Structured Occurrence Nets: Incomplete, contradictory and uncertain failure evidence

Brian Randell and Maciej Koutny

TECHNICAL REPORT SERIES

No. CS-TR-1170 September 2009

TECHNICAL REPORT SERIES

No. CS-TR-1170 September, 2009

Structured Occurrence Nets: Incomplete, contradictory and uncertain failure evidence

B. Randell and M. Koutny

Abstract

Occurrence Nets (ONs) are directed acyclic graphs that represent causality and concurrency information concerning a single execution of a system. Structured Occurrence Nets (SONs), consist of multiple ONs associated together by means of various types of formal relationship, and are intended for recording information about either (i) the actual or envisaged behaviour of complex systems, as they interact and evolve, or (ii) evidence that is being gathered and analysed concerning the past behaviour of complex evolving systems. The present report is intended as an addition to our prior work, focussed on the problem of supporting the analysis of evidence about the activities of complex (hardware, software and human) systems that were involved in cybercrime, or are implicated in major accidents, situations in which all that is likely to be to hand is incomplete, contradictory and uncertain evidence.

Bibliographical details

RANDELL, B., KOUTNY, M..

Structured Occurrence Nets: Incomplete, contradictory and uncertain failure evidence
[By] B. Randell, M. Koutny

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2009.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-1170)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-1170

Abstract

Occurrence Nets (ONs) are directed acyclic graphs that represent causality and concurrency information concerning a single execution of a system. Structured Occurrence Nets (SONs), consist of multiple ONs associated together by means of various types of formal relationship, and are intended for recording information about either (i) the actual or envisaged behaviour of complex systems, as they interact and evolve, or (ii) evidence that is being gathered and analysed concerning the past behaviour of complex evolving systems. The present report is intended as an addition to our prior work, focussed on the problem of supporting the analysis of evidence about the activities of complex (hardware, software and human) systems that were involved in cybercrime, or are implicated in major accidents, situations in which all that is likely to be to hand is incomplete, contradictory and uncertain evidence.

About the author

Brian Randell graduated in Mathematics from Imperial College, London in 1957 and joined the English Electric Company where he led a team that implemented a number of compilers, including the Whetstone KDF9 Algol compiler. From 1964 to 1969 he was with IBM in the United States, mainly at the IBM T.J. Watson Research Center, working on operating systems, the design of ultra-high speed computers and computing system design methodology. He then became Professor of Computing Science at the University of Newcastle upon Tyne, where in 1971 he set up the project that initiated research into the possibility of software fault tolerance, and introduced the "recovery block" concept. Subsequent major developments included the Newcastle Connection, and the prototype Distributed Secure System. He has been Principal Investigator on a succession of research projects in reliability and security funded by the Science Research Council (now Engineering and Physical Sciences Research Council), the Ministry of Defence, and the European Strategic Programme of Research in Information Technology (ESPRIT), and now the European Information Society Technologies (IST) Programme. Most recently he has had the role of Project Director of CaberNet (the IST Network of Excellence on Distributed Computing Systems Architectures), and of two IST Research Projects, MAFTIA (Malicious- and Accidental-Fault Tolerance for Internet Applications) and DSoS (Dependable Systems of Systems). He has published nearly two hundred technical papers and reports, and is co-author or editor of seven books. He is now Emeritus Professor of Computing Science, and Senior Research Investigator, at the University of Newcastle upon Tyne.

Maciej Koutny is a Professor of Computing Science in the School of Computing Science at Newcastle University. He received his MSc (1982) and PhD (1984) in Applied Mathematics from the Warsaw University of Technology, Poland. In 1985 he joined the then Computing Laboratory of the University of Newcastle upon Tyne to work as a Research Associate. In 1986 he became a Lecturer in Computing Science at Newcastle, and from 1994 to 2000 he held an established Readership at Newcastle University. His research interests centre on the theory of distributed and concurrent systems, including both theoretical aspects of their semantics and application of formal techniques to the modelling and verification of such systems; in particular, model checking based on net unfoldings. He has also investigated non-interleaving semantics of priority systems, and the relationship between temporal logic and process algebras. Recently, he has been working on the development of a formal model combining Petri nets and process algebras as well as on Petri net based behavioural models of membrane systems. He is a member of the Steering Committee of the International Conferences on Application and Theory of Petri Nets and Other Models of Concurrency(http://www.daimi.au.dk/PetriNets/), and a member of the IFIP Working Group 2.2 on Description of Programming Concepts. He serves as an editor of the LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC), and the Scientific Annals of Computer Science journal. He has been a Visiting Professor at Xidian University, China, University of Evry, France, and University Paris 12, France. His Programme Committee chairmanship includes: ICATPN'01, ACSD'08 and CHINA'08. He is the scientific co-director of the 5th Advanced Course on Petri Nets to be held in 2010.

Suggested keywords

FAILURES
ERRORS
FAULTS
DEPENDABILITY
JUDGEMENT
OCCURRENCE NETS
ABSTRACTION
FORMAL ANALYSIS

Structured Occurrence Nets: Incomplete, contradictory and uncertain failure evidence

Brian Randell and Maciej Koutny

School of Computing Science
Newcastle University
Newcastle upon Tyne, NE1 7RU
United Kingdom
{brian.randell,maciej.koutny}@ncl.ac.uk

Abstract. Occurrence Nets (ONs) are directed acyclic graphs that represent causality and concurrency information concerning a single execution of a system. Structured Occurrence Nets (SONs) [22, 32], consist of multiple ONs associated together by means of various types of formal relationship, and are intended for recording information about either (i) the actual or envisaged behaviour of complex systems, as they interact and evolve, or (ii) evidence that is being gathered and analysed concerning the past behaviour of complex evolving systems. The present report is intended as an addition to [32], focussed on the problem of supporting the analysis of evidence about the activities of complex (hardware, software and human) systems that were involved in cybercrime, or are implicated in major accidents, situations in which all that is likely to be to hand is incomplete, contradictory and uncertain evidence.

Keywords: failures, errors, faults, dependability, judgement, occurrence nets, abstraction, formal analysis.

1 Introduction

The concept of a 'structured occurrence net' (SON), which as its name implies is based on that of an 'occurrence net', was introduced in [32] and described in more detail in [22]. Occurrence Nets (ONs) are directed acyclic graphs that represent causality and concurrency information concerning a single execution of a system. SONs consist of multiple ONs associated together by means of various types of formal relationship, and are intended for recording information about either (i) the actual or envisaged behaviour of complex systems, as they interact and evolve, or (ii) evidence that is being gathered and analysed concerning the past behaviour of complex evolving systems. One particular possible application of SONs, the focus of the present paper, is support for the analysis of evidence about the activities of complex (hardware, software and human) systems that are involved in cybercrime, or are implicated in major accidents.

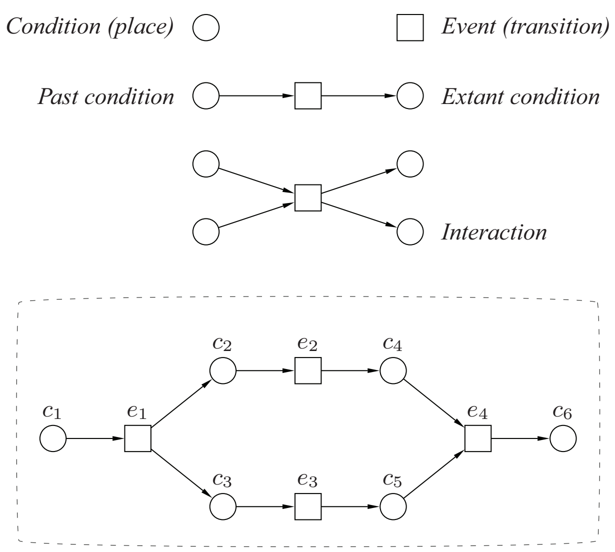


Fig. 1. Basic notation (top) and an occurrence net (bottom). We adopt the graphical convention that, in order to distinguish them from ordinary occurrence nets, structured occurrence nets which contain two or more component occurrence nets are shown surrounded by a solid line bounding box.

ONs can portray the (alleged) past and present state of affairs, in terms of places (conditions, represented by circles), transitions (events, represented by squares) and arrows (each from a place to a transition, or from a transition to a place, representing (alleged) causality) — see the notation shown in Figure 1. (This figure, and a number of others used in the present paper, come from [22].) The ON formalism is well-established and used extensively, and is well-supported by tools for system validation and synthesis [6, 14–18]. For simple nets, an actual graphical representation suffices. However, in the case of complex nets, these are better represented in some linguistic or tabular form. ONs could be 'generated' by executing Petri nets representing computing systems, but they can in fact be used to record the execution of any (potentially asynchronous) process — carried out by computer systems and/or humans — no matter what notation or language might be used to define it. In fact, various other graphical notations similar to ONs can be found in both the hardware and the software design worlds, e.g., strand spaces [35], signal diagrams [24] and message sequence charts [27].

Our original interest in what we now term SONs arose out of a wish to gain greater understanding of basic dependability concepts, and in particular of 'fault-error-failure chains' in evolving complex systems, and the notion of 'failure judgements'. Now, however, our main motivation for defining and exploring the formal properties of SONs is our belief that, due to their structuring, SONs can be used to achieve a significant reduction of the cognitive and computational difficulties that arise in using ONs for modelling and analysing the behaviour of complex evolving systems. (In fact SONs provide yet another case of the 'divide and conquer' approach to complexity reduction, and thus should help to limit state space explosion in analysis tools.)

In [22] we discussed how automatically-generated fully-detailed SONs (produced either from actual systems, or from models of intended systems), could be used to extend the capabilities of (i) existing ON-based model-checking approaches to system validation [14, 31], and (ii) existing tools for the automated synthesis of systems (e.g., VLSI designs) from exemplar ONs [15]. However, we also discussed the use of SONs that had been created after the fact from whatever evidence of a system's actions was available to assist the task of analysing (accidental or malicious) failures in large complex evolving systems.

A possibility that we find particularly interesting is that of using such SONs to facilitate the analysis of failures in systems involving software, hardware and people. For example, this could be in order in order to identify those responsible for a complex (cyber)crime, or to determine the causes(s) of a major accident, situations in which all that is likely to be to hand is incomplete, contradictory and uncertain evidence. It is the problem of support for such analysis tasks that the present paper concentrates on.

In the appendix we outline a formalisation of some of the notions described in the main body of this paper.

2 Relations between ONs

In [22] we defined two classes of relations between ONs, namely ordered relations (abstraction and behaviour), and unordered relations (system interaction, retention and judgement). Abstraction, which has two basic forms (spatial and temporal), provides means of abbreviating/condensing a perhaps very large ON. In its simplest form spatial abstraction involves identifying a subgraph consisting of a set of conditions (that are not causally linked within this subgraph but instead are potentially co-existing) in an ON, and representing this subgraph by a single condition in a more abstract ON. Similarly basic temporal abstraction involves identifying a subgraph consisting of a set of events (and any intervening conditions) that are causally linked within this subgraph in an ON, and relating this to a single event in a more abstract ON. In general, however, abstraction involves elements of both spatial and temporal abstraction.

Figure 2 shows some of the spatial (s) and temporal (t) relations involved in an example in which the activities of two systems, represented by ONs with differently-shaded conditions and events, have been in part abbreviated and in part merged into a single activity. This single abstract activity thus hides the intercommunication that is shown as taking place between the original two activities. (Temporal abstraction is in general somewhat tricky because of the fundamental requirement — arising from the intention of representing causality — that all ONs must be acyclic. Such problems of temporal abstraction, which are dealt with in Section 5 of [22], are not discussed further here.)

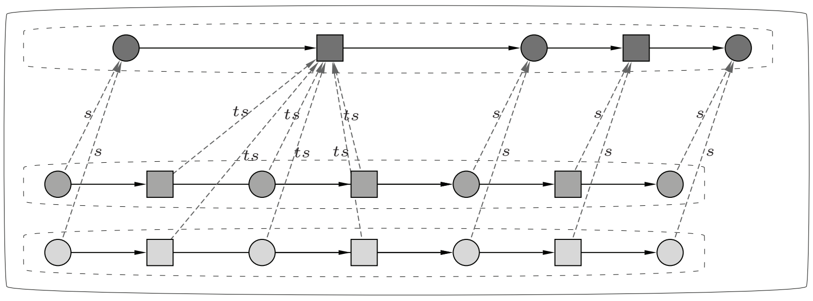


Fig. 2. Combined spatial and temporal abstraction — the 'spatially abstracts' and 'temporally abstracts' correspondences are indicated by groups of s-labelled and t-labelled edges, respectively.

The behaviour relation is one that we introduced in [32] as a result of our belated realisation that the concepts of 'system' and 'state' are not separate, but just a question of viewpoint, so that (different related) ONs can represent a system and its states using the same symbol — a 'place'.

Specifically, a connected subgraph comprised just of states and events of a given system is explicitly related to a single state of a higher level activity, an activity which is portraying this given system's existence, for example recording its creation, continued existence and termination.

This type of relation is of importance in situations where systems evolve, or suffer modifications, in ways that could affect their behaviour, as it allows the behaviour of a system to be related to that of (particular states of) the evolving system itself. (See the SON portrayed in Figure 3, which shows, in its upper level, the history of two systems, each of which suffers a modification, and in its lower level ONs representing the activities that the initial unmodified, and subsequently the modified, versions of these systems give rise to. Note that the same shading is used for the higher and the lower level view of each system.)

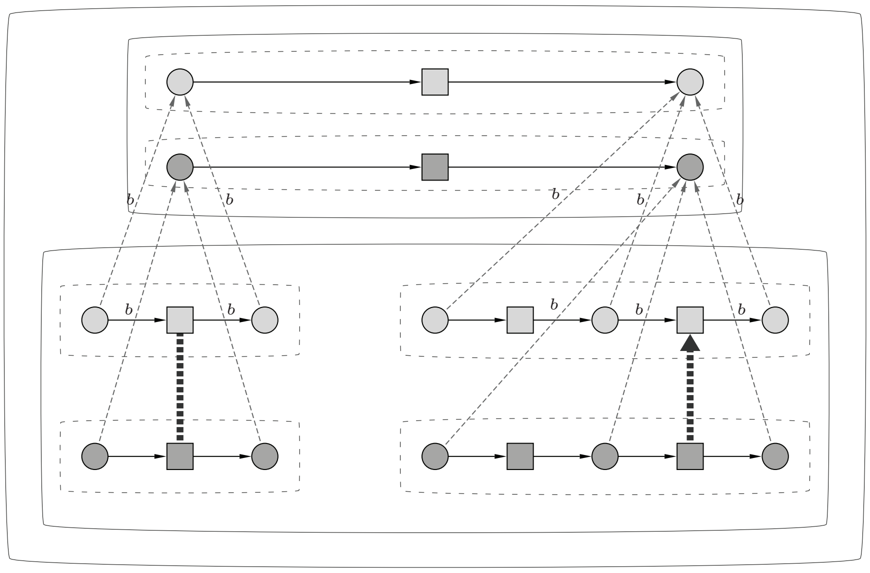


Fig. 3. The behaviour of a pair of systems, while they each suffer one modification.

Being ordered relations, the abstraction and behaviour relations must not result in any cycles — one cannot for example use spatial abstraction to model a system containing itself as a component! (This acyclicity requirement is additional to that concerning causality.) On the other hand, such acyclicity restrictions do not apply to the other relations between

ONs that are defined in [22], namely system interaction, retention and judgement, since they are in general unordered.

Two forms of system interaction relation are defined — causal and synchronous — that relate events of distinct ONs (i.e., the activities of separate systems). The former relation means that an event in one ON is a causal predecessor of another event in another ON (information flow was unidirectional), while the latter means that two events have been executed synchronously (information flow was bidirectional). Such system interaction relations are represented by thick dashed arcs in the SON shown in Figure 4. (These thick dashed arcs and edges are abstractions of the details that represent such system interactions when one describes such a compound system by a single large unstructured ON.)

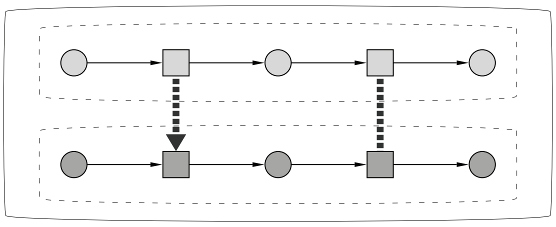


Fig. 4. Causal and asynchronous system interaction.

The retention relation arises as a result of one system storing (via a retain operation) information (regarding conditions, events and/or links) that is, or purports to be, information about the activity of another system. (Such information will be held until an appropriate discard operation takes place.) One possible reason for retaining such data is to provide means of error recovery and back-up to this other system. However, here we concentrate on issues related to data that has been retained (or perhaps generated or supplemented by inference and guesswork) as potential evidence for purposes of post-hoc failure analysis. For example, and illustrating the unordered nature of the retention relation, a cyber-crime scenario might be modelled as involving several systems (e.g., an investigative agency and two rival crime syndicates), each of which is, amongst its other activities, attempting to obtain, retain and exploit information about the activities of the other two systems in order to disrupt them.

We view such failure analysis as being performed by a system that is acting as a judgment system, such as a judge in charge of a fatal accident enquiry, or the detective team conducting a major criminal investigation. (In other situations the judgements might, for example, be made by (i) a technician equipped with an authoritative system specification, or even (ii) a fully-automated checking facility.) Figure 5 from [22] illustrates a simple example of such a situation by showing (a) some fragmentary evidence, and (b) the judgement activity that obtained and retained this fragmentary evidence, together with the unknown, and perhaps unknowable, actual SON that gave rise to this evidence.

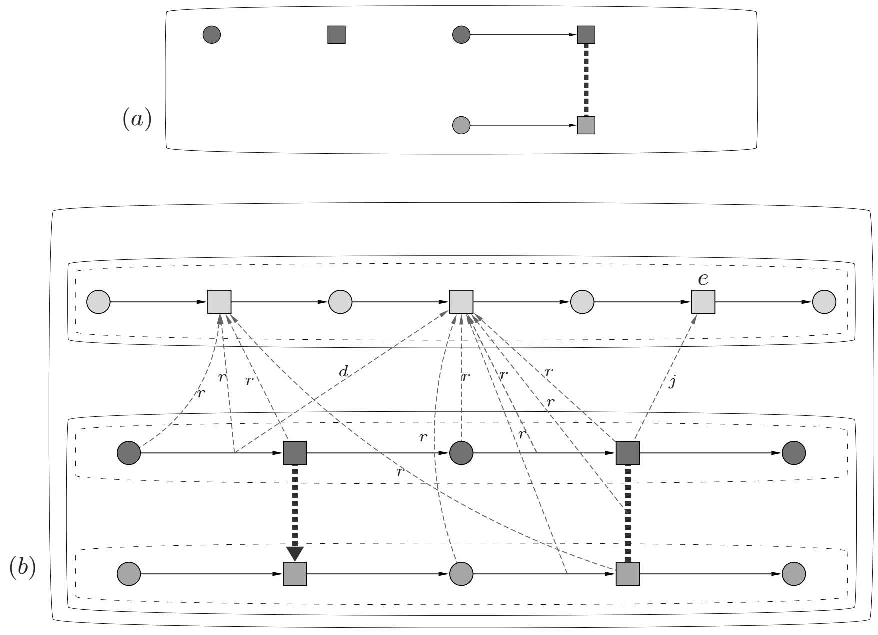


Fig. 5. An evidence, in the form of SON fragments (a); and the judgement SON from which the retained evidence could have been obtained (b).

The judgement relation arises as a result of an act of judgement, which typically will involve an interaction between the judgement system and the system being judged, so that the future course of action of the latter can be appropriately affected (e.g., release or imprisonment, in the case of legal judgements). By regarding failures simply as events that are identified as such by judgement systems, we can allow for the fact that (i) depending on the viewpoint adopted, the same behaviour may or may not be regarded as a failure, and that (ii) failure judgements may themselves turn out to be judged later by some further system to have been incorrect — a situation that is allowed for in the legal world with its hierarchy of courts, the lower levels of which might have their judgements overturned by higher courts. Similarly, by regarding data retention as an act performed by a system (perhaps as part of an investigative process), one can allow for the possibility that such a data retention system could fail (in the eyes of some judgement system) by, for example, losing or interfering with evidence. Thus such concepts as 'chain of custody' and 'provenance of evidence', and their related problems, can readily be modelled.

As we have illustrated, unlike the detailed SONs that are created from system models, e.g., as part of a model-checking exercise, any such retained SON is very likely to be fragmentary. It nevertheless has to be consistent with the various theorems that are stated and proved in [22], e.g., regarding acyclicity, a fact that could and should be exploited by the experimental tool that we hope to build for supporting the storage, manipulation and analysis of SONs. (Such a SON tool is likely to be based on existing tools for ONs [31].) In particular, the tool should be able not just to enforce acyclicity, but also to assist by drawing attention to, and facilitating the filling in of, gaps between SON fragments.

Such a tool is best regarded as constituting a somewhat general infrastructure (in fact essentially a form of database management system), which could support system evaluation using complete SONs, system synthesis from exemplar SONs, or failure analysis of possibly fragmentary SONs, via appropriate specialised applications. (The applications would typically make use of additional information — in effect annotations of places, links, conditions and/or relations - held in a given SON database.) The infrastructure tool would embody fairly general facilities for assessing and reporting on the completeness and consistency of a database representing a given SON, using algorithms based on the various theorems and propositions given in [22]. But the task of creating the representation of the SON and of performing specialised analyses and manipulations of it would be the responsibility of such specialised applications, using infrastructure-provided facilities for:

– visualising ONs;
– creating new (empty) ONs,

– adding or deleting single events, or conditions (with their links);
– maintaining and providing access to event, condition, link, and relation annotations;
– establishing and destroying relations between ONs; and
– destroying complete ONs (provided they are not related to any other occurrence nets).

Validating such a database (w.r.t. the rules concerning well-formedness of sets of related ONs) might be done incrementally, as data is added, so that the database is always (syntactically) valid, or could be a process that is done occasionally, and followed by attempts at resolving any detected errors. There would of course also be a need to validate the semantics of the database, taking into account all annotations — an application-specific task that could be aided by appropriate visualisation tools. The result could we believe provide useful support for the 'knowledge-based' modelling technique that has been developed for formulating plausible scenarios worthy of further police investigation [13] or the 'Why-Because' causal analysis scheme designed for aircraft accident analyses [23] and also used for security incident analyses [34].

However to provide really useful infrastructure support for applications related to very complex accident and criminal investigations, we believe that it is necessary to extend the SON concept. This would be in order to provide explicit support for the representation and analysis of contradictory and uncertain, as opposed to merely fragmentary, evidence. These issues are the subject of the following main sections of this paper.

3 Contradictory Evidence

There is an already-existing extension to the basic ON notation, namely the 'barb' [20], used for an event that could have occurred, given the condition(s) that existed, but which did not — see Figure 6, where barbs are represented by a box containing a circle. In some cases such barbs will have required the coincident existence of several conditions.

We now further extend the ON notation, by building on the barb concept, so as to allow the simultaneous modelling of multiple alternative scenarios that could have occurred in particular states, when there is insufficient evidence to indicate which particular scenario actually occurred. This is illustrated in Figure 7, which uses letters A and B to distinguish between two alternative system activities. Specifically, it shows

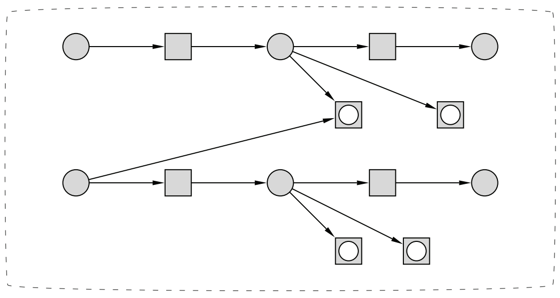


Fig. 6. Events that could have occurred — but didn’t.

that two alternative ways in which the given system’s final state might have come about have been envisaged and are being represented in a single ON. (This ON can be regarded as a combination of the two alternative ONs.)

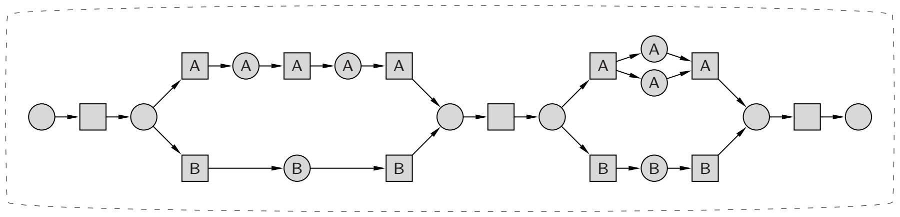


Fig. 7. Two alternative scenarios.

Note that such alternative activities are indicated by multiple causal links emanating from a condition, in contrast to parallel activity, whose initiation is marked by multiple causal links emanating from an event (see Figure 1). Evidently, alternative chains of activity within a single ON represent happenings in what are in effect different ‘worlds’, and it would not make sense to have any interactions between such worlds. Figure 8 shows examples of possible causal links: ones that are allowed between parallel activities from potentially the same world, and ones that are disallowed between the alternative (A and B) worlds. (An infrastructure tool supporting SONs could and should police these rules, which are somewhat reminiscent of those concerning atomicity in transaction processing.)

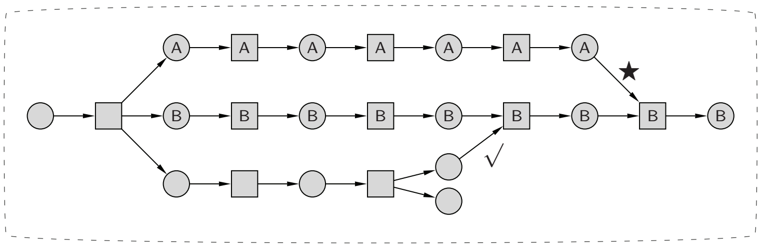


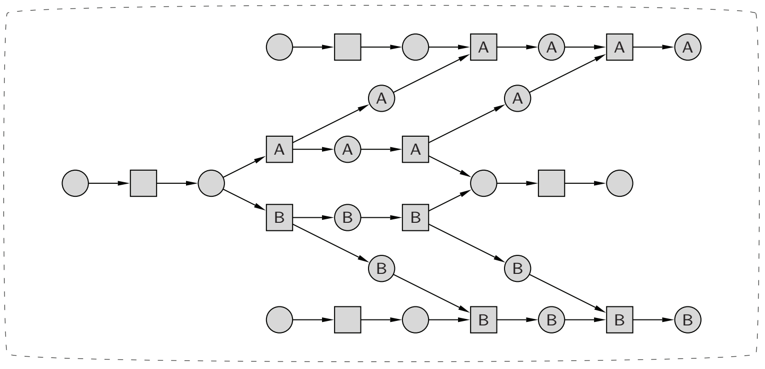
Fig. 8. Examples of allowed (√) and disallowed (★) causal links.

There is also the very real possibility of contradictory or inadequate evidence concerning relations between ONs, so motivating the incorporation of multiple alternative relations within a single SON, and therefore in some cases there being multiple alternative ONs. Figure 9(a) shows an ON representing the activities of three systems, in which there are alternative causal links from the middle activity to either the upper or the lower activity. The pair of alternative events lead on not only to links to other activities, but also to links that, through coming back to a single condition, show that no matter which alternative is chosen, the middle activity continues in its own right. (As in the previous figure, letters are used to distinguish alternative consequences.) Figure 9(b) is an equivalent SON, in which the three activities are simply regarded as three separate ONs and some of their parts have been subjected to temporal abstraction, the details of whose interactions are hidden, being instead represented by (alternative) system interaction relations — the fact that these relations are alternatives is again implied by the letter markings.

There may also, for example, be contradictory or inadequate evidence linking systems to their behaviours, leading to the inclusion of alternative behaviour relations in a SON. Figure 10(a) shows behaviour that has been identified as either that of system A or that of system B (which underwent a modification during this period); Figure 10(b) shows that two alternate behaviours (A and B) have been ascribed to a given system, over a period when it suffered a modification.

Similarly, particularly when there has been inadequate attention paid to the problem of the evidential ‘chain of custody’, one might have a situation in which there are differing rival sets of evidence about a given activity, and hence a need to incorporate alternative retention relations. Re-

(a)



(b)

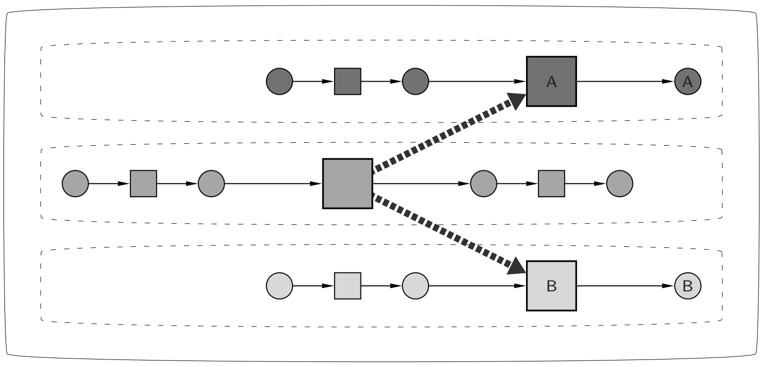


Fig. 9. Alternative interactions, in an ON (a), and in an equivalent SON (b).

garding spatial and/or temporal abstraction relations in general, multiple alternative abstractions of the same ON could also sometimes be useful, and so could give rise to contradictory (abstract) accounts of what actually happened — for example different judges may take differing views, i.e., construct different abstractions of, the same evidence. (The opposite also holds: two or more differing models of a given activity might have a single common abstraction.)

4 Uncertain Evidence

As we have indicated, during post hoc inquiries the full details of the activity of an actual complex system under investigation will almost certainly not be known — all that will be available is the evidence that has

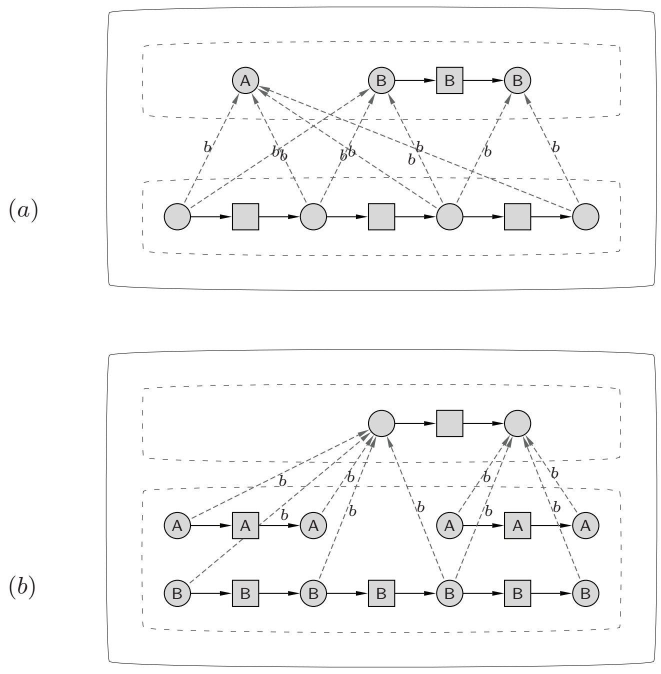


Fig. 10. Alternative behaviour relations.

been obtained (or surmised) by this activity's judges, i.e., in all probability just a fragmentary set of inaccurate and perhaps contradictory ONs. Turning these into a plausible more 'complete' (i.e., connected and self-consistent) SON will involve adding events, conditions, links and relations (based either on evidence or guesswork). Moreover, as we indicated above, it might well also be wished to include various sets of alternative behaviours within the SON, at least until it is decided what actually happened so that all but one of each set of alternative behaviours can be discarded. (One could go further and also introduce and use a further type of relation 'provides evidence for the existence of' — between an element of a SON and an extant state of another SON that documents this evidence and which is buttressed by 'chain of custody' activity recording the provenance of this evidence.)

It would be useful therefore to be able to annotate particular events, conditions, links and relations with some form of probability estimate — indicating the current degree of certainty a judge has about the accuracy of their representation. It is therefore attractive to provide means by which a modeller can specify estimates of the relative likelihood of the different alternatives that are modelled, in fact by associating probabilities with the links or interaction relations leading to the starting events of alternatives. These means could be very simple, e.g., numerical probability estimates, on say a five point scale, or actual probability distributions and their parameters.

Once such probability estimates have provided for each such starting link or relation, then an infrastructure tool could readily calculate estimates for all uncertain events, conditions, links and relations. However any events, conditions, links and relations that are not within any of the separate worlds that have been ‘created’ by alternative links or interaction relations will be regarded as certain events, for which no probability estimate need be recorded. This is illustrated in Figure 11, in which calculated (as opposed to assigned) estimates of link probabilities are shown in italics. (As before states and events within alternative activities are given distinguishing labels.)

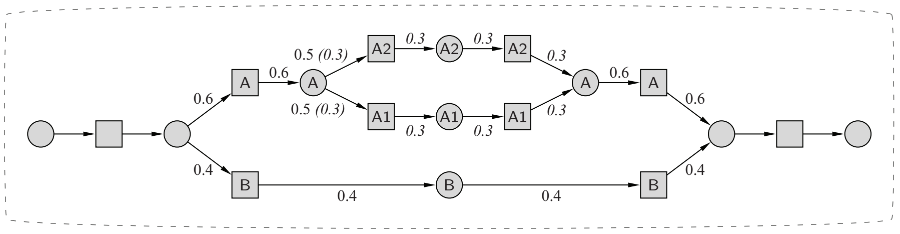


Fig. 11. Uncertainty annotations.

Figure 12 shows such annotations augmenting the SON that was shown in Figure 9(b).

There could of course be multiple judges who, even if they agree on the modelling of complex activity, might have different estimates of the relative probabilities of any alternative sub-activities involved. Thus there would be merit in associating sets of probability estimate annotations with particular judges.

It is worth pointing out that concepts essentially equivalent to the above notions of associating probability estimates and evidential information with particular linked events, and of providing alternative event

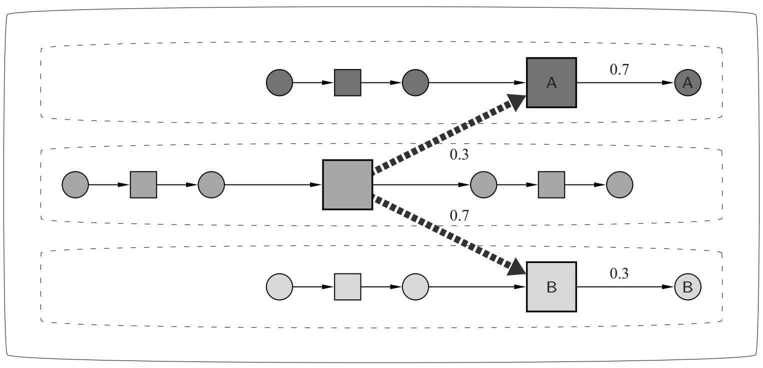


Fig. 12. Uncertainty annotations for alternative interaction relations.

sequences, are in fact already supported in some lineage-linked database systems (systems used by genealogists to record and analyse family trees) such as [10, 36]. (Family trees are by no means always actual trees because of events such as intermarriage between cousins; they are in fact acyclic directed graphs, and could readily be represented as ONs.) The more advanced lineage-linked database systems provide also facilities for such tasks as merging incomplete inaccurate family trees or sections of family trees, and for performing various types of validity check (though they do not provide anything comparable to the structuring facilities we have defined based on inter-ON relations).

For example, one important attribute of events in lineage-linked databases is that of 'date of occurrence' (e.g., of births, marriages and deaths). When this attribute is recorded for a reasonable fraction of the events in the database, then it is possible to calculate useful date ranges for any undated events, and to provide some useful automated consistency checking [5]. This can lead to the identification of inaccurate date attributes, or to questioning the accuracy of one or more links in the database. Support for the storing and analysis of such temporal attributes would seem to be an appropriate facility also for structured ONs being used for conjectural failure analysis.

5 Conclusions

To the best of our knowledge the basic ideas that we described in [32] and [22] for structuring occurrence nets are novel, and we have some confidence that they could facilitate the production of significantly-improved tools for modelling and analysing complex software and hardware systems, a topic with which we are rather familiar. However we have here attempted to identify an appropriate set of extensions to the basic SON concepts aimed at providing reasonably general means of representing a large body of incomplete, contradictory and uncertain records of complex activities. As indicated the type of activities we particularly have in mind involve hardware, software and humans, and give rise to voluminous detailed records that have to be investigated after various kinds of system failure (due to either accidental or malicious faults).

Our knowledge of this area is second-hand — but we have studied the survey of the numerous existing modelling notations for activity analyses and event reconstruction, for example, see [12], as well as the several accounts of particular investigation techniques referenced earlier [13, 23, 34]. We note the cautionary comments in, for example, [25, 26] about the limitations of activity (or 'incident') analysis on its own as a technique of accident investigation and prevention, and accept the necessity for system analyses (e.g., concerning a system's safety control structures) as well as activity analyses in this arena at least. However, it seems clear that in very complex failure situations activity analysis is likely to remain an important and challenging task; it is our belief that our formally-defined concept of Structured Occurrence Nets has significant potential advantages over the various other notations currently in use in support of such analyses, especially in situations in which very large amounts of evidence have to be recorded and analysed (e.g., in major accident and crime investigations).

Our belief is in part due to the various SON relations we define and discuss in [22], in part to the scheme we describe above for showing contradictory (alternative) activities and relations within a SON. This scheme is aimed at both ease of use and of (automated) consistency and (syntactic) validity checking. These aims have motivated the method we propose for associating explicit indications of any uncertainties about a SON's accuracy to a structured scheme of alternative activities and relations.

However, it is evident that the practical usability of this scheme in, and indeed the applicability of the basic SON concept to, post hoc investigations of complex accidents or crimes is dependent on the provision of adequate tool support. Thus we regard our work to date on SONs as just a starting point for the development of an infrastructure tool (perhaps incorporating existing occurrence net tools). As mentioned earlier, such an infrastructure tool could be viewed as a database management system that is specialised towards databases that record causality and asynchrony. But the infrastructure tool would otherwise be rather general in intent, and would we hope enable practical experiments related to computer-aided support to people involved in complex criminal investigations and safety enquiries, as well as to computer design activitie, such as system verification via model checking, that rely on having a fully-detailed model of the system.

References

1. A.Avizienis, J.-C.Laprie, B.Randell and C.Landwehr (2004). Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. on Dep. and Sec. Comp.* 1, 11–33
2. P.Baldan, T.Chatain, S.Haar and B.König (2008). Unfolding-Based Diagnosis of Systems with an Evolving Topology. Proc. of *CONCUR'08*, LNCS 5201, 203–217
3. E.Best and R.Devillers (1988). Sequential and Concurrent Behaviour in Petri Net Theory. *Theoretical Computer Science* 55, 87–136
4. E.Best and B.Randell (1981). A Formal Model of Atomicity in Asynchronous Systems. *Acta Informatica* 16, 93–124
5. V. Brox. GEDCOM Estimator, 2001. http://home.no.net/gedcom/
6. J.Esparza and K.Heljanko (2008). *Unfoldings: A Partial-Order Approach to Model Checking*. Springer
7. J.Esparza, S.Römer and W.Vogler (2002). An Improvement of McMillan's Unfolding Algorithm. *Formal Methods in System Design* 20, 285–310
8. J.Foreman (1996). Product Line Based Software Development. Proc. of *Future Challenges*, Proceedings of the Software Technology Conference
9. P.Gladyshev and A.Patel (2005). Formalising Event Time Bounding in Digital Investigations. *International Journal of Digital Evidence* 4
10. L.H.Hoffman (2003). *Getting the Most Out of the Master Genealogist*. Gateway Press
11. A.W.Holt, R.M.Shapiro, H.Saint and S.Marshall (1968). Information System Theory Project. Report RADC-TR-68-305, US Air Force, Rome Air Development Center
12. C.W.Johnson (2003). *Failure in Safety-Critical Systems: A Handbook of Accident and Incident Reporting*. University of Glasgow Press
13. J.Keppens and B.Schafer (2006). Knowledge Based Crime Scenario Modelling. *Expert Syst. Appl.* 30, 203–222
14. V.Khomenko and M.Koutny (2007). Verification of Bounded Petri Nets Using Integer Programming. *Formal Methods in System Design* 30, 143–176
15. V.Khomenko, M.Koutny and A.Yakovlev (2006). Logic Synthesis for Asynchronous Circuits Based on STG Unfoldings and Incremental SAT. *Fundamenta Informaticae* 70, 49–73

16. V.Khomenko (2008). Derivation of Monotonic Covers for Standard-C Implementation Using STG Unfoldings. Proc. of *ASYNC 2008*, IEEE Computer Society, 141–150
17. V.Khomenko (2009). Efficient Automatic Resolution of Encoding Conflicts Using STG Unfoldings. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 17, 855–868
18. V.Khomenko, A.Madalinski and A.Yakovlev (2008). Resolution of Encoding Conflicts by Signal Insertion and Concurrency Reduction Based on STG Unfoldings. *Fundamenta Informaticae* 86, 299–323
19. H.C.M.Kleijn and M.Koutny (2004). Process Semantics of General Inhibitor Nets. *Information and Computation* 190, 18-69
20. J.Kleijn, M.Koutny and G.Rozenberg (2006). Towards a Petri Net Semantics for Membrane Systems. Proc. of *WMC'05*, LNCS 3850, 292–309
21. D.Koppad, D.Sokolov, A.Bystrov and A.Yakovlev (2006). Online Testing by Protocol Decomposition. Proc. of *IOLTS'06*, IEEE CS Press, 263–268
22. M.Koutny and B.Randell (2009). Structured Occurrence Nets: A formalism for aiding system failure prevention and analysis techniques. Report TR-1162, Scool of Computing Science, Newcastle University
23. P.B.Ladkin (2000). Causal Reasoning about Aircraft Accidents. Proc. of *SAFECOMP 2000*, LNCS 1943, 344–360
24. S.Lenk (1994). Extended Timing Diagrams as a Specification Language. Proc. of *European Design Automation*, IEEE Computer Society Press, 28–33
25. N.Levenson (2004). A New Accident Model for Engineering Safer Systems. *Safety Science* 42
26. N.Levenson (2008). Applying Systems Thinking to Analyze and Learn from Events. Proc. of *NeTWorK 2008*, Event Analysis and Learning from Events 42
27. S.Mauw (1996). The Formalization of Message Sequence Charts. *Computer Networks and ISDN Systems* 28, 1643–1657
28. K.L.McMillan (1995). A Technique of State Space Search Based on Unfoldings. *Formal Methods in System Design* 6, 45-65
29. S.Melzer and S.Römer (1997). Deadlock Checking Using Net Unfoldings. Proc. of *CAV'97*, LNCS 1254, 352–363
30. P.M.Merlin and B.Randell (1978). State Restoration in Distributed Systems. Proc. of *FTCS-8*, IEEE Computer Society Press, 129–134
31. I.Poliakov, V.Khomenko and A.Yakovlev (2009). WORKCRAFT — A Framework for Interpreted Graph Models. Proc. of *PETRI NETS 2009*, LNCS 5606, 333–342
32. B.Randell and M.Koutny (2007). Failures: Their Definition, Modelling and Analysis. Proc. of *ICTAC'07*, LNCS 4711, 260–274
33. G.Rozenberg and J.Engelfriet (1998). Elementary Net Systems. Proc. of *Advances in Petri Nets. Lectures on Petri Nets I: Basic Models*, LNCS 1491, 12–121
34. A.Tarigan and I.M.Wiryana (2003). *Analyzing Security Incidents with Why Because Analysis*. Network and Distributed System Working Group, Faculty of Technology, Bielefeld University, Germany
http://antareja.rvs.uni-bielefeld.de/avinanta/Publication/WBASec/dnsincwba-3.ps
35. F.J.Thayer, J.C.Herzog and J.D.Guttman (1999). Strand Spaces: Proving Security Protocols Correct. *Journal of Computer Security* 7, 191–230
36. The Master Genealogist, Wholly Genes Corp.
http://www.whollygenes.com/Merchant2/merchant.mvc?screen=TMG
37. http://www.rail-reg.gov.uk/upload/pdf/incident-ladbrokegrove-ladbroke-optim.pdf

A Alternate occurrence nets

In what follows, we assume the notation introduced and explained in [22]. The only new concept is that of a set of *alternative views (or worlds)*, $AV \overset{\text{df}}{=} \{\mathsf{A}_1, \ldots, \mathsf{A}_\ell\}$.

Definition 1 (alt-occurrence net AON). *An* alternative occurrence net *is a tuple* $\textsc{aon} \overset{\text{df}}{=} (C, E, F, \vartheta)$ *where* $C \neq \varnothing$ *and* E *are finite disjoint sets of respectively* conditions *and* events*,* $F \subseteq (C \times E) \cup (E \times C)$ *is a* flow relation*, and* $\vartheta : C \cup E \cup F \to 2^{AV} \setminus \{\varnothing\}$ *is a mapping, such that, for every* $\mathsf{A} \in AV$*:*

$$\textsc{aon} \downdownarrows \mathsf{A} \overset{\text{df}}{=} (C \downdownarrows \mathsf{A}, E \downdownarrows \mathsf{A}, F \downdownarrows \mathsf{A})$$

is an occurrence net, where we define, for $X = C, E, F$*,* $X \downdownarrows \mathsf{A}$ *to be the set of all* $x \in X$ *such that* $\mathsf{A} \in \vartheta(x)$*.*
Moreover, we assume that, for every $\mathsf{A} \in AV$*:*

$$\begin{aligned} &\mathit{Init}_{\textsc{aon} \downdownarrows \mathsf{A}} \subseteq \{c \in C \mid \neg\exists e \in E : \ (e, c) \in F\} \\ &\mathit{Fin}_{\textsc{aon} \downdownarrows \mathsf{A}} \subseteq \{c \in C \mid \neg\exists e \in E : \ (c, e) \in F\} \ . \qquad \diamond \end{aligned}$$

The above definition incorporates the basic principles behind our modelling of alternative histories of systems. In essence, what we want to express is that a history with alternatives is an overlay of individual sound histories, each such individual history being tagged by a symbol A in AV as specified by the mapping ϑ. Crucially, what is allowed is that the same element may be tagged by multiple tags, meaning that it is part of many views. In an extreme case, when the element is tagged by all the tags in AV, it becomes a 'certain' element which has been verified to exist by all the views represented in the alternative occurrence net.[1] Hence $\textsc{aon} \downdownarrows \mathsf{A}$ is the behavioural report of what has happened from the point of view of one of the possible alternatives $\mathsf{A} \in AV$. The two conditions at the end of the definition mean that the views are not completely arbitrary and, in particular, a non-initial condition in one view should not be an initial condition in another view.

[1] Note that in the main part of this paper, e.g., in Figure 9, we simply left such 'certain' elements un-tagged.

Since we are now dealing with multiple accounts of a system's evolution, it is natural to define the initial and final situations as sets:

$$\begin{aligned} Init_{\mathrm{AON}} &\overset{\mathrm{df}}{=} \{Init_{\mathrm{AON} \downdownarrows \mathsf{A}_1}, \ldots, Init_{\mathrm{AON} \downdownarrows \mathsf{A}_\ell}\} \\ Fin_{\mathrm{AON}} &\overset{\mathrm{df}}{=} \{Fin_{\mathrm{AON} \downdownarrows \mathsf{A}_1}, \ldots, Fin_{\mathrm{AON} \downdownarrows \mathsf{A}_\ell}\} \, . \end{aligned}$$

In the same way, a cut of AON is any of the cuts of the occurrence nets AON $\downdownarrows \mathsf{A}_i$.

A *step execution* of the alt-occurrence net AON as in Definition 1 is a sequence $\chi \overset{\mathrm{df}}{=} D_0\, G_1\, D_1 \ldots G_n\, D_n$ ($n \geq 0$), where each D_i is a set of conditions and each G_i is a possibly empty set of events (called a *step*), for which there is $\mathsf{A} \in AV$ such that χ is a step sequence of the occurrence net AON $\downdownarrows \mathsf{A}$. We also say that the step execution χ *leads* to the cut D_n, and that D_n is *reachable*.

The following result can be seen as a validation of the soundness of the notions just introduced.

Theorem 1. *Given the step execution as above, we have that: each D_i is a cut of* AON*; no event occurs more than once; and $D_n \in Fin_{\mathrm{ON}}$* iff *each event of one of the occurrence nets* AON $\downdownarrows \mathsf{A}$ *occurs in the execution. Moreover, each cut of* AON *can be reached from one of the initial cuts through some step execution, and there is a step execution involving all the events in each occurrence net* AON $\downdownarrows \mathsf{A}$. $\diamond$

Basically, AON is sound in the sense of obeying some natural temporal properties as well as testifying to the fact that AON does not contain redundant parts. We also have a complete characterisation of global states reachable from the default initial states — these are all the cuts of AON. In practical terms, the latter means that we can verify state properties of the computations captured by AON by running a model checker which inspects all the cuts.

B Alternate communication SONs

The AONs introduced in the previous section are based on occurrence nets which are flat records of concurrent execution histories. In [22] we proposed to structure such histories through different classes of structured occurrence nets, each such class being aimed at dealing with one of the complex relationships involved in dynamic evolutions. We will now show how one can enhance the fundamental concept of communication structured occurrence net with the notion of alternative execution.

Definition 2 (communication ASON). *Let* $\mathrm{AON}_i \overset{\mathrm{df}}{=} (C_i, E_i, F_i, \vartheta_i)$ *for* $i = 1, \ldots, k$ *be alternative occurrence nets (*$k \geq 1$*) with* disjoint *sets of nodes.*
Let κ *and* σ *be two relations (*σ *being symmetric) comprising pairs of events coming from different* AON_i*'s, and* $\vartheta' : \kappa \cup \sigma \rightarrow 2^{AV} \setminus \{\varnothing\}$.
An alternative communication structured occurrence net *is a tuple*

$$\mathrm{AC\_SON} \overset{\mathrm{df}}{=} (\mathrm{ON}_1, \ldots, \mathrm{ON}_k, \kappa, \sigma, \vartheta_1 \cup \ldots \vartheta_k \cup \vartheta)$$

such that, for every $\mathsf{A} \in AV$*,*

$$\mathrm{AC\_SON} \downdownarrows \mathsf{A} \overset{\mathrm{df}}{=} (\mathrm{ON}_1 \downdownarrows \mathsf{A}, \ldots, \mathrm{ON}_k \downdownarrows \mathsf{A}, \kappa \downdownarrows \mathsf{A}, \sigma \downdownarrows \mathsf{A})$$

is a communication structured occurrence net $\diamond$

Similarly as in the case of AONs, the definition takes a number of CSONs and overlays them into a single structure. Again, this may result in some of the elements becoming 'certain', and some only being parts of a subset of potential views on the execution history represented by ACSON.

For the ACSON as above, cuts and step executions need to be redefined, by taking into account the fact that there are multiple accounts of a system's evolution:

$$\begin{aligned} Init_{\mathrm{AC\_SON}} &\overset{\mathrm{df}}{=} \{Init_{\mathrm{AC\_SON} \downdownarrows \mathsf{A}_1}, \ldots, Init_{\mathrm{AC\_SON} \downdownarrows \mathsf{A}_\ell}\} \\ Fin_{\mathrm{AC\_SON}} &\overset{\mathrm{df}}{=} \{Fin_{\mathrm{AC\_SON} \downdownarrows \mathsf{A}_1}, \ldots, Fin_{\mathrm{AC\_SON} \downdownarrows \mathsf{A}_\ell}\} . \end{aligned}$$

In the same way, a cut of $\mathrm{AC\_SON}$ is any of the cuts of the occurrence nets $\mathrm{AC\_SON} \downdownarrows \mathsf{A}_i$.

Proposition 1. *If Cut is a cut of the* $\mathrm{AC\_SON}$*, then there is* $\mathsf{A} \in AV$ *such that* $Cut \cap C_i$ *is a cut of* $\mathrm{ON}_i \downdownarrows \mathsf{A}$*, for every* $i \leq k$*.* $\diamond$

That is, projecting cuts of ACSON produces valid cuts of the component AONs according to one of the possible views.

A *step execution* of $\mathrm{AC\_SON}$ is a sequence $\chi \overset{\mathrm{df}}{=} D_0\, G_1\, D_1 \ldots G_n\, D_n$ ($n \geq 0$), where each D_i is a set of conditions and each G_i is a possibly empty set of events (called a *step*), for which is $\mathsf{A} \in AV$ such that χ is a step sequence of the occurrence net $\textsc{ac\_son} \downdownarrows \mathsf{A}$. We also say that the step execution χ *leads* to the cut D_n, and that D_n is *reachable*.

The following result extends the observation behind Proposition 1 to step executions.

Theorem 2. *Given the step execution as above, for every $j \leq k$, there is $\mathsf{A} \in AV$ such that the sequence*

$$D_0 \cap C_j \;\; G_1 \cap E_j \;\; D_1 \cap C_j \;\ldots\; G_n \cap E_j \;\; D_n \cap C_j$$

is a step execution of the occurrence net $\textsc{on}_j \downdownarrows \mathsf{A}$. ◇

The final result is a re-statement of Theorem 1 formulated for AONs.

Theorem 3. *Given a step execution as above, we have that: each D_i is a cut of* $\textsc{ac\_son}$*; no event occurs more than once; and $D_n = Fin_{\textsc{ac\_son}}$* iff *each event of one of the occurrence nets $\textsc{ac\_son} \downdownarrows \mathsf{A}$ occurs in the execution. Moreover, each cut of* $\textsc{ac\_son}$ *can be reached from the initial cut through some step execution, and there is a step execution involving all the events in $\textsc{ac\_son} \downdownarrows \mathsf{A}$, for every $\mathsf{A} \in AV$.* ◇