to so many interesting questions. For example, it led us to do a bunch of thinking about type systems for concurrency, and I can see echoes of those ideas in the work that you, Nobuko, and colleagues have done more recently with session types. Though I don't consider myself a core concurrency researcher any more, the experience gave me an appreciation for the theory of concurrency that has drawn me back to the area over and over.

Nobuko: Uwe, how did it influence your research?

Uwe: I did my PhD in Erlangen (University of Erlangen-Nürnberg), which was a place that was not so much known at that time for theory, and especially not for concurrency theory. I had the opportunity by a bilateral travel exchange programme between these two universities pushed by my other supervisor, Terry Stroup, at that time. And when I came into Edinburgh, not only was there so much competence around, which is mainly what Benjamin summarised, but there was so much openness. There was so much openness for any kind of ideas. So much curiosity and joy. So I was very lucky that I could regularly, every couple of months, visit the LFCS for a few days. There, I was pumped up with content and ideas, and did a presentation in the pi club in Robin's tiny office, with almost ten people sitting around a tiny blackboard, listening to my ideas and my problems. It was just unbelievable at this time. That kind of culture and atmosphere was so great. I traced it back, in May or June '95, since we're talking about this particular paper, it was culminating in the crucial part where I was just before proving choice encodings correct. I only needed two ingredients. One came a week later by Davide Sangiorgi posting, for the first time, a short note on asynchronous bisimilarity. And the other was that we were rediscovering, mostly together in the pi club with Ole-Høgh Jensen and Robin Milner, the notion of coupled similarity. Both Ole and Robin had different ideas and came to the same conclusion. I came back to Erlangen and found the old paper on coupled similarity by [Joachim] Parrow and [Peter] Sjödin and within a week all of the pieces were just about there. I "simply" had to write down the details and convince myself that it went all the way through. That was the crucial moment and without Edinburgh, without this culture, these possibilities, this openness, it would not have happened and maybe I would not even have become a professor in Berlin. Just because of this tiny situation and getting together of bright people.

Nobuko: Studying expressiveness this way was quite new and at the beginning at that time, so you probably cared a lot about presentation and how to communicate your ideas. Do you have any comments about this aspect? I found your paper is still very readable and very clearly written for such a subtle paper. How did you go about writing with this in mind? Apart from technical details.

Uwe: I was a great fan of Benjamin's presentation and communication skills at that time. I was seeing him on stage and reading his papers and I had the possibility to closely interact with this impressive guy and learn from him. Just recently, I learnt about a citation that summarises this approach about writing: *"Do not try to write such that you are understood. Try to write such that you cannot be misunderstood."* I think this expresses precisely what I think I learnt back then in trying to get this paper out, and making these subtle observations, and finding the right notation. It's often underestimated how important the role of good notation is for getting things across. The same goes for graphical presentations. And then, polishing, polishing, polishing, polishing. *"Get simpler sentences,"* Benjamin always said. I'm German, you know, we like complicated constructions which are somewhat nice and deeply nested. I learnt at the time to get it as simple as possible. Presentations were another thing. I found my presentation back at the Newton Institute again and I remember I had this table of contents written with **ABCDE**, which were the initial letters of the concepts that I presented: **A**synchrony, **B**isimulation, **C**oupled similarity, **D**ecoding, and I think E was for **E**nd or something. I obviously like playing with words, and I admire the power and joy of well-chosen language.

Nobuko: I do remember your presentation. You highlighted a coupled simuluation as a part of Rob [van Glabbeek]'s famous diagram at branching bisimilarities' CONCUR paper. I still remember your presentation at Newton Institute.

Benjamin: I have always cared a lot about good writing. Communicating ideas is really one of the most important parts of an academic's job. So it feels important to acknowledge the people I learned about writing from. The first was Don Knuth – his level of attention to writing, among all the other things he did, is totally inspiring to me. The other was John Reynolds, who was one of my two supervisors as a PhD student and who is the most careful writer that I've ever worked closely with. He gave me one time a draft of one of his papers to proofread, and I said to myself, *"Aha, this is my chance to get back at him for all the mistakes and flaws he has found in my writing over the years!"* So I started reading it, and I got more and more frustrated because I couldn't find *anything* to improve. Anything at all! In the whole paper – not a comma, not a notational choice, not the way something was worded. Nothing. That experience was both an inspiration and a humbling lesson to me.

The biggest thing I've learned over the years about writing is that the biggest ingredient of good writing is exactly what Uwe brought to this paper: the willingness to iterate until it's good. Good writers are people that stop polishing later than bad writers.

Nobuko: How much of your later work has built on your award-winning paper? What follow-up result of yours are you most proud of and why?

Uwe: I would like to mention three. Funnily, neither of them was in the decade following the paper. The reason may be because I was dragged into other projects, having to do with security protocols, pi-calculus, and object calculi. (1) By accident, I got back in contact with Ursula Goltz, who was one of my PhD referees: she was working on a project on synchronous and asynchronous systems and she asked me for literature because she knew I was digging deep in the 80s about results on the first CSP implementations. In the course of this project, I got back to actually directly building upon my PhD work, and I found Kirstin Peters, at that time a PhD student, who got interested in that. We found a number of remarkable observations having to do with distributed implementability and notions of distributability and what this may have to do with encodings between calculi. We discovered a hierarchy of calculi where you can very easily see which of them are at the same level of distributed implementability. We found that the asynchronous pi-calculus is actually not fully implementable in a distributed system, like many others. There is the ESOP paper in 2013, which I'm very proud of. Kirstin pushed this reasearch much further. (2) Another follow-up work concerns the notion of correctness that we were applying in the awarded paper, it was a lot about a direct comparison between terms and their translations. Not by plain full abstraction on two different levels and having an if and only if, but a direct translation so you could not distinguish a term from its translation. This kind of observation led to a rerun of, say, the research on what we actually want from an encoding. What is a good criterion for a good encoding? This culminated in the work with Daniele Gorla where we criticised the notion of full abstraction in the sense that it's a very important notation but you can easily misuse it and get to wrong results or useless results. (We also emphasised the importance of operational correspondence, and Daniele went on to establish his, by now, quite standard and established set of criteria for what a good encoding is.) That is a nice highly abstract paper with Daniele in Mathematical Structures in Computer Science in 2016, only, so also well, well after the CONCUR paper in 1996. (3) In just the last two or three years, my PhD student Benjamin Bisping finally studied algorithms and implementations for checking coupled similarity. We found an amazing wealth of new views on these kinds of equivalences that are slightly weaker than weak bisimilarity. So back to the roots, in a sense, to what we were doing 25 years ago. (Like Kirstin Peters and Rob van Glabbeek who further showed that coupled similarity is in fact very closely connected to encodings, in general.) Seeing these developments makes a lot of fun.

Nobuko: This was a TACAS paper, right?

Uwe: Yes, and we also published the survey article "Coupled Similarity – The First 32 Years", for the Festschrift for Robert van Glabbeek. It's basically an advertising paper for this great notion of equivalence, which is highly underestimated. It's, in a sense, much better than weak bisimilarity. Especially if you're

interested in – and this is my favourite domain – distribution, distributability, distributed implementations.

Nobuko: Benjamin, do you have any further comments?

Benjamin: For me, the answer is a little more oblique. I haven't written papers about choice encodings and things like that, besides this one. But what it did for me was to really solidify my interest in the asynchronous pi-calculus as a foundation for programming languages – and as a foundation for thinking about concurrency – because this paper, Uwe's result, teaches us that the asynchronous pi-calculus is more powerful than it looks – powerful enough to do a lot of programming in. You know there's this famous quote attributed to Einstein, *"Make everything as simple as possible, but no simpler."* I felt like the asynchronous pi-calculus was kind of "it" after seeing this result. And that calculus then became the foundation for a whole bunch of my later work on type systems for concurrency and language design.

Uwe: Actually the encodings we did back then went into what is now called the "localised asynchronous pi-calculus", but it simply wasn't yet known back then. The localised asynchronous pi-calculus is at this perfect level of distributed implementability, as we know by now.

Nobuko: This is partly also Massimo Merro did with Davide Sangiorgi, right?

Uwe: Yes, they did this few years later, towards the end of the '90s.

Nobuko: What do you think of the current state and future directions of the study of expressiveness in process calculi, or more generally, concurrency theory as a whole?

Uwe: Back then, in Cambridge, I was having discussions with Peter Sewell. Quite many of them. At the time, we were making fun by saying, *"now we know how to do process calculi, we can do five of them for b reakfast."* We know the techniques, we know how to write down the rules, we know what to look for in order to make it good. And I would say that for studying encodings nowadays it's kind of the same level of maturity; we know what to look for when writing down encodings, pitfalls to avoid, and it's done. So what I found most interesting today, is that often enough, the proximity between encodings and actually doing implementations is very close and that is may be because the maturity of programming languages we can use is much higher. We can use convenient abstractions in order to more-or-less straightforwardly write down encodings.

What's going on? Current state and future directions. The EXPRESS/SOS workshop is still alive and kicking. It attracts great papers and not that many are submitted but typically they're great papers and I think that's good. I think we had an impact on concurrent programming, and for example, if you look at Google Go, the concurrency primitives that you find in there is pretty much a process

calculus. It's message passing, and choice, and even mixed choice, and stuff like that. I cannot say right now that there are deep, deep, deep questions to be solved about encodings except for finding out what Robert van Glabbeek's criteria have to do with Daniele Gorla's criteria. There is an ongoing debate, but the issues are quite technical. What could use more research is typed languages, typed calculi, and typed encodings. It has been done and we have many nice results, but I think there are still some open questions on what the ideal criteria should be on those.

Nobuko: What advice would you give a young researcher interested in working on concurrency theory and process calculus like today?

Benjamin: My best advice for people that want to do theory is: keep one foot in practice. Don't stop building things. Because that's the way you find interesting problems, it's the way you keep yourself grounded, it's the way you make sure that the directions you're looking and the questions that you're asking have something to do with ... something! It's the way to stay connected to reality while also generating great questions.

Nobuko: Uwe, do you have anything to add to that?

Uwe: Having a foot in practice is also good for actually checking and finding mistakes in your reasoning. Building systems not only for finding problems but also for finding out that you have a problem in your thinking. Apart from that, I would not like to push for any particular area for concurrency theory. I mean, concurrency theory is incredibly wide. My advice is: get the best possible supervisor that you can find and then work on his project. I think this is very good advice. Be patient, dig deep. This is very general advice. Never give up. It took me two years until, in one week, the pieces fell together. So be patient, dig deep, and train your communication skills, practice networking. All the general things. Ah, and maybe what I found very useful for my own career: learn the basics and the history of your field. Understand what has already been found and what that means even twenty years after. I learned a lot from the early 80s papers that I was mentioning beforehand on first implementations of the communication primitives of CSP. There is one published supposedly deadlock-free algorithm, which almost twenty years later was discovered to be incorrect. The proof was incorrect; it was not actually deadlock free. So, work on hard problems, dig deep, be patient. And communicate well. This is also the best way to get help.

Nobuko: Wow. Anyone who can satisfy everything would be quite a fantastic student. (Laughs.) Like you, Uwe, you know.

Nobuko: This is the last question: what are the research topics that currently excite you the most? Can I ask Benjamin first?

Benjamin: I will name two. One is machine-checked proofs about real software. Over the past fifteen or twenty years, the capabilities of proof assistants,

and the community around them, have reached the point where you can really use them to verify interesting properties of real software; this is an amazing opportunity that we are just beginning to exploit.

On a more pragmatic level, I'm very interested lately in testing. Specifically, specification-based (or property-based) testing in the style popularised by QuickCheck. It's a beautiful compromise between rigor and accessibility. Compared to the effort of fully verifying a mathematically stated property, it is incredibly easier and lower-cost, and yet, you can get tremendous benefit, both from the process of thinking about the specification in the mathematical way that we're used to in this community and from the process of testing against, for example, randomly generated or enumerated examples. It's a sweet spot in the space of approaches to software quality.

Nobuko: These things are still very difficult for concurrency or distributed systems. Do you have any comment because proof assistants for concurrency theory is, I think, still quite difficult compared to hand-written proof.

Benjamin: Yes, in both domains – both verification and testing – concurrency is still hard. I don't have a deep insight into why it is hard in the verification domain, beyond the obvious difficulty that the properties you want are subtle; but in the testing domain, the reason is clear: the properties have too many quantifier alternations, which is hard for testing. Not impossible – not always impossible, anyway – but it raises hard challenges.

Uwe: There's a recurring pattern in what I like doing and that is always to do with looking at different levels of abstractions. You can think of it in terms of encodings or as a distributed system, and I was always wondering about the relation between global (higher-level) properties and local (lower-level) implementation of systems. And throwing formal methods, formal models, and theories at this problem has always been what I liked, and I still do that, nowadays again, more on fault-tolerant distributed algorithms. Maybe also because of the recent hype due to blockchain and the strong interest in practical fault-tolerant Byzantine algorithms, and so on. And, here I meet Benjmain again, at best doing mechanical verification of those. Mechanical verification is still hard and you can easily pull PhD students into a miserable state by dragging them onto a problem that takes an awful lot of time, and then you get out one paper, with the proof in Isabelle, in our case. On the other hand, it's getting more and more a tool that we just use. The more you've done, using a proof assistant, the more you integrate it into your everyday life. Some students, as a standard, test their definitions and their theorems and do their proofs in Isabelle and we now even have bachelor students using that. Good ones, I mean bright ones, of course, but it's becoming more and more an everyday thing. The other idea: Benjamin, you're well-known also for the software foundations series. I don't know whether you've done pedagogical research, learning theory, on top of that in the following sense. What we are inter-

ested in, just recently, is understanding how people learn how to do proofs. It's a long, difficult, mental process and there are a number of theories about this actually works, and whether this works, and there's magic involved, and whatnot. And getting used to, of course. Learning from patterns. But then, what could be the impact of using proof assistants for learning how to do proofs? Does it actually help? Or does it actually hinder?

Benjamin: It turns people into hackers. (Laughs.)

Uwe: Yeah, yeah, yeah. We're talking about computer science students, not maths students, right? Programming is proving, proving is programming. This is of course a slogan from type theory, but one may actually use it as a motivation to write down first proofs, getting feedback from the proof assistant, and go on from there. This is one of the interests that we have, in actually understanding this process of learning how to do proofs.

Nobuko: I now conclude this interview. Thank you both very much for giving us your time.

3 Interview with Ahmed Bouajjani and Javier Esparza

Nathalie: You receive the CONCUR ToT Award 2021 for your paper with Oded Maler *Reachability Analysis of Pushdown Automata: Application to Model-Checking*, which appeared at CONCUR in 1997. In that article, you develop symbolic techniques to represent and manipulate sets of configurations of pushdown automata, or even of the broader class of alternating pushdown systems. The data structure you define to represent potentially infinite sets of configurations is coined alternating multi-automata, and you provide algorithms to compute the set of predecessors (pre*) of a given set of configurations. Could you briefly explain to our readers what alternating multi-automata are?

Ahmed: The paper is based on two ideas. The first one is to use finite automata as a data structure to represent infinite sets of configurations of the pushdown automaton. We called them multi-automata because they have multiple initial states, one per control state of the pushdown automaton, but there is nothing deep there. The second idea is that this representation is closed under the operation of computing predecessors, immediate or not. So, given a multi-automaton representing a set of configurations, we can compute another multi-automaton representing all their predecessors. If you compute first the immediate predecessors, then their immediate predecessors, and so on, you don't converge, because your automata grow bigger and bigger. The surprising fact is that you can compute all predecessors in one go by just adding transitions to the original automaton, without

adding any new states. This guarantees termination. Later we called this process "saturation".

Once you can compute predecessors, it is not too difficult to obtain a model-checking algorithm for LTL model checking. But for about branching-time logics you must also be able to compute intersections of sets of configurations. That's where alternation kicks in, we use it to represent intersections without having to add new states.

Nathalie: Could you also tell us how you came to study the question addressed in your award-winning article? Which of the results in your paper did you find most surprising or challenging?

Javier: In the late 80s and early 90s many people were working on symbolic model-checking, the idea of using data structures to compactly represent sets of configurations. BBDs for finite-state model-checking were a hot topic, and for quite a few years dominated CAV. BDDs can be seen as acyclic automata, and so it was natural to investigate general finite automata as data structure for infinite-state systems. Pierre Wolper and his group also did very good work on that.

About your second question, when I joined the team Ahmed and Oded had already been working on the topic for a while, and they had already developed the saturation algorithm. When they showed it to me I was blown away, it was so beautiful. A big surprise.

Nathalie: In contrast to most previous work, your approach applied to model checking of pushdown systems treats in a uniform way linear-time and branching time logics. Did you apply this objective in other contributions?

Javier: I didn't. The reason is that I was interested in concurrency, and when you bring together concurrency and procedures even tiny fragments of branching-time logics become undecidable. So I kind of stuck to the linear-time case. Did you work on branching-time, Ahmed?

Ahmed: Somehow yes (although it is not precisely about linear vs branching time properties), in the context of Regular Model Checking, a uniform framework for symbolic analysis of infinite-state systems using automata-based data structures. There, we worked on two versions, one based on word automata for systems where configurations can be encoded as words or vectors of words, such as stacks, queues, etc., and another one based on tree automata for configurations of a larger class of systems like heap manipulating programs, parametrised systems with tree-like architectures, etc. The techniques we developed for both cases are based on the same principles.

Nathalie: As it is often the case, the paper leaves some open questions. For instance, I believe, the precise complexity of verification of pushdown systems against CTL specifications is PSPACE-hard and in EXPTIME. Did you or others close this gap since? Did your techniques help to establish the precise complexity?

Ahmed: In our paper we showed that model checking the alternating modal

mu-calculus is EXPTIME-hard. CTL is less expressive, and it was the most popular logic in the verification community at the time, so it was natural to ask if it had lower complexity.

Javier: Yes, as a first step in the paper we showed that a fragment of CTL, called EF, had PSPACE complexity. But I made a mistake in the proof, which was later found by Igor Walukiewicz. Igor cracked the problem in a paper at FSTTCS'00. It turns out that EF is indeed PSPACE-complete (so at least we got the result right!), and full CTL is EXPTIME-complete. I wish Igor had used our technique, but he didn't, he applied the ideas of his beautiful CAV'96 paper on parity pushdown games.

Nathalie: It is often interesting to understand how research collaborations start as it can be inspiring to PhD students or colleagues. Could you tell us how you started your collaboration on the award-winning paper? Did you continue working together (on a similar topic or on something totally different) after 1997?

Ahmed: Javier and I first met in Liege for CAV 9 5. French universities have this program that allows us to bring foreign colleagues to France for a month as invited professors, and I invited Javier to Grenoble in 96.

Javier: It was great fun; Verimag was a fantastic place to do verification, we both liked cinema, Ahmed knew all restaurants, and the Alps were beautiful. Ahmed invited me again to Grenoble in 97. This time I came with my wife, and we again had a great time.

When I arrived in Grenoble in 96 Ahmed and Oded had already written most of the work that went into the paper. My contribution was not big, I only extended the result to the alternation-free mu-calculus, which was easy, and proved a matching lower bound. I think that my main contribution came *after* this paper. Ahmed and Oded were too modest, they thought the result was not so important, but I found it not only beautiful, I thought it'd be great implement the LTL part, and build a model checker for programs with procedures. We could do that thanks to Stefan Schwoon, who started his PhD in Munich around this time–he is now at Paris-Saclay—and was as good a theoretician as a tool builder. Around 2000 he implemented a symbolic version of the algorithms in MOPED, which was quite successful.

Ahmed: In 99 I moved to LIAFA in Paris, and I remember your kids were born.

Javier: Yes, you sent my wife beautiful flowers!

Ahmed: But we kept in touch, and we wrote a paper together in POPL'03 with my PhD student Tayssir Touili, now professor in Paris. We extended the ideas of the CONCUR paper to programs with both procedures and concurrency. Other papers came, the last in 2008.

Javier: And Ahmed is visiting Munich next year, pandemic permitting, so I hope there'll be more.

Nathalie: How would you say this award-winning paper influenced your later work? Did any of your subsequent research build explicitly on it?

Ahmed: This paper was the first of many I have co-authored on verification of infinite-state systems using automata. All of them use various automata classes to represent sets of configurations, and compute reachable configurations by iterative application of automata operations. We call these procedures accelerations; instead of computing a fixpoint of a function by repeated iteration, you "jump" to the fixpoint after finitely many steps, or at least converge faster. Accelerations were implicitly present in the CONCUR'97 paper. They have been also used by many other authors, for example Bernard Boigelot and Pierre Wolper.

My first paper on accelerations was with Peter Habermehl, my PhD student at the time and now at IRIF. We worked on the verification of systems communicating through queues, using finite automata with Presburger constraints as data structure. Then came several works on communicating systems with my student Aurore Annichini and Parosh Abdulla and Bengt Jonsson from Uppsala. As a natural continuation, with the Uppsala group and my student Tayssir Touili we developed the framework of Regular Model Checking. And then, with Peter Habermehl, Tomas Vojnar and Adam Rogalewicz from TU Brno, we extended Regular Model Checking to Abstract Regular Model Checking, which proved suitable and quite effective for the analysis of heap manipulating programs.

We also applied the CONCUR'97 results to the analysis of concurrent programs. The first work was a POPL'03 paper with Javier, Tayssir, and me on an abstraction framework. Two years later, Shaz Qadeer and Jacob Rehof proposed bounded-context switch analysis for bug detection. That paper created a line of research, and we contributed to it in many ways, together with Shaz, Mohamed Faouzi Atig, who was my student then, and is now Professor at Uppsala, and others.

Javier: The CONCUR'97 paper was very important for my career. As I said before, it directly led to MOPED, and later to jMOPED, a version of MOPED for Java programs developed by Stefan Schwoon and Dejvuth Suwimonteerabuth. Then, Tony Kucera, Richard Mayr, and I asked ourselves if it was possible to extend probabilistic verification to pushdown systems, and wrote some papers on the topic, the first in LICS'04. This was just the right moment, because at the same time Kousha Etessami and Mihalis Yannakakis started to write brilliant papers on recursive Markov chains, an equivalent model. The POPL'03 paper with Ahmed and Tayssir also came, and it triggered my work on Newtonian program analysis with two very talented PhD students, Stefan Kiefer, now in Oxford, and Michael Luttenberger, nw my colleague at TUM. So the CONCUR'97 paper was at the root of a large part of my work of the next 15 years.

Nathalie: Is there any result obtained by other researchers that builds on your work and that you like in particular or found surprising?

Javier: After implementing MOPED, Stefan worked with Tom Reps on an extension to weighted pushdown automata, the Weighted Pushdown Library. Tom and Somesh Jha also found beautiful applications to security. This was great work. I was also very impressed by the work of Luke Ong and his student Matthew Hague. In 97 Ahmed and I tried to apply the saturation method to the full mu-calculus but failed, we thought it couldn't be done. But first Thierry Cachat gave a saturation algorithm for Büchi pushdown games, then Luke, Matthew cracked the mu-calculus problem, and then they even extended it to higher-order pushdown automata, together with Arnaud Carayol, Oliver Serre, and others. That was really surprising.

Ahmed: I agree. I'd also mention Qadeer and Rehof's TACAS'05 paper. They built on our results to prove that context-bounded analysis of concurrent programs is decidable. They initiated a whole line of research.

Nathalie: What are the research topics that you find most interesting right now? Is there any specific problem in your current field of interest that you'd like to see solved?

Javier: Ten years ago I've had said the complexity of the reachability problem for Petri nets and of solving parity games, but now the first one is solved and the second almost solved! Now I don't have a specific problem, but in the last years I've been working on parameterised systems with an arbitrary number of agents, and many aspects of the theory are still very unsatisfactory. Automatically proving a mutual exclusion algorithm correct for a few processes was already routine 20 years ago, but proving it for an arbitrary number is still very much an open problem.

Ahmed: I think that invariant and procedure summary synthesis will remain hard and challenging problems that we need to investigate with new approaches and techniques. It is hard to discover the right level of abstraction at which the invariant must be expressed, which parts of the state are involved and how they are related. Of course the problem is unsolvable in general but finding good methodologies on how to tackle it depending on the class of programs is an important issue. I think that the recent emergence of data-driven approaches is promising. The problem is to develop well principled methods combining data-driven techniques and formal analysis that are efficient and that offer well understood guarantees.

Nathalie: Would you have an anecdote or a tip from a well-established researcher to share to PhD students and young researchers?

Javier: Getting this award reminded me of the conference dinner at CAV 12 in St. Petersburg. I ended up at a table with some young people I didn't know. The acoustics was pretty bad. When the CAV Award was being announced, somebody at the table asked "What's going on?", and somebody else answered "Not much, some senior guys getting some award". Never take yourself very seriously ...

Nathalie: Oded Maler passed away almost 3 years ago. Do you have any

memory of him to share with our readers?

Ahmed: Oded was very amused by the number of citations. He used to say "Look at all the damage we've done".

Javier: Yes, Oded had a wonderful sense of humour, very dry and deadpan. When I arrived in Grenoble it took me a few days to learn how to handle it! I miss it very much.

4 Interview with Rajeev Alur, Thomas A. Henzinger, Orna Kupferman and Moshe Y. Vardi

Luca: You receive the CONCUR 2021 Test-of-Time Award for your paper "Alternating Refinement Relations", which appeared at CONCUR 1998. In that article, you gave what I consider to be a fundamental contribution, namely the introduction of refinement relations for alternating transition systems. Could you briefly explain to our readers what alternating transition systems are? Could you also tell us how you came to study the question addressed in your award-winning article and why you focused on simulation- and trace-based refinement relations? Which of the results in your paper did you find most surprising or challenging?

AHKV: When we model a system by a graph, our model abstracts away some details of the system. In particular, even when systems are deterministic, states in the model may have several successors. The nondeterminism introduced in the model often corresponds to different actions taken by the system when it responds to different inputs from its environment. Indeed, a transition in a graph that models a composite system corresponds to a step of the system that may involve some components. Alternating transition systems (ATSs) enable us to model composite systems in more detail. In an ATS, each transition corresponds to a possible move in a game between the components, which are called agents. In each move of the game, all agents choose actions, and the successor state is deterministically determined by all actions. Consequently, ATSs can distinguish between collaborative and adversarial relationships among components in a composite system. For example, the environment is typically viewed adversarially, meaning that a component may be required to meet its specification no matter how the environment behaves.

In an earlier paper[1], some of us introduced ATSs and Alternating Temporal Logics, which can specify properties of agents in a composite system. The CONCUR 1998 paper provided refinement relations between ATSs which correspond to alternating temporal logics. Refinement is a central issue in a formal approach

[1] See https://www.cis.upenn.edu/~alur/Jacm02.pdf.

to the design and analysis of reactive systems. The relation "I refines S" intuitively means that system S has more behaviors than system I. It is useful to think about S being a specification and I an implementation. Now, if we consider a composite implementation $I||E$ and specification $S||E$ and we want to check that the component I refines the component S, then the traditional refinement preorders are inappropriate, as they allow I to achieve refinement of $I||E$ with respect to $S||E$ by constraining its environment E. Alternating refinement relations are defined with respect to ATSs that model the interaction among the underlying components, and they enable us to check, for example, that component I has fewer behaviors than component S no matter how component E behaves. They are called "alternating" because refinement may restrict implementation actions but must not restrict environment actions. In other words, refinement may admit fewer system actions but, at the same time, more environment actions.

It was nice to see how theoretical properties of preorders in the traditional setting are carried over to the game setting, and so are the results known then about the computational price of moving to a game setting. First, the efficiency of the local preorder of simulation with respect to the global preorder of trace containment is maintained. As in the traditional setting, alternating simulation can be checked in polynomial time, whereas alternating trace-containment is much more complex. Second, the branching vs. linear characterizations of the two preorders is preserved: alternating simulation implies alternating trace containment, and the logical characterization of simulation and trace-containment by CTL and LTL, respectively, is carried over to their alternating temporal logics counterparts. The doubly-exponential complexity of alternating trace containment, as opposed to the PSPACE complexity of trace containment, is nicely related to the doubly-exponential complexity of LTL synthesis, as opposed to its PSPACE model-checking complexity.

Luca: In your paper, you give logical characterisations of your alternating refinement relations in terms of fragments of alternating temporal logic. Logical characterisations of refinement relations are classic results in our field and I find them very satisfying. Since I teach a number of those results in my courses, I'd be interested in hearing how you would motivate their interest and usefulness to a student or a colleague. What would your "sales pitch" be?

AHKV: There is extensive research on the expressive power of different formalisms. Logical characterization of refinement relations tells us something about the distinguishing power of formalisms. For example, while the temporal logic CTL* is more expressive than the temporal logic CTL, the two logics have the same distinguishing power: if you have two systems and can distinguish between them with a CTL* formula (that is, your formula is satisfied only in one of the sys-

tems), then you should be able to distinguish between the two systems also with a CTL formula. Moreover, while CTL is not more expressive than LTL, we know that CTL is "more distinguishing" than LTL. These results have to do with the logical characterizations of trace containment and simulation. The distinguishing power of a specification formalism is useful when we compare systems, in particular an implementation and its abstraction: if we know that the properties we care about are specified in some formalism L, and our system refines the abstraction according to a refinement relation in which the satisfaction of specifications in L is preserved, then we can perform verification on the abstraction.

Luca: I am interested in how research collaborations start, as I like to recount "research-life stories" to PhD students and young researchers of all ages. Could you tell us how you started your collaboration on the award-winning paper?

AHKV: Subsets of us were already collaborating on other topics related to reactive models and model checking, and all of us shared a common belief that the field was in need to move from the limited setting of closed systems to a more general setting of open systems, that is, systems that interact with an environment. Open systems occur not only when the environment is fully or partly unknown, but also when a closed system is decomposed into multiple components, each of them representing an open system. To build "openness" into models and specifications as first-class citizens quickly leads to the game-theoretic (or "alternating") setting. It was this realization and the joint wish to provide a principled and systematic foundation for the modeling and verification of open systems which naturally led to this collaboration.

Luca: Did any of your subsequent research build explicitly on the results and the techniques you developed in your award-winning paper? Which of your subsequent results on alternating transition systems and their refinement relations do you like best? Is there any result obtained by other researchers that builds on your work and that you like in particular or found surprising?

AHKV: Various subsets of us pursued multiple research directions that developed the game-theoretic setting for modeling and verification further, and much remains to be done. Here are two examples. First, the game-theoretic setting and the alternating nature of inputs and outputs are now generally accepted as providing the proper semantic foundation for interface and contract formalisms for component-based design. Second, studying strategic behavior in multi-player games quickly leads to the importance of probabilistic behavior, say in the form of randomised decisions and strategies, of equilibria, when players have non-complementary objectives, and of auctions, when players need to spend resources

for decisions. All of these are still very active topics of research in computer-aided verification, and they also form a bridge to the algorithmic game theory community.

Luca: One can view your work as a bridge between concurrency theory and multi-agent systems. What impact do you think that your work has had on the multi-agent-system community? And what has our community learnt from the work done in the field of multi-agent systems? To your mind, what are the main differences and points of contact in the work done within those communities?

AHKV: Modeling interaction in multi-agent systems is of natural interest to planning problems studied in the AI community. In 2002, the International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS) was formed and the annual International Conference on Autonomous Agents and Multiagent Systems (AAMAS) was launched. The models, logics, and algorithms developed in the concurrency and formal methods communities have had a strong influence on research presented at AAMAS conferences over the past twenty years. Coincidentally, this year our paper on Alternating-Time Temporal Logic was chosen for the IFAAMAS Influential Paper Award.

Luca: What are the research topics that you find most interesting right now? Is there any specific problem in your current field of interest that you'd like to see solved?

AHKV: Research on formal verification and synthesis, including our paper, assumes that the model of the system is known. Over the last few years, reinforcement learning has emerged as a promising approach to the design of policies in scenarios where the model is not known and has to be learned by agents by exploration. This leads to an opportunity for research at the intersection of reactive synthesis and reinforcement learning. A potentially promising direction is to consider reinforcement learning for systems with multiple agents with both cooperative and adversarial interactions.

The realization that reactive systems have to satisfy their specifications in all environments has led to extensive research relating formal methods with game theory. Our paper added alternation to refinement relations. The transition from one to multiple players has been studied in computer science in several other contexts. For the basic problem of reachability in graphs, it amounts to moving from reachability to alternating reachability. We recently studied this shift in other fundamental graph problems, like the generation of weighted spanning trees, flows in networks, vertex covers, and more. In all these extensions, we consider a game between two players that take turns in jointly generating the outcome. One player aims at maximizing the value of the outcome (e.g., maximize the weight of the

spanning tree, the amount of flow that travels in the network, or the size of the vertex cover), whereas the second aims at minimizing the value. It is interesting to see how some fundamental properties of graph algorithms are lost in the alternating setting. For example, following a greedy strategy is not beneficial in alternating spanning trees, optimal strategies in alternating flow networks may use fractional flows, and while the vertex-cover problem is NP-complete, an optimal strategy for the maximizer player can be found in polynomial time. Many more questions in this setting are still open.

Luca: What advice would you give to a young researcher who is keen to start working on topics related to alternating transition systems and logics?

AHKV: One important piece of advice to young researchers is to question the orthodoxy. Sometimes it is necessary to learn everything that is known about a topic but then take a step back, look at the bigger picture, reexamine some of the fundamental assumptions behind the established ways of thinking, change the models that everyone has been using, and go beyond the incremental improvement of previous results. This is particularly true in formal methods, where no single model or approach fits everything. And young researchers stand a much better chance of having a really fresh new thought than those who have been at it for many years.

References

[1] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In D. Sangiorgi and R. de Simone, editors, *CONCUR '98: Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998, Proceedings*, volume 1466 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 1998.

[2] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In A. W. Mazurkiewicz and J. Winkowski, editors, *CONCUR '97: Concurrency Theory, 8th International Conference, Warsaw, Poland, July 1-4, 1997, Proceedings*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1997.

[3] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1996.

[4] U. Nestmann and B. C. Pierce. Decoding choice encodings. In U. Montanari and V. Sassone, editors, *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings*, volume 1119 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 1996.