Theoretical Computer Science 546 (2014) 3–6

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Trust in event structures

Mogens Nielsen

Department of Computer Science, Aarhus University, Denmark

ARTICLE INFO

ABSTRACT

A tribute to Glynn Winskel on his 60th birthday, including some notes on the role of Winskel's event structures in computational trust.

1. Introduction

It is hard to believe that Glynn is now turning 60. I first met Glynn 35 years ago in Edinburgh, when he was a PhD student under the supervision of Gordon Plotkin. The topic of Glynn's PhD studies was mathematical models for concurrent computations, and it just so happened that this was also my main interest during my postdoc stay in Edinburgh at the time. I was fortunate to get involved in joint work with Glynn and Gordon on a new model called event structures, a fortune which turned out to be the start of a long lasting close collaboration and friendship, with invaluable impact not only for me personally, but also for my home university.

After submitting his PhD dissertation in Edinburgh in 1980, Glynn joined us at Aarhus University as a postdoc. He then went to Carnegie Mellon University for a few years, before taking up a position at Cambridge University. But we were lucky to get Glynn back to Aarhus for a professorship in Theoretical Computer Science in 1988, and he stayed with us for twelve years until he took up his current professorship at Cambridge University.

During his twelve years as a professor in Aarhus, Glynn was the key player in a dramatic development of theoretical computer science at our university. In the early 90's the Danish National Research Foundation was established, introducing a new concept of a Centre of Excellence in Denmark. This was a very attractive concept with sizable funding focusing on basic research, with corresponding high competition across all of academia. Under Glynn's leadership we were fortunate to get one of the first 23 such centres, BRICS – Basic Research in Computer Science. As the director of BRICS, Glynn's visions, ideas and inspiration were the key factors not only in obtaining the centre, but also in its success over more than a decade.

In his BRICS inaugural talk in 1994, Glynn gives the following one-line definition of computer science: *Computer science is the management of complexity in the construction and analysis of systems*. And managing complexity is an important trademark for Glynn's approach to research. He always has a clear focus on important and challenging fundamental questions. This was clearly evident in his role as director of BRICS, and also more generally in his role as one of the leading figures internationally within the broad field of the mathematical foundation of computation.

Glynn's significant contributions to this field are many and diverse. As one prime example one has to mention his work on event structures as a mathematical model for concurrent computations. Today, event structures are established as a well-known independent research area – studied, developed further, and applied by many researchers all over the world. Event structures are used in a variety of otherwise very different research areas, including e.g. areas such as hardware design, data security, systems biology, business processes, and still today they turn up in unexpected new contexts – more on this later.

As another example, Glynn initiated the categorical approach to models for concurrency, not only contributing to a deep understanding of the relationships between the many models studied within concurrent computation, but also to a foundation for defining and understanding the many fundamental concepts of such models, including e.g. the foundation

http://dx.doi.org/10.1016/j.tcs.2014.02.039
0304-3975/2014 Published by Elsevier B.V.

for denotational semantics of process languages. This categorical approach also led to an understanding of and a guideline for the notion of behavioural equivalence between processes, notably the influential concept of open maps.

But these are only a few examples from the past. During his career, Glynn has contributed to many different areas of theoretical computer science, including semantics, domain theory, logic of programs, verification of hardware, model checking, security, probabilistic and quantum computation and much more. Currently Glynn pursues a major task in establishing a new foundation for concurrent computation based on the notion of concurrent games, for which he has been awarded one of the highly prestigious Advanced Grants from the European Research Council.

I have focused here on Glynn's research contributions, but I am sure that I speak on behalf of many of his students when I also mention his inspiration in teaching and supervision. His book "The formal semantics of programming languages" has been used widely all over the world as a standard text book for undergraduate courses. And as a supervisor, Glynn has trained more than 20 PhD students, who have all experienced and benefitted from his enthusiasm and his generosity with ideas, – many of them now holding prestigious positions in academia and industry worldwide.

I have personally experienced the same generosity many times. Glynn is always eager to share and discuss new ideas whenever we meet, and fortunately this has happened many times over the past 35 years of personal friendship with Glynn and his closest family, his wife Kirsten and their two daughters, Sofie and Stine.

Thank you Glynn – and Happy Birthday.

2. Computational trust and event structures

As mentioned above, event structures have turned up in many unexpected contexts as a useful modelling tool, and here I would like to mention briefly one such context, in which I have been involved personally.

A few years ago I was involved in a European project dealing with *computational trust*, SECURE [4]. Computational trust covers the study of alternatives to traditional security mechanisms in the Global Computing scenario, making security related decisions based on computational notions of trust resembling the trust relationship among human beings. The study started in the late 90's with the so-called trust management systems [1,2]. Later computational trust has been developed in many different directions. For good surveys on computational trust see e.g. [5,6], and [12].

In SECURE we were mainly concerned with the idea of building trust in a computing agent based on a notion of *reputation*. The computational notions of reputation are typically based on behavioural *observations* of computing agents, as well as *recommendations* communicated among computing agents. In the following I focus on some of our work on modelling these concepts, which in SECURE served as the core of a considerable implementation effort as reported e.g. in [3].

A computing agent in the Global Computing scenario will typically interact with its environment exchanging messages following the patterns of interaction protocols, and we were looking for a computational version of building trust in an agent based on the agent's behaviour in such protocols. And in the modelling of protocols, event structures turned out to be particularly well suited, introduced as a generic model for patterns of behaviour over an abstract set of events. The idea is simple and easy to explain.

We focus here on the original definition of an event structure from [11] in terms of three fundamental relationships between events in concurrent computations: *causality* (the occurrence of one event being depending on the occurrence of another), *conflict* (two events excluding each other from occurring), and *independence* (two events potentially occurring concurrently). And as mentioned above we restrict ourselves to finite protocols and hence finite event structures.

Definition 1 *(Finite event structure).* A finite *event structure* is a triple $(E, \leqslant, \#)$ consisting of a finite set E of *events* which are partially ordered by $\leqslant$, the *causality relation*, and where $\#$ is a binary, symmetric, irreflexive relation $\# \subseteq E \times E$, called the *conflict relation*. The relations $\leqslant$ and $\#$ satisfy for all $e, e', e'' \in E$

$$\text{if } e \# e' \text{ and } e' \leqslant e'' \text{ then } e \# e''.$$

We say that events $e, e' \in E$ are *independent* if they are neither in the causality or the conflict relation, and that they are in *immediate conflict* if they are in conflict, $e \# e'$, and

$$\text{for all } e_0 \leqslant e \text{ and } e'_0 \leqslant e', \text{ if } e_0 \# e'_0 \text{ then } e_0 = e \text{ and } e'_0 = e'.$$

As an example, the event structure in Fig. 1 represents a protocol, where an agent may ask a bank for the transfer of electronic cash from its bank account to an electronic wallet. After making the request, the agent observes that the request is either rejected (event `r`) or granted (`g`). After a successful grant, the agent could observe that the cash sent in the transaction is forged (`f`), or perhaps run an authentication algorithm to establish that it is authentic (`a`). Also, the agent could observe a withdrawal from its bank account with the present transaction's id, and this withdrawal may be a correct (`c`) or an incorrect (`i`) amount.

The basic relations on event structures have an intuitive meaning in our setup. An event may exclude the possibility of the occurrence of a number of other events. In our example the occurrence of the event 'request rejected' clearly excludes (and hence is in conflict with) the event 'request granted'. The causality relation is also natural: some events are only possible when others have already occurred. In the example structure, 'forged' only makes sense in a transaction where the

$$\begin{tikzcd}
& \mathtt{a} \arrow[r, squiggly, no head] & \mathtt{f} & & \mathtt{c} \arrow[r, squiggly, no head] & \mathtt{i} \\
\mathtt{r} \arrow[rrr, squiggly, no head] & & & \mathtt{g} \arrow[ull] \arrow[ul] \arrow[ur] \arrow[urr] & &
\end{tikzcd}$$

Fig. 1. An event structure describing our example. The curly lines $\sim$ describe the immediate conflict relation, and pointed arrows the causality relation.

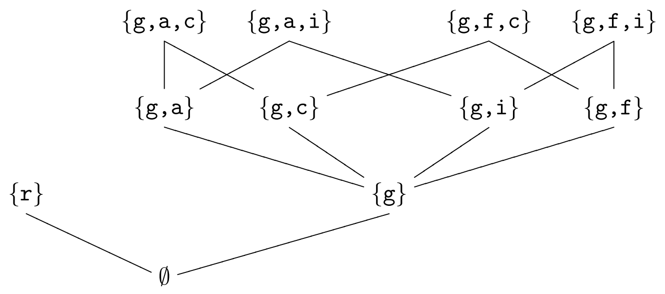

Fig. 2. Configurations of the event structure in Fig. 1.

transfer of e-cash actually did occur (and hence is causally depending on the event 'request granted'). On the other hand the authentication of the e-cash and the correctness of the withdrawal are two independent observations, which can be observed in any order and for which the ordering in time is inessential, which is modelled as independence in the event structure.

Following this intuition, the set of possible executions of an event structure is captured formally by the notion of a configuration, i.e. a set of events which can occur in a particular run of the protocol modelled.

Definition 2 *(Event structure configurations).* A *configuration* of an event structure $ES = (E, \leqslant, \#)$ is a set of events $x \subseteq E$ satisfying the following two properties:

1. $\forall e, e' \in x.\ (e, e') \notin \#$ (x is *conflict free*)
2. $\forall e \in x, e' \in E.\ e' \leqslant e \Rightarrow e' \in x$ (x is *causally closed*)

We write $\mathcal{C}_{ES}$ for the set of configurations of ES.

The configurations of our example from Fig. 1 are given in Fig. 2 along with the inclusion ordering of configurations.

And the idea of modelling behavioural observations is now intuitively very simple. In our example, we imagine a computing agent (e.g. a mobile phone) recording its interactions with a particular bank in terms of the status (read configuration) of all its completed or non-completed runs of the protocol with the bank. Formally, such a recording may be represented as a *decoration* of event structure configurations with the number of registered occurrences (forgetting about the ordering of their occurrences).

Definition 3 *(Decorations).* Let ES be an event structure. A *decoration* d of its configurations is a function $d : \mathcal{C}_{ES} \to \mathbb{N}$.

As an example, imagine we have registered the following 7 protocol outcomes of interacting with a bank

$$\{\mathtt{g}, \mathtt{a}, \mathtt{c}\} \qquad \{\mathtt{r}\} \qquad \{\mathtt{r}\} \qquad \{\mathtt{g}, \mathtt{c}\} \qquad \{\mathtt{g}\} \qquad \{\mathtt{r}\} \qquad \{\mathtt{g}\}$$

then the corresponding decoration of the configurations in Fig. 2 will have the following non-zero elements:

$$d\big(\{\mathtt{r}\}\big) = 3, \qquad d\big(\{\mathtt{g}\}\big) = 2, \qquad d\big(\{\mathtt{g}, \mathtt{c}\}\big) = 1, \qquad d\big(\{\mathtt{g}, \mathtt{a}, \mathtt{c}\}\big) = 1.$$

The idea is now to represent trust in a bank in terms of the expected future outcome of interactions with the bank based on such recordings of interactions from the past. In our example a computing agent will trust a bank depending on the expected outcomes of the "good" outcomes, e.g. the configuration $\{\mathtt{g}, \mathtt{a}, \mathtt{c}\}$. It is beyond the scope of this paper to explain how this very simple idea can be made concrete and useful, but let me briefly indicate a few examples of how we (co-workers Karl Krukow and Vladimiro Sassone) have exploited the idea with some explicit references to Glynn's work.

The work of Glynn and his co-authors on *probabilistic* event structures [13,14] was the foundation for our development of a probabilistic framework for estimating the probabilities of outcomes of interactions. To be more specific, we developed a framework for estimating the probabilities of events in confusion-free event structures, based on Bayesian analysis. For more information see e.g. [8].

In [10] we presented some preliminary work on how to transfer trust-related information from one context to another, a well-known phenomenon from human trust. As an example our trust in a bank's service with respect to its handling of e-cash is likely to be influenced by our trust in the bank's other services. In our setting, a context is represented by a specific protocol, and hence we aimed at formalizing the intuition that behaviour in one event structure may be seen as an indication of a certain behaviour in another event structure. In our formalization, we used Glynn's notion of event structure *morphisms* [16,17] as the underlying tool for specifying a transfer correspondence between configurations.

As a final example, we have addressed the question of a mathematical framework for the formal semantics of the *trust policy languages* used in computational trust. Such languages are intended to be used by computing agents in order to specify their trust based not only on their own observations but also on the trust reported by other (trusted) agents, i.e. recommendations. This gives rise to languages with mutual references in the so-called "web of trust". We proposed a denotational semantics based on standard domain theoretic concepts, but in contrast to Glynn's event structure semantics for process languages [15], our denotations are e.g. decorated finite event structures. For more details see e.g. [9,7].

References

[1] M. Blaze, J. Feigenbaum, J. Lacy, Decentralized trust management, in: IEEE Symposium on Security and Privacy, 1996, pp. 164–173.
[2] M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis, The role of trust management in distributed systems security, in: Springer Lecture Notes in Comput. Sci., vol. 1603, 1999, pp. 185–210.
[3] C. Bryce, P. Couderc, J.M. Seigneur, V. Cahill, Implementation of the SECURE trust engine, in: Proceedings of iTrust'05, 2005, pp. 397–401.
[4] V. Cahill, E. Gray, J.M. Seigneur, C. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G.d.M. Serugendo, C. Bryce, M. Carbone, K. Krukow, M. Nielsen, Using trust for secure collaboration in uncertain environments, IEEE Pervasive Comput. 2 (3) (2003) 52–61.
[5] T. Grandison, M. Sloman, A survey of trust in internet applications, IEEE Commun. Surv. Tutor. 3 (4) (2000) 2–16.
[6] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decis. Support Syst. 43 (2) (2007) 618–644.
[7] K. Krukow, M. Nielsen, Trust structures: denotational and operational semantics, Int. J. Inf. Secur. 6 (2) (2007) 153–181.
[8] M. Nielsen, K. Krukow, V. Sassone, A Bayesian model for event-based trust, Electron. Notes Theor. Comput. Sci. 172 (2007) 499–521.
[9] M. Nielsen, K. Krukow, On the formal modelling of trust in reputation-based systems, in: Springer Lecture Notes in Comput. Sci., vol. 3113, 2004, pp. 192–204.
[10] M. Nielsen, K. Krukow, Transfer of trust in event-based reputation systems, Theoret. Comput. Sci. 429 (2012) 236–246.
[11] M. Nielsen, G. Plotkin, G. Winskel, Petri nets, event structures and domains, Theoret. Comput. Sci. 13 (1981) 85–108.
[12] J. Sabater, C. Sierra, Review on computational trust and reputation models, Artif. Intell. Rev. 24 (1) (2005) 33–60.
[13] D. Varacca, G. Winskel, Distributing probability over nondeterminism, Math. Structures Comput. Sci. 16 (1) (2006) 87–113.
[14] D. Varacca, H. Völzer, G. Winskel, Probabilistic event structures and domains, Theoret. Comput. Sci. 358 (2–3) (2006) 173–199.
[15] G. Winskel, Event structure semantics for CCS and related languages, in: Springer Lecture Notes in Comput. Sci., vol. 140, 1982, pp. 561–576.
[16] G. Winskel, Categories of models for concurrency, in: Springer Lecture Notes in Comput. Sci., vol. 197, 1984, pp. 246–267.
[17] G. Winskel, M. Nielsen, Models for concurrency, in: Handbook of Logic in Computer Science, vol. 4, Oxford University Press, 1995, pp. 1–148.