Living in terms of Causes and Effects

Alex Yakovlev

Newcastle University, Newcastle upon Tyne, NE1 7RU, UK

alex.yakovlev@ncl.ac.uk

Abstract. The essay reflects on my most memorable experiences in studying causality and concurrency in the realm of electronic hardware. It covers thirty years of my friendship and collaboration with Maciej Koutny. This bond helped us build an important synergy at the cognitive and methodological level and create a unique academic environment enabling young researchers to tackle theoretical and practical problems in one of the most intriguing fields of computer science and engineering, the area of concurrency theory and asynchronous systems design. For me personally, Maciej has made strong impact on my understanding of semantics of concurrency and discovering the intricacies of the relationship between different models of concurrency that laid foundation to design methods and tools for asynchronous digital hardware. The paper concludes with the discussion of what actually Causality is, is it about representation (purely cognitive notion), or has it a physical meaning? My view it has place in both!

Keywords: Asynchronous Systems, Causality, Concurrency, Energy current, Petri nets, Signal Transition Graphs, Theory of Regions

1 Instead of introduction

When I joined the group of Professor Victor Varshavsky in 1980 as a PhD student ('aspirant' in Russian), I had very little idea about concurrency. However, the group used term "parallel asynchronous processes", actually meaning concurrent processes by them. In order to have a good feel what such processes are actually in real systems, I had a two prong attack at the literature. One side was understanding concurrency by reading about operating systems and various mechanisms used in them, such as semaphores and monitors. The other was modelling and designing the so-called Muller circuits, i.e. circuits whose behaviour didn't depend on delays. My first mentors here were Dr Leonid Rosenblum and Dr Vyacheslav Marakhovsky, as well as two other Varshavsky's aspirants Alex Taubin and Mike Kishinevsky, who had joined the group before me.

Muller's theory of asynchronous or better say speed-independent circuits was a great vehicle in learning the fine grain concurrency effects, such as arbitration, non-persistency and synchronisation. Understanding the dynamic behaviour of those circuits was like going through the baptism of concurrency.

Another good training vehicle was modelling communication protocols, and trying to reduce the model of a protocol between two objects to absolute minimum, keeping only the essential aspects of the inter-object interaction while hiding all the internal actions of the objects.

So, these three ingredients, parallel programs (in operating systems), protocols and asynchronous logic circuits formed me as a researcher in concurrent systems, and that brought me to my PhD level at the end of 1982. Of course, the beloved Petri nets were the key modelling formalism and they equipped me with the necessary understanding of the

relationship between structure and behaviour, static and dynamics, of concurrency models, let alone the wonderful axes of correspondence between Petri nets and asynchronous circuits.

Fig. 1. We weren't even 30 then! (left to right: Maciej Koutny, Luigi Mancini, Wouter Batelaan, Alex Yakovlev, Thomas Dongman Lee and Paul Ezhilchelvan, Newcastle Summer 1985)

Two years later I managed to land in Heathrow as a postdoc, funded by the Higher Education Ministry of the USSR and The British Council, and on the 17th October 1984 I first time arrived in Newcastle to what was then the Computing Laboratory (led by Prof Harry Whitfield). That was the place where I had met Prof Brian Randell, and heard about atomic actions in designing distributed systems, about various types of concurrent systems and languages to program them. I had also met with Prof David Kinniment, whose name was known to me as a pioneer in metastability, synchronisers and arbiters. At the end of 1984, a young research associate arrived from Warsaw to work with Brian on the formalisation of the problem defined as the Merlin-Randell protocol. After a couple of silent passings by each other, we started to talk. And that's how I first met with Maciej. I liked him from the start. We were close in age, in height, and Maciej had a very 'scientific beard', which to me was a good sign. I used to have had beard myself before but the Soviet Education Ministry instructors told me that I should look tidier when I go to Britain and so I had shaved it before that.

That was the start of our great friendship with Maciej. We used to spend hours walking on the coast in Tynemouth or in Newcastle's parks, talking about everything in life. Poland was the first country I had visited abroad as a student (1976), and so I had lots of warm memories of that time. Maciej, on the other hand, had already visited USSR before, too. We could talk about our impressions of each other's country. Also, as it happened both our families we far away from us – and here again was a similarity, let alone that Maciej's daugher Ola and my son Greg were borne in the same year of 1981. In August 1985 I had to go back to Leningrad while Maciej was of course staying in Newcas-

tle and later that year he was joined by Marta and Ola, but that was after I had already gone back to the USSR. We exchanged Christmas cards and yearly updates after that, little did we know then that our roads will meet again soon!

2 Asynchronous circuit theory

Working in Varshavsky's group was both a challenge and great fun. One could design circuits for some real applications such as, for example, a token-ring communication channel for an on-board multi-computer system with fault-tolerance levels achieved due to the effect of self-checking properties of asynchronous logic. At the same time, there was an encouragement to study theory of asynchronous circuits to fully understand and capture, formally, such notions as hazards and deadlocks in these circuits. Varshavsky associated asynchronous circuits with parallel programs and concurrent processes – this was a very useful analogy. There was always a team of people with whom to discuss concurrency. Petri nets were often in the centre of these debates.

One requirement, however, that Varshavsky imposed on us was that the formalisms had to be used in close relation with two main problems in asynchronous circuit design process, one of analysis and the other of synthesis. Analysis concerned taking an asynchronous circuit description and verifying whether its behaviour contained potential hazards, deadlocks and arbitration conditions. Synthesis was concerned with the way how one could derive the logic circuit netlist (i.e. a system of Boolean equations of the circuit gates) from an initial behavioural specification of the circuit.

Analysis in those days was approached as the process of generating the set of reachable states in the so-called Muller model of the circuit. Informally, the Muller model considered a circuit to be a network of logic elements. Each element was a combination of a functional block (described by a Boolean function) with zero delay and a delay element, whose delay was finite but unbounded. Due to this description of an element, it was possible to separate the notion of excitation of the output signal of the element (when the value of its Boolean function was different from the value of the output of its delay element) and the notion of firing the signal, i.e. changing its value to the opposite one, say from 0 to 1, or from 1 to 0. An interesting property of Muller circuits was that of semi-modularity. The circuit was semi-modular if it would never reach a state in which an element would be excited say in its value 0 (we wrote it with an asterisk 0*) and not change to 1. Thus, the circuit with a transition from state where some signal was labelled 0* to the state with a stable 0, would be called non-semi-modular. This behaviour raised an immediate analogy with a notion of non-persistence in Petri nets. To detect non-semi-modularity in a circuit we had to go through all reachable states.

The reachable state space, for the whole circuit, could of course grow fast due to the amount of concurrency in the circuit. So, the team was concerned with finding efficient ways to tackle the state space explosion problem. Alexander Taubin's PhD was largely focused on analysis. Together with Mike Kishinevsky, they developed a method of analysis based on Boolean characteristic functions of the reachable state space as well as characteristic functions of some properties such as semi-modularity of the circuit. At some point, Yury Mamrukov joined the group with a bunch of fresh ideas on how to implement these methods by his highly efficient ways of representing characteristic functions in bracketed form, which was well before the world learned about BDDs! And moreover, he could pack large state vectors in a small number of memory words, so that his software, written in PL (if my memory doesn't let me down!), could easily cope with the state spaces of the scale

of 2^50 or more. We rarely could produce circuits (manually) bigger than 50 signals. So, overall, Varshavsky and Marakhovsky, who were the main circuit generators, were quite happy with analysis … but for some time.

In terms of automating synthesis, things were lagging behind, although the methods and theory was gradually developed. Kishinevsky had developed in his PhD thesis some new methods for synthesis of distributive circuits using NAND only gates, and semi-modular circuits using NAND and NOR gates, with limited fan-in and fan-out. Those were theoretically important results, even though they didn't lead to particularly efficient practical designs. The main ways of synthesis were the following: (i) manual gate-level design with subsequent analysis of semi-modularity, (ii) direct translation of Petri nets to circuits using so called David cells, (iii) formally justified (yet still unautomated procedures) of synthesis from state graphs (aka Muller diagrams) and, (iv) some initial (again, unautomated yet) procedures of synthesis from signal graphs (later called STGs by T.-A. Chu). The latter was the area where I contributed with my PhD thesis with application to the design of controllers for protocols and interfaces, such as UNIBUS, VME Bus etc. Those were relatively small control circuits. Larger controllers, such as those for the above-mentioned token-ring communication channel for our industrial partners developing on-board computers, were designed using direct translation of Petri nets into David cell circuits.

As things progressed into VLSI in the latter half of the 1980s, more computational power emerged. There was no longer a need for Mamrukov's renting night hours at the Computer Centre of Gostinny Dvor (the biggest department store in Leningrad). People could use IBM PCs!

3 Models with true concurrency

We needed models and tools to handle synthesis and analysis more efficiently and, ideally, interactively. The road to efficiency was seen in avoiding an explicit exploration of states for concurrent events produced by interleaving of these events. Namely, if two events A and B were concurrently enabled in the circuit, the reachability analysis explored both traces A;B and B;A, thereby producing four states forming a so-called diamond. As the number of concurrent events (and hence signals) grew as n, the state space exploded as O(2^n). Something different had to be used for analysis and synthesis of highly concurrent models – we all sought to liberate ourselves from the tyranny of interleaving! Leonid Rosenblum and I had already pioneered Signal Graphs in our paper at Turin's Workshop on Timed Petri nets, July 1985. Interestingly, Tam-Anh Chu from MIT had his first paper on STGs published a bit later, in November 1985. We had no idea about each other's work until a few years after that.

That was the time when the model of Change Diagrams emerged. It didn't have a Petri net underneath, though it retained the important aspect of modelling true concurrency in the form of partial orders. Change diagrams captured both forms of causal dependency or precedence, namely strong (AND) causality and weak (OR) causality. The fact that OR causality could be captured in Change Diagrams in an explicit form was very useful because logic circuits, build of gates with NAND, NOR, AND-OR-INVERT functions, ultimately could exhibit both of those forms of precedence. However, one somewhat esoteric feature of Change Diagrams was a bit unwieldy and non-intuitive. Firstly, they required use of negative marking, namely tokens were borrowed from the input arcs which were not active for the OR-causal vertices. Secondly, because Change Diagrams weren't meant to model processes with choice, they had so-called disengageable arcs, in order to capture the

initialisation of a choice-free process. I wasn't happy about those features which I had thought would be difficult to promote to practitioners in the future. Nevertheless, Alex Taubin, Mike Kishinevsky and Alex Kondratyev managed to bring Change Diagrams to the level of excellent software tools TRANAL and TRASYN, that were developed within the suite of tools under Varshavsky's co-operative TRASSA – these were the first self-timed CAD tools in the world that could be run on a PC. A nice book was later published by John Wiley & Sons, which I had a pleasure to translate into English (when I was already in Newcastle), and which described all the theory and algorithms behind analysis and synthesis methods for Change Diagrams. The book was equipped with a CD containing these tools. Some people from the Asynchronous design community around the world used these tools, but they didn't get far, possibly partly because of those unintuitive aspects of Change Diagrams.

Fig. 2. No place for interleaving!? (left to right: Alex Yakovlev, Leonid Rosenblum, Marta Koutny, Maria Yakovlev, Maciej Koutny; place: Pizzeria Francesca, Newcastle 2000)

For me, it was fairly clear that use of Petri nets was essential, both for the reasons of their wider acceptance by research community and also because I was really curious about certain relationships between Petri nets and state-based models. From the late 1980s I teased myself with the following problems: (a) what classes of Petri nets corresponded to the classes of distributive, semi-modular and speed-independent circuits (strictly speaking, lattices on cumulative states in terms of Muller theory), and in particular what were the conditions of Petri net transition labelling; and (b) what type of concurrency relations between events would lead to state-models with distributivity and semi-modularity. I managed to solve those problems around 1990 as I was leaving Russia and spent a year in Wales, after which I came back to Newcastle as a lecturer.

The solution to the first problem was that the class of safe and persistent nets with an injective labelling corresponded to distributive lattices, while k-bounded and persistent nets to semi-modular lattices. As for the second problem, the correspondence was reached

at the level of binary concurrency relations which could always be generalised to n-ary relations for distributive lattices. For example, if we had three actions, A, B and C, which were mutually pairwise concurrent, i.e. CONC={(A,B),(B,C),(A,C)}, then for a distributive process we could also generalise CONC relation to a 3-way concurrency tuple (A,B,C), whereas for a semi-modular yet non-distributive process the 3-way relation didn't hold. This was an interesting result which somehow led to the notion of resource-constrained behaviours. For example, in a non-distributive case, although actions could have been executed in parallel without any data dependency, if we had only two processors, we could not execute all three of them in parallel.

These results were later published in my papers of the early 1990s when I was in Newcastle. I also managed then to clarify and remove some modelling restrictions in Tam-Ahn Chu's STGs, and present a general unified STG model with the appropriate characterisations and relationships with state-based models and trace-theoretic models of delay-insensitive circuits (of the Dutch school at TU Eindhoven). In this work a great help came from my friend Luciano Lavagno, who was then finalising his PhD thesis at UC Berkeley under the supervision of Prof Alberto Sangiovanni-Vincentelli.

Later, together with Kishinevsky, Kondratyev and Lavagno we solved another interesting problem, namely that of how to unify Change Diagrams and Petri nets. We managed to identify the classes of Petri nets exactly matching Change Diagrams at the level of bijective labelling. We also showed that it was possible to unify and generalise these models by Causal Logic Nets, which finally raised the modelling power to that of Turing Machine and allowed the modelling of non-commutative state transition behaviour in a purely causal form. Some crucial proofs were done by Marta Koutny, who was then my PhD student! Marta co-authored our key paper on modelling OR-causality in Formal Methods in System Design. In all those developments I was also very happy to discuss ideas with Maciej, who gave his word of wisdom what concerned the semantics of concurrency and notions of steps in all our inter-modelling characterisations.

In those 1990s, in search of escape from semantical interleaving, a big push was also made on Petri net unfoldings. Firstly, it was through our work with my first PhD student at Newcastle Alex Semenov, with whom we developed many algorithms for the unfoldings of Petri nets, STGs, timed Petri nets, and Petri nets with read arcs. Semenov developed first unfolding-based software tools for verification and synthesis of asynchronous circuits – called PUNT. All this work was nicely reported in his thesis, which was successfully passed in front of Maciej, who acted as internal examiner, and Prof Steve Furber, who was external examiner. Unfortunately, this unfolding research was interrupted in 1997 when Alex Semenov decided to leave academia for work in the City of London. But, to our luck, this interruption wasn't for that long, as in 1999 Maciej accepted Victor Khomenko as his PhD student to work on unfoldings, and I gladly joined their team. Victor brought unfolding methods and tools based on them to another level of sophistication and computational power (in particular, an interesting result was obtained for merged processes which effectively compressed the branching process to a compact structurally cyclic, but semantically acyclic graph), and after a few years, new methods and tools for verification and synthesis of asynchronous logic appeared, which used unfoldings and SAT solvers. They were PUNF (a parallel unfolder) and MPSAT (unfolding-based verification and synthesis engine). Up till now they are being advanced and used within our Workcraft tool suite. Thus, true concurrency was elevated from its pure theoretical attraction to the level of a working instrument!

4 Synthesis of Petri nets and Petrify

Let's now get back to the year of 1994. That year was very important for the whole community of people surrounding me in promoting Petri nets as a key modelling language underlying asynchronous control logic design. Why? In that year, after the previous two years of establishing close collaboration with Kishinevsky, Kondratyev and Lavagno on the investigation of STGs as a theoretical and practical underpinning for asynchronous circuit design, all of us started to closely interact with Jordi Cortadella, a young professor from UPC in Barcelona. June 1994 was the time when Luciano and I visited Jordi before going to the Petri nets annual conference held in Zaragoza, where I presented our work on OR-causality models. In the same year, Mike Kishinevsky was a visiting EPSRC fellow in my group in Newcastle and in August 1994 we went to Windermere (Lake District) to participate in the first Amulet workshop organised by Prof Steve Furber. Prior to that trip we had an interesting chat with Maciej, who had for the first time mentioned to us the fact of an interesting theoretical characterisation of behavioural models of concurrent systems – Theory of Regions. He pointed us to the work of Nielsen, Rozenberg and Thiagarajian of 1992 (NPT92) on Elementary Transition Systems. 'Infected' with this fascinating read, we went to Windermere. While we interacted at length with the whole of Furber's group, 'infecting' them with the use of STGs in designing controllers for pipelines in the first Amulet processor (the PhD dissertation of Nigel Paver was very useful then!), our minds were captivated by this idea of Regions. It was so elegant and beautiful – Mike and I spent hours walking around our hotel at night discussing what would be its use in our more practical setting of asynchronous circuit design automation. Our first idea was that, with Regions we could take the model of a circuit in states, obtained for example, through the reachability analysis, and convert it into the model with true concurrency – say for the purposes of visualisation. A circuit with a state graph with thousand vertices could be shown nicely in a Petri net graph with some fifty vertices! Shortly after we got back from Windermere, we tried to formalise the synthesis of Petri nets and STGs from the state graphs through the notion of regions, by re-defining the key state and event separation axioms of the region theory in terms of excitation closure. That first draft was immediately sent to Jordi Cortadella, Alex Kondratyev and Luciano Lavagno. A very hot discussion of what can be done with all this new theory followed, and all of us had agreed that the visualisation was a good anecdote to promote it! Jordi, with his remarkable talent of making things happen in real and very fast, had managed, within a couple of weeks (!), to develop practical algorithms of constructing regions in the BDD framework. And that was virtually the birth of a new tool called Petrify. It is hard for me to remember who exactly came up with this funny name but we all loved it – as we were all petrified by its elegance!

So, the truly historic tool Petrify was born somewhere around September 1994. Maciej's suggestion to look at Regions and the NPT92 paper was monumental!

Shortly after, I managed to have solved an interesting challenge posed by Charlie Molnar to the asynchronous community, which was to design control logic for a counterflow pipeline processor (CFPP). Molnar's description of the control consisted of a 5-state diagram which intertwined concurrency and arbitrating choice in a very smart and compact interleaving. The puzzle was as to how to unravel that interleaving? Some solutions appeared in the community, but all of them departed from the original state graph into other interleaving-based models such as CSP and process algebra. I managed to find a way of performing an equivalent (very close to isomorphism) transformation on this state graph – by splitting a state and inserting a dummy event. And bingo, that gave me a way to cover

the 6-state graph with regions, and hence synthesize an equivalent Petri net, involving produce and consume arcs, as well as read-arcs. The rest was a piece of cake with my techniques for refining and converting the Petri net to an STG and then synthesise it to a control circuit. This CFPP example of using Regions and Petrify in synergy, not only for visualisation but circuit synthesis, had convinced me that we were on the right track (a paper was published in Formal Methods in System Design).

Then followed a sequence of developments in Petrify of various features, more related to asynchronous control logic synthesis, such as solving state coding problem, logic decomposition for hazard-free technology mapping, introduction of relative timing for timing optimisation, and numerous model conversions between state and event based representations. The paper of 1997 (Japan's IEICE Transactions on Information and Systems) about Petrify is one of the top-cited papers for all of us. Around 2000, Petrify and STGs, underpinning it, became a de facto standard technology in asynchronous circuit synthesis. People in academia and industry had used it with enthusiasm. In 2002 we published a book on STG based synthesis in Springer. In the same year we were nominated as finalists of the European Descartes Prize!

Meanwhile, the years 1999-2000, were significant for our Newcastle team, and largely thanks to our work on Unfoldings and Regions. In 2000, Marta defended her PhD thesis (with Dr Philippe Darondeau as external examiner) – which was largely about Regions. The thesis title was "Relating formal models of concurrency for the modelling of asynchronous digital hardware". Its particular, emphasis was on the class of semi-elementary transition systems and elementary net systems with inhibitor arcs (ENI-systems). The use of steps, a-priori and a-posteriori semantics in Marta's research helped me to understand better some intricacies of the modelling of concurrency in systems with inhibitors. The other important fact about that period was that in those days Maciej and I founded and actively promoted Asynchronous Systems Lab (ASL) and our ASL seminar, which had later gone through two decades of its successful run. ASL seminar series helped many of our RAs and PhDs to familiarise themselves with the fine grains of concurrency theory and its use in better understanding the behaviour of asynchronous circuits and systems. Causality and its various forms were essential in this process. These foundations helped building a new generation of theories of Petri net synthesis and tools for modelling electronic systems. Examples of such a wonderful synergy of theory and applications were methods and tools for visualisation of asynchronous circuits models, particularly in the PhD work of Victor Khomenko and Agnes Madalinski, synthesis of nets with policies in our work with Philippe Darondeau, modelling of GALS systems in the PhD work of Sohini Dasgupta and Johnson Fernandes.

Our constant interactions in the ASL community had led to other impressive results. They included methods and tools for direct mapping of Petri nets and STGs (recall David cells!), in the PhD work of Danil Sokolov, with active involvement of Dr Alex Bystrov who worked as an RA on a series of crucial EPSRC grants. Two EPSRC-sponsored and highly productive visits of Dr Nikolay Starodoubtsev to Newcastle had led to methods of synthesis of asynchronous control logic in negative gates and wire delay-insensitivity, all through finding elaborate ways of refining causality in STGs. This later helped our PhD student Yu Li to develop his method of refining STGs to lift the isochronic fork timing assumptions to much weaker timing assumptions. Work on Asynchronous Communication Mechanisms (aka ACMs), with active involvement of Dr Fei Xia and Dr Ian Clark, and our collaboration with Prof Tony Davies (from Kings College London) and Dr Hugo Simpson and Eric Campbell (from MBDA), helped Delong Shang, in his PhD work, to turn design methods for ACMs to real silicon chips. For example, our HADIC chip, fabricated in

0.6micron CMOS process in 2000, was the first harbinger in the series of chip tape-outs in the group, which were all strongly coupled to our research on causality and concurrency.

5 Tools, tools, tools: Interpreted Graph Models and Workcraft

In the late 1990s, I started to actively interact with Dr Oleg Mayevsky, my friend and former colleague from Varshavsky group. Oleg was an associate professor at the Kyrgyz-Slavic Russian University (KRSU), located in Bishkek, the capital of Kyrgyzstan, a relatively small country in Central Asia (a former Soviet republic). Thanks to the energy of Oleg's head of department Prof Gennady Desyatkov and Dr Chris Phillips, who acted as coordinator of EU-funded TEMPUS project between Newcastle and KRSU, aimed at developing a Master curriculum in Computer Science in KRSU, we had an excellent revival of the research collaboration with Oleg. The first result of this collaboration was the EPSRC-funded PhD work of Danil Sokolov between 2001 and 2004, which was linked to the EPSRC BESST project on asynchronous control logic synthesis from Petri nets.

Fig. 3. ASL is conquering the Great Wall with Concurrency and Causality (Maciej, Marta, Ola, Maria and Alex; trip to China, 2006)

Around 2002, thanks to the ASL funding, we had Oleg Mayevsky at Newcastle as a senior RA for six months, and productive work with David Kinniment, Gordon Russell, Alex Bystrov and myself on time measurement and time-to-digital converters (a famous Time Amplifier was invented then!). Subsequently, in 2005, when I had another vacant PhD studentship from EPSRC, linked to a project on networks on chip, I was fortunate to be introduced by Mayevsky and Desyatkov to Andrey Mokhov, who was finishing his MSc degree at KRSU and had just been amongst the finalists of the World Programming Contest. The arrival of Andrey in Newcastle in September 2005 had given a massive boost in our ASL team as he was able to inject his great potential and enthusiasm into the group. In his research he took the model of partial orders and added there conditionals to capture

choice, produce an entirely new model Conditional Partial Order Graphs (CPOGs), which enabled a whole range of new developments, from the models of communication channels based on phase encoding (PhD of Enzo d'Alessandro), to control of microarchitecture in CPUs (PhD of Max Rykunov), and to more general graph algebraic approaches (promoted by Andrey himself) in wide variety of applications, requiring compact modelling of multitudes of scenarios.

With Andrey's PhD on CPOGs completed in 2009, and PhD work of another KRSU alumni, Ivan Poliakov, who started work on unifying our graph based models, such as Petri nets, Circuit Petri nets, STGs, CPOGs and data-flow structure under the new tool set called Workcraft. Ivan's work was linked to the EPSRC-funded project SEDATE, which launched our collaboration with the Manchester CAD group led by Dr Doug Edwards. The aim of the project was to develop methods and tools for synthesis of asynchronous data-paths. In this work we actively promoted our approach to modelling strong (AND) and weak (OR) causality, previously used for control models such as STGs to static data-flow structures (SDFS). Here, it was possible to relate these causality paradigms with the data-path notions of full acknowledgement and early propagation, respectively. Notions of forward data tokens and anti-tokens, expressed in the work of others, were unified by our Petri net based interpretations. Danil and Andrey were closely working with Ivan, as well as we had fruitful interactions with our PhD student Zhou Yu and Will Toms (from Manchester). Accompanied by the PhD work of Julian Murphy and Ashur Rafiev (also from KRSU), on exploiting causality and concurrency in circuits for cryptographic applications, this gave the group a great pathway to impact – in which we worked with Atmel Smart card ICs, where our tools (largely thanks to Danil's efforts) were used in mid 2000s.

The significance of the transition to the toolset of Workcraft, around 2007, is hard to overestimate. Our previous tools were all command-driven and based on textual representation of state and event based models, where the key format was that of a "dot-g" (we often pronounced it 'dodgy'!) file. Workcraft stipulated that the tools had to be linked to an interactive GUI-based framework, to which other problem-oriented tools, such as Petrify and MPSAT, could be connected as backends or plugins. That view, promoted in Ivan's thesis, and later driven by the developments of new tools by PhD students Arseniy Alekseyev (again, from KRSU!), and Stas Golubcovs, coordinated by Danil Sokolov, has resulted in a powerful CAD environment, which has by now, the year 2018, given rise to the new evolution step thanks to the efforts of Danil, Victor and Andrey.

All this work is now at an entirely new level of maturity and no wonder it has been noticed again (after our prior experience with Atmel) by industry. In 2014, we launched a fruitful collaboration with Dialog Semiconductor, a company specialising in developing IP solutions for electronic power management and communications. This is the area of analogue-mixed-signal (AMS) electronics, in which the significant role is played by islands of embedded digital logic. We gave this logic the name of 'little digital' on the analogy of the digital logic that was of modest complexity in the era before VLSI design. Then in the 50s and 60s, that digital design was largely clock-free, i.e. asynchronous. Normally AMS designers would design such logic by hand and then include its components into the overall process of painstaking simulations – which would last for days if not weeks. With a bit of destiny turn, in 2013, I was approached by Dr David Lloyd, from Dialog, who used to be a member of the Amulet group in Manchester and was interested in asynchronous design. David wanted to try our Petrify tool in designing control logic for power regulators. We started active collaboration which was partly initially sponsored by EPSRC in the A4A (Async for Analog) grant. The prospect of automating the design of little digital was very attractive.

At the moment this industrial take-up is strong and what's important, today we are teaching practical engineers to think in terms of causality and concurrency, and they can immediately see these concepts in action – all through the Workcraft tools – they can run extensive simulations, they can verify circuits and their specifications, they can synthesise and decompose their circuits, and they can visualise the behaviour the models using such representations as waveforms. For example, Jonny Beaumont's recent PhD work was about Concepts – these are elementary building blocks of the designer's knowledge about the asynchronous circuit behaviour. They are captured and can be converted to synthesisable specifications.

I have not mentioned many other developments within Workcraft in which Petri nets play the behaviour-interpreting role to capture the semantics of the specific instrumental models. These include the models of structured occurrence nets (SONs), which have been actively investigated by Maciej, Brian Randell and their ex-PhD student Bowen Li and applied in the application domain of crime investigations, models of the communication fabrics of xMAS for networks on chip and GALS systems, developed by Dr Frank Burns, models of instruction set architectures, developed in the PhD theses Alessandro De Gennaro and Georgy Lukianov, and other models. We are also advancing STGs into new applications for AMS through PhD work of Vladimir Dubikhin, who combines our Workcraft tool with the analogue verification methods based on Labelled Petri nets and tool LEMA of Prof Chris Myers (University of Utah) to produce a new design flow for AMS with asynchronous control (behaviour mining and model generation are the core of his method). Further to that, work of Sergey Mileiko is aimed on applying STG based methods to design controllers for Switched Capacitor Converters, in collaboration with Alex Kushnerov (graduate of Ben-Gurion University and now working at Intel, Israel).

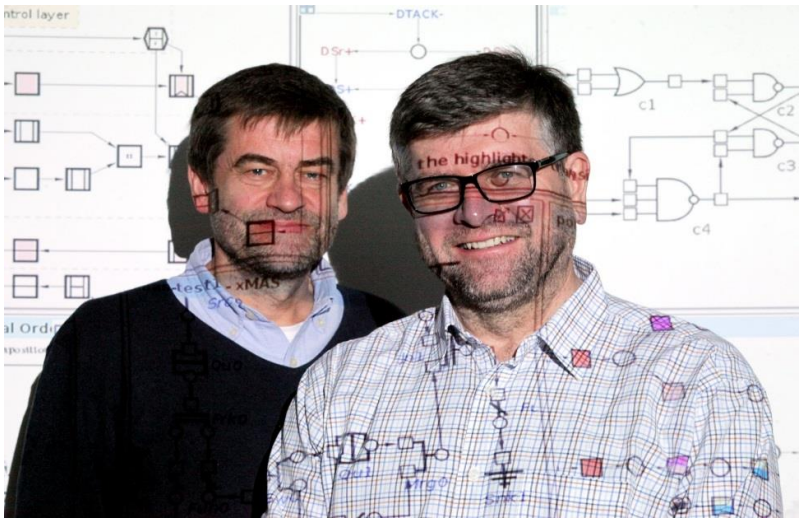

Fig. 4. Tools, only tools, nothing but tools (our REF-2014 Impact showcase; Newcastle 2015)

All these models have distinct elements of Maciej's fundamental notions of the semantics of concurrency and causal representations based on graphs, showing causality through the relations of precedence, control and data dependency, timing relations.

6 What is actually Causality?

So far, I have been addressing the notion of causality in the framework of event-based models, namely Petri nets. The relationship between events (transitions or their occurrences) in Petri nets is based on temporal aspect, i.e. event A precedes event B. This precedence relation is captured by an arc in a graph, and it is sometimes important to show the dynamics of this relation using the notion of an explicit condition (place) which can have a token or not have it. The idea of a token is important to be able to interpret the behaviour in the form of states of the system (token marking). If a condition between two events A and B is marked with a token the causal dependence is active, meaning that the event A has occurred and may lead to the occurrence of B at any moment, possibly conditioned by other causes of B incident on transition B.

The notion of causality based on precedence is probably best captured by models such as occurrence nets or event structures, which are acyclic and show the causal relationships purely in the form of directed arcs.

People studying causality in different forms and application domains may give it other interpretations and emphasis, which may not be directly linked to the temporal aspect. However, very often the temporal aspect, or the notion of precedence, is implied there. For example, the work of Professor Judea Pearl on causality modelling is more of the probabilistic nature, where causation between facts can be implied by some forms of high likelihood or probability of something to be true based on the fact of something else being true. This form of causation associated with knowledge or facts thereof can often be cast on the timeline. Therefore, Pearl's models can be translated to say SONs or event structures. In similar vein, of more cognitive nature, one can see the works of other researchers of causality, such for example as Steven Sloman (see next section).

My interest in causality is nowadays extending from cognitive to physical nature. I have recently become interested in electromagnetism and its relation to causality. Here one important element is the idea of energy flow, i.e. the transfer of energy in space and in time and its effects on the matter. An example of a physical causal system could be a system consisting of energy storage, such as a capacitor, and a load, a switching circuit whose power signal is connected to the capacitor. Here clearly we cannot obtain any switching behaviour if the voltage level on the capacitor is not sufficient to overcome the threshold voltage of the switching device in the load.

Further thoughts in physical direction bring me to the connections between types of logical (AND, OR) causality at the event level and notions of integration, differentiation and proportionality, typically used in automatic control. Indeed, think about PID control and relations between cause(s) and effect(s). Fig. 5 illustrates this connection. AND causality is linked to the integration paradigm, which is a positive hysteresis – we accumulate enough of the increasing causes before we see the effect in the rising edge of the signal. Similarly, we need to accumulate the decreasing causes before the falling edge of the effect. OR causality is linked to the differentiation paradigm, which is a negative hysteresis – we react to the effects early, for both the rising edge and the falling edge. The third case, of combinational dependence, has no hysteresis at all – it corresponds to the proportional paradigm.

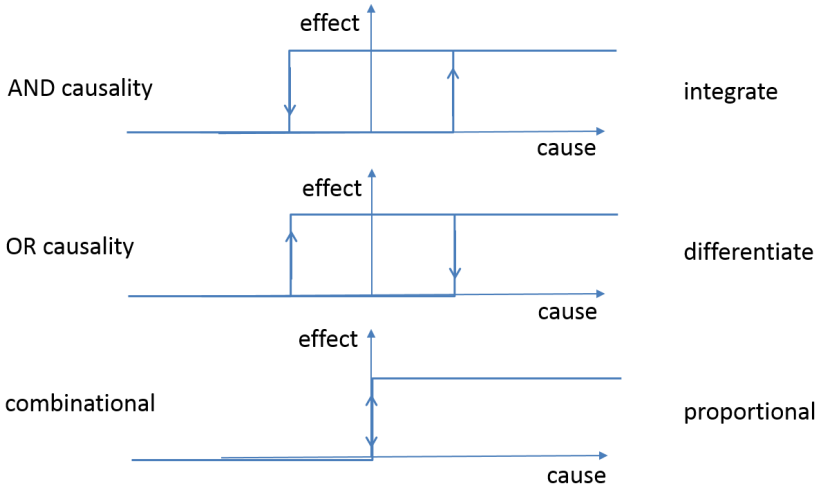

Fig. 5. Relationship between types of causality and PID control

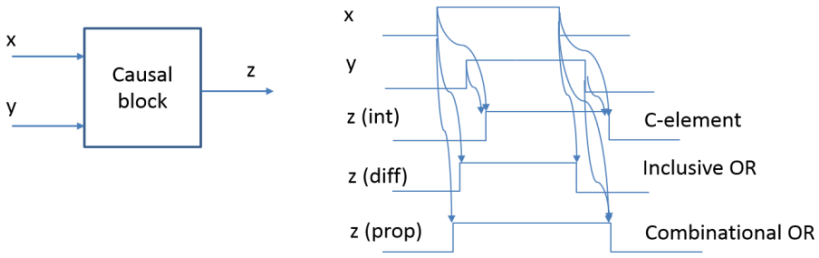

Fig. 6. Example of causality and PID forms for the case of two causes x and y and effect z.

All these paradigms can be implemented using logic circuits such as C-element (Integrate), inclusive OR (differentiate) and combinational OR (proportional) as illustrated in Fig. 6.

So, what is Causality after all (or better say before all!)? This question is probably equivalent to what is Time?

Steven Sloman in his book on Causal Models (published by Oxford University Press, 2005) refers to Bertrand Russell, who argued in his famous article "On the notion of cause" (1913) that "invariant laws aren't causal at all but rather mathematical". Sloman then writes: "Perhaps the world is nothing but a flow of energy … Perhaps, everything we misconstrue as cause and effect is just energy flow directed by mathematical relations that have determined the course of history and will determine our destiny in a long chain of events linked by the structure of energy in time and space." He then makes a sort of disclaimer that his book "isn't about metaphysics. It's about representation".

That's exactly the Gordian knot of the question of what Causality is. I agree with Sloman in that Causality has place in representation and hence plays a key instrumental role in cognitive science – that's what we have been exploring with Maciej in our research in concurrency. However, I tend to disagree that Causality has no physical meaning. Even if the world is modulated by the flow of energy, as Oliver Heaviside called "energy current"

in his Electrical Papers (Vol.2, pp.91-95), this is physics and it has causal meaning. Not all "invariant laws" aren't causal! If we go back to the work of Galileo and Newton, we can find important geometric proportionality between energy and momentum (associated with mass) via a spatiotemporal coefficient that is velocity. It is important that this proportionality is geometric – this makes relation between energy as cause and momentum as effect physical! As formulated by Heaviside for electromagnetism, we have this proportionality between energy current (aka Poynting vector) and electromagnetic fields mediated by the velocity of light, and this is also causal. There are of course some relationships in physics which are, quite mistakenly, interpreted as causal, which are in fact purely algebraic – such for example as the relationship between electric and magnetic field. In other words, when one looks at equations in physics one has to distinguish geometric (physical) proportionality from algebraic (mathematical). Some philosophy researchers, e.g. Ed Dellian, even put the names of the great scientists at this boundary – Newton at the former and Leibniz at the latter.

In my opinion, the physical world in which everything is governed by the energy current does not stop being Causal. Causality is not only representational – it is fundamental! On this highly optimistic point I should perhaps stop.

7 Instead of conclusion

The last word here will however have to be with classic philosophers …

PROPOSITION 28 in Part I of Spinoza's Ethics says: "Every individual, or each thing which is finite and has a determined existence, is not able to exist or to be determined to act unless it is determined to exist and act by another cause which is also finite and which has a determined existence: and again, this further cause is not able to exist or to be determined to act unless it is determined to exist and act by another which is again finite and has a determined existence, and so on to infinity."'

Happy Birthday, Dear Maciej!!!