COMPUTING SCIENCE

Structured Occurrence Nets: A formalism for aiding system failure prevention and analysis techniques

Maciej Koutny and Brian Randell

TECHNICAL REPORT SERIES

No. CS-TR-1162 August 2009

TECHNICAL REPORT SERIES

No. CS-TR-1162 August, 2009

Structured Occurrence Nets: A formalism for aiding system failure prevention and analysis techniques

Maciej Koutny and Brian Randell

Abstract

This paper introduces the concept of a 'structured occurrence net', which as its name indicates is based on that of an 'occurrence net', a well-established formalism for an abstract record that represents causality and concurrency information concerning a single execution of a system. Structured occurrence nets consist of multiple occurrence nets, associated together by means of various types of relationship, and are intended for recording or predicting, either the actual behaviour of complex systems as they communicate and evolve, or evidence that is being gathered and analysed concerning their alleged past behaviour. We provide a formal basis for the new formalism and show how it can be used to gain better understanding of complex fault-error-failure chains (i) among co-existing communicating systems, (ii) between systems and their sub-systems, and (iii) involving systems that are controlling, creating or modifying other systems. We then go on to discuss how, with appropriate tools support, perhaps using extended versions of existing tools, structured occurrence nets could form a basis for improved techniques of system failure prevention and analysis.

(This is a revised and significantly extended version of TR-1120.)

Printed and published by the University of Newcastle upon Tyne,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

RANDELL, B., KOUTNY, M.

Structured Occurrence Nets: A formalism for aiding system failure prevention and analysis techniques
[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-1162)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-1162

Abstract

This paper introduces the concept of a ‘structured occurrence net’, which as its name indicates is based on that of an ‘occurrence net’, a well-established formalism for an abstract record that represents causality and concurrency information concerning a single execution of a system. Structured occurrence nets consist of multiple occurrence nets, associated together by means of various types of relationship, and are intended for recording or predicting, either the actual behaviour of complex systems as they communicate and evolve, or evidence that is being gathered and analysed concerning their alleged past behaviour. We provide a formal basis for the new formalism and show how it can be used to gain better understanding of complex fault-error-failure chains (i) among co-existing communicating systems, (ii) between systems and their sub-systems, and (iii) involving systems that are controlling, creating or modifying other systems. We then go on to discuss how, with appropriate tools support, perhaps using extended versions of existing tools, structured occurrence nets could form a basis for improved techniques of system failure prevention and analysis.

(This is a revised and significantly extended version of TR-1120.)

About the authors

Professor Brian Randell’s earliest work was during the period 1957-1964 while I was English Electric, was on compilers. This led to the book: *Algol 60 Implementation*. (Co-author L. J. Russell). Academic Press, London, 1964. He then joined IBM T.J. Watson Research Center, Yorktown Heights, N.Y. where, with an intervening year during 1965-66 in California, he worked on high performance computer architectures (the ACS Project), then on operating systems and system design methodology. During this time, and shortly after Brian returned to the UK to became Professor of Computing Science at the University of Newcastle upon Tyne, he was co-editor of the reports on the two NATO Software Engineering Conferences.

In 1971 Brian 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. Brian has been Principal Investigator on a succession of research projects on system dependability funded by the Science Research Council (now Engineering and Physical Sciences Research Council), the Ministry of Defence, the European Strategic Programme of Research in Information Technology (ESPRIT), and the European Information Society Technologies (IST) Programme. Most recently he has performed the role of Project Director for CaberNet (the IST Network of Excellence on Distributed Computing Systems Architectures) and for two IST Research Projects, MAFTIA (Malicious- and Accidental-Fault Tolerance for Internet Applications) and DSoS (Dependable Systems of Systems). Currently he is involved with the RODIN IST project, and the ReSIST IST Network of Excellence.

Brian’s current computing science research continues to be focused on Dependability (for example on failure analysis) and, to a lesser extent, on the History of Computing.

He was a founder-member of IFIP WG2.3 (Programming Methodology) and am a founder-member of IFIP WG10.4 (Dependability and Fault Tolerance). In 1979 he helped found MARI (Microelectronics Applications Institute) and in 1993 was involved in setting up the Northern Informatics Applications Agency, both of which flourished and did some excellent work for a number of years.

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 Univerity.

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, OCCURENCE NETS, ABSTRACTION, FORMAL ANALYSIS

Structured Occurrence Nets: A formalism for aiding system failure prevention and analysis techniques

Maciej Koutny and Brian Randell

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

Abstract. This paper introduces the concept of a 'structured occurrence net', which as its name indicates is based on that of an 'occurrence net', a well-established formalism for an abstract record that represents causality and concurrency information concerning a single execution of a system. Structured occurrence nets consist of multiple occurrence nets, associated together by means of various types of relationship, and are intended for recording or predicting, either the actual behaviour of complex systems as they communicate and evolve, or evidence that is being gathered and analysed concerning their alleged past behaviour. We provide a formal basis for the new formalism and show how it can be used to gain better understanding of complex fault-error-failure chains (i) among co-existing communicating systems, (ii) between systems and their sub-systems, and (iii) involving systems that are controlling, creating or modifying other systems. We then go on to discuss how, with appropriate tools support, perhaps using extended versions of existing tools, structured occurrence nets could form a basis for improved techniques of system failure prevention and analysis. (This is a revised and significantly extended version of TR-1120.)
Keywords: failures, errors, faults, dependability, judgement, occurrence nets, abstraction, formal analysis.

1 Introduction

The concept of a failure of a system is central both to system dependability and to system security, two closely associated and indeed somewhat overlapping research domains. Specifically, particular types of failures (e.g., producing wrong results, ceasing to operate, revealing secret information, causing loss of life, etc.) relate to, indeed enable the definition of, what can be regarded as different attributes of dependability/security: respectively reliability, availability, confidentiality, safety, etc. The paper by Avizienis et al. [1] provides an extended (informal) discussion of the basic concepts and terminology of dependability and security, and contains a detailed taxonomy of dependability and security terms. Our aims in this present paper are: (i) to improve our understanding — in part by formalising — of the concept of failure (and error and fault) as given by [1]; (ii) to deal with systems that are evolving, e.g., through suffering modifications; (iii) to reduce (in fact by uniting the apparently different concepts of 'system' and 'state') the number of base concepts, i.e., concepts that the paper uses without explicit definition; and (iv) to provide a basis for an investigation of possible improved techniques of system failure prevention and analysis. The paper is a greatly extended version of [21], providing proofs for the various results that were merely indicated in our earlier paper, together with several new concepts, definitions and supporting results.

Complex real systems, made *up* of other systems, and made *by* other systems (e.g., of hardware, software and people) evidently fail from time to time, and reducing the frequency and severity of their failures is a major challenge — common to both the dependability and the security communities. Indeed, a dependable/secure system can be regarded as *one whose (dependability/security) failures are not unacceptably frequent or severe* (from some given viewpoint).

We will return shortly to the issue of viewpoint. But first let us quote the definitions of three basic and subtly-distinct concepts, termed 'failure', 'fault' and 'error' in [1]:

> 'A system *failure* occurs when the delivered service deviates from fulfilling the system function, the latter being what the system is *aimed at*. An *error* is that part of the system state which is *liable to lead to subsequent failure*: an error affecting the service is an indication that a failure occurs or has occurred. The *adjudged or hypothesised cause* of an error is a *fault*.'

Note that errors do not necessarily lead to failures — such occurrences may be avoided by chance or design. Similarly, failures in a component system do not necessarily constitute faults to the surrounding system — this depends on how the surrounding system is relying on the component. These three concepts (respectively an event, a state, and a cause) are evidently distinct, and so need to be distinguished, whatever names are chosen to denote them. The above quotation makes it clear that judgement can be involved in identifying error causes, i.e., faults. However it is also the case that identifying failures and errors involves judgement (not necessarily simple adherence to some pre-existing specification) — a critical point that we will return to shortly.

A failure can be judged to have occurred when an error 'passes through' the system-user interface and affects the service delivered by the system — a system being composed of components which are themselves systems. This failure may be significant, and thus constitute a fault, to the enclosing system.

Thus the manifestation of failures, faults and errors follows a 'fundamental chain':

$$\ldots \rightarrow \text{failure} \rightarrow \text{fault} \rightarrow \text{error} \rightarrow \text{failure} \rightarrow \text{fault} \rightarrow \ldots,$$

i.e.,

$$\ldots \rightarrow \text{event} \rightarrow \text{cause} \rightarrow \text{state} \rightarrow \text{event} \rightarrow \text{cause} \rightarrow \ldots.$$

It is critical to note that this chain can flow from one system to: (i) another system that it is interacting with; (ii) a system which it is part of; and (iii) a system which it creates or sustains.

Typically, a failure will be judged to be due to multiple co-incident faults, e.g., the activity of a hacker exploiting a bug left by a programmer. Identifying failures (and hence errors and faults), even understanding the concepts, is difficult. There can be uncertainties about system boundaries, the very complexity of the systems (and of any specifications) is often a major difficulty, the determination of possible causes or consequences of failure can be a very subtle and iterative process, and any provisions for preventing faults from causing failures may themselves be fallible. Attempting to enumerate a system's possible failures beforehand is normally impracticable. Instead, one can appeal to the notion of a 'judgemental system'.

The 'environment' of a system is the wider system that it affects (by its correct functioning, and by its failures), and is affected by. What constitutes correct (failure-free) functioning *might* be implied by a system specification — assuming that this exists, and is complete, accurate and agreed. (Often the specification is part of the problem!) However, in principle a third system, a *judgemental system*, is involved in determining whether any particular activity (or inactivity) of a system in a given environment constitutes or would constitute — *from its viewpoint* — a *failure*. Note that the judgemental system and the environmental system might be one and the same, and the judgement might be instant or delayed. The judgemental system might itself fail — as judged by some further system — and different judges, or the same judge at different times, might come to different judgements.

The term 'Judgemental System' is deliberately broad — it covers from on-line failure detector circuits, via someone equipped with a system specification, to the retrospective activities of a court of enquiry (just as the term 'system' is meant to range from simple hardware devices to complex computer-based systems, composed of hardware, software and people). Thus the judging activity may be clear-cut and automatic, or essentially subjective — though even in the latter case a degree of predictability is essential, otherwise the system designers' task would be impossible. The judgement is an action by a system, and so can in

principle fail either positively or negatively. This possibility is allowed for in the legal system, hence the concept of a hierarchy of crown courts, appeal courts, supreme courts, etc., in the British legal system.

In this paper we describe a means of modelling the activity of systems — operational computing systems, the systems of people and computers that created them or are adapting them, the systems that are passing judgements on them, etc. The formalism that we use in this paper is based on that of *occurrence nets* [3, 8, 22]. We introduce this formalism not just in order to clarify such concepts as fault-error-failure chains, and the role of judgemental systems, but also because the occurrence net formalism is well-supported by tools for system validation and synthesis [6, 10–12, 19], tools which we believe could be significantly enhanced by being extended so as to take advantage of the concept that we introduce in Sections 3-6 of this paper of 'structured occurrence nets'. (Section 7 sketches the ways in which we envisage exploiting such enhanced tools.)

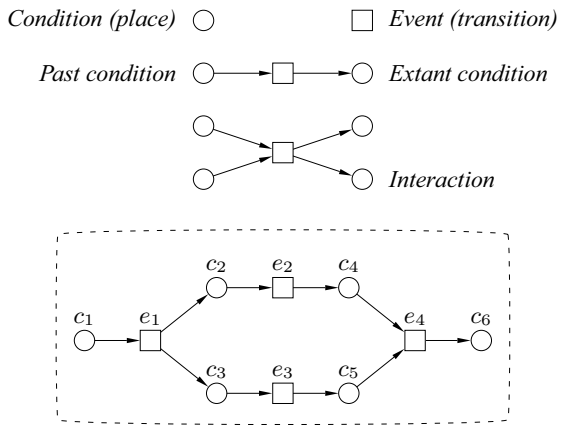


Fig. 1. Basic notation (top) and an occurrence net (bottom).

As can be seen in Figure 1, occurrence nets are directed acyclic graphs that portray the (alleged) past and present state of affairs, in terms of places (i.e., conditions, represented by circles), transitions (i.e., events, represented by squares) and arrows (each from a place to a transition, or from a transition to a place, representing (alleged) causality). For simple nets, an actual graphical representation suffices — and will be used here using the notation shown in Figure 1. (In the case of complex nets, these are better represented in some linguistic or tabular form.) We will also take advantage of our belated realisation that the concepts of 'system' and 'state' are not separate, but just a question of abstraction, so that (different related) occurrence nets can represent both systems and their states using the same symbol — a 'place'. In fact in this paper we introduce and define, and discuss the utility of, several types of relationship, and term a set of related occurrence nets a *structured occurrence net*. These types of relationship differ depending on the specific means and objectives of a particular investigation. However, there are some fundamental constraints that any structured occurrence net ought to satisfy. Crucially, we will require that the structures we admit *avoid cycles* in systems' temporal behaviour as these contradict the accepted view on the way physical systems could possibly behave.

Note that it is easy to understand how occurrence nets could be 'generated' by executing Petri nets representing computing systems, but they could in fact be used to record the execution of any (potentially asynchronous) process, hardware or software, indeed human, no matter what notation or language might be used to define it. It is also worth noting that various other graphical notations similar to occurrence nets can be found in both the hardware and the software design worlds, e.g., strand spaces [23], signal diagrams [16] and message sequence charts [17].

Proofs of all the results included in the paper proper, together with a number of auxiliary properties, are presented in the appendix. We also recall there standard mathematical notions and notations used throughout the paper.

2 Occurrence nets

In this section, we present the basic model of an occurrence net that is standard within Petri net theory [3, 8, 22]. Later on, we will extend it to express more intricate features of our approach to the modelling of complex behaviours. In a nutshell, an occurrence net is an abstract record of a single execution of some computing system (though they can be used to portray behaviours of quite general systems, e.g., ones that include people) in which only information about causality and concurrency between events and visited local states is represented. Together with a natural requirement that causal cycles do not occur in the physical world, this means that the underlying mathematical structure of an occurrence net is that of a partial order. This should be contrasted with an 'interleaving' record of a computation which presupposes a sequential ordering of all the events involved, and has as an underlying structure a total order.

Definition 1 (occurrence net ON). *An* occurrence net *is a triple* $\textsc{on} \stackrel{\text{df}}{=} (C, E, F)$ *where* $C \neq \varnothing$ *and* E *are finite*[1] *disjoint sets of respectively* conditions *and* events *(collectively, conditions and events are the* nodes *of* ON*), and* $F \subseteq (C \times E) \cup (E \times C)$ *is a* flow relation *satisfying the following: (i) for every condition* c *there is at most one event* e *such that* $(e, c) \in F$*, and at most one event* f *such that* $(c, f) \in F$*; (ii) for every event* e *there is at least one condition* c *such that* $(c, e) \in F$*, and at least one condition* d *such that* $(e, d) \in F$*; and (iii) the relation* $Prec_{\textsc{on}} \stackrel{\text{df}}{=} (F \circ F)|_{C \times C}$ *is acyclic (in other words, its transitive closure is irreflexive), and so* ON *forms an acyclic graph and* F^+ *is a partial order relation.* ◇

In the above definition — aimed at capturing the essence of a computation history — E represents the events which have actually been executed and C represents conditions (or holding of local states) enabling their executions as well as resulting from their executions. Here we will discuss computation histories as though they have actually occurred; however, the term will also be used of 'histories' that *might* have occurred, or that might occur in the future, given the existence of an appropriate system. The flow relation records the causality relationship between events and conditions. Thus the direct precedence (or causality) relationship between conditions is captured by the relation $Prec_{\textsc{on}}$, and the indirect precedence is captured by the transitive closure $Prec^+_{\textsc{on}}$. The first condition in the above definition means that each non-initial condition is uniquely caused, and each of the non-final conditions caused a unique event.[2] The second condition states that each event has at least one cause and at least one effect, and the third one simply renders formal a common belief that causality is not circular.

> Figure 1 depicts an occurrence net ON such that $C = \{c_1, \ldots, c_6\}$, $E = \{e_1, \ldots, e_4\}$ and $F = \{(c_1, e_1), (e_1, c_2), (e_1, c_3), \ldots, (c_5, e_4), (e_4, c_6)\}$. From the acyclic graph of the relation $Prec_{\textsc{on}}$ shown in Figure 2 one can, for example, deduce that the conditions c_1 and c_6 are causally related; in other words, $(c_1, c_6) \in Prec^+_{\textsc{on}}$.

Now we introduce a few useful notations based on the structure of an occurrence net ON but used later also to capture its dynamic properties: (i) for each node x we use $pre(x)$ and $post(x)$ to denote the set of all nodes y such that $(y, x) \in F$ and $(x, y) \in F$, respectively; (ii) two nodes, x and y, are *causally related* if $(x, y) \in F^+$ or $(y, x) \in F^+$ and otherwise they are *concurrent*; (iii) a *cut* is a maximal (w.r.t. set inclusion) set of mutually concurrent conditions; and (iv) $Init_{\textsc{on}}$ and $Fin_{\textsc{on}}$ are the sets of all conditions

[1] For simplicity, we only discuss finite behaviours and so all (structured) occurrence nets considered in this paper will be finite.

[2] Note that if an event is only *conditional* on the presence of a condition, but does not invalidate it, then the event can re-establish this condition by producing a fresh copy of the condition.

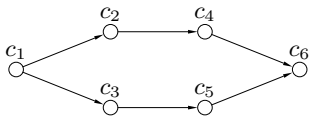


Fig. 2. Causal relationships for the occurrence net of Figure 1.

c such that $pre(c) = \varnothing$ and $post(c) = \varnothing$, respectively. Moreover, for each event e, $pre(e)$ and $post(e)$ will be called *pre-conditions* and *post-conditions*, respectively, and, for a set of events G, we respectively denote by $pre(G)$ and $post(G)$ the sets of all pre-conditions and post-conditions of events in G.

Intuitively, $pre(x)$ and $post(x)$ correspond to the arcs incoming to node x and outgoing from x, respectively. Cuts are the (global) states that the system has passed through during the execution captured by the occurrence net. In particular, $Init_{\text{ON}}$ and Fin_{ON} are cuts; the former corresponds to the initial state of the history represented by ON, and the latter to its final state.

For the occurrence net ON depicted in Figure 1, we have $pre(e_4) = \{c_4, c_5\}$ and $post(c_1) = \{e_1\}$. The nodes c_3 and e_4 are causally related whereas c_2 and c_5 are concurrent. The initial and final cuts are $Init_{\text{ON}} = \{c_1\}$ and $Fin_{\text{ON}} = \{c_6\}$, respectively, and the other four cuts of this occurrence net are $\{c_2, c_3\}$, $\{c_2, c_5\}$, $\{c_4, c_3\}$ and $\{c_4, c_5\}$.

An occurrence net is usually derived from a single execution history of the system. However, since it only records essential (causal) orderings, it can also convey information about other potential executions with the same underlying causal ordering of events. This calls for a precise notion of an execution of a given occurrence net.

Definition 2 (sequential execution of ON). *A* sequential execution *of the occurrence net* ON *as in Definition 1 is a sequence* $\gamma \stackrel{\text{df}}{=} D_0\, e_1\, D_1 \ldots e_n\, D_n$ *($n \geq 0$), where each D_i is a set of conditions and each e_i is an event, such that $D_0 = Init_{\text{ON}}$ and, for every $i \leq n$, we have $pre(e_i) \subseteq D_{i-1}$ and $D_i = (D_{i-1} \setminus pre(e_i)) \cup post(e_i)$.* ◇

Thus an execution starts in the initial global state, and each successive event transforms a current global state into another one according to the set of conditions in its vicinity. Basically, an event can be executed (or is *enabled*) if all its pre-conditions (local states) hold. After the execution, they cease to hold, and all post-conditions (local states) of the event begin to hold.

For the occurrence net depicted in Figure 1, $\gamma = \{c_1\}\, e_1\, \{c_2, c_3\}\, e_2\, \{c_4, c_3\}\, e_3\, \{c_4, c_5\}\, e_4\, \{c_6\}$ is a sequential execution leading from the initial to final cut, and is illustrated graphically in Figure 3. Another possible sequential execution is $\{c_1\}\, e_1\, \{c_2, c_3\}\, e_3\, \{c_4, c_3\}$.

The above definition implies a couple of simple, yet important facts. Basically, these imply that ON is sound in the sense of obeying some natural temporal properties as well as testifying to the fact that ON does not contain redundant parts. We also have a complete characterisation of global states reachable from the default initial one — these are all the cuts of ON. In practical terms, the latter means that we can verify state properties of the computations captured by ON by running a model checker which inspects all the cuts. Such a model checker could be based on a SAT-solver or integer programming, e.g., as in [5, 10, 12].

Theorem 1 (see [3]). *Given the sequential execution as in Definition 2, we have that: each D_i is a cut of* ON*; no event occurs more than once; and $D_n = Fin_{\text{ON}}$* iff *each event of E occurs in the execution. Moreover, each cut of* ON *can be reached from the initial cut through some sequential execution, and there is a sequential execution involving all the events in E.* ◇

An alternative, more concurrent, notion of execution considers that in a single computational move, a set of events (called a *step*) rather than a single event is executed.

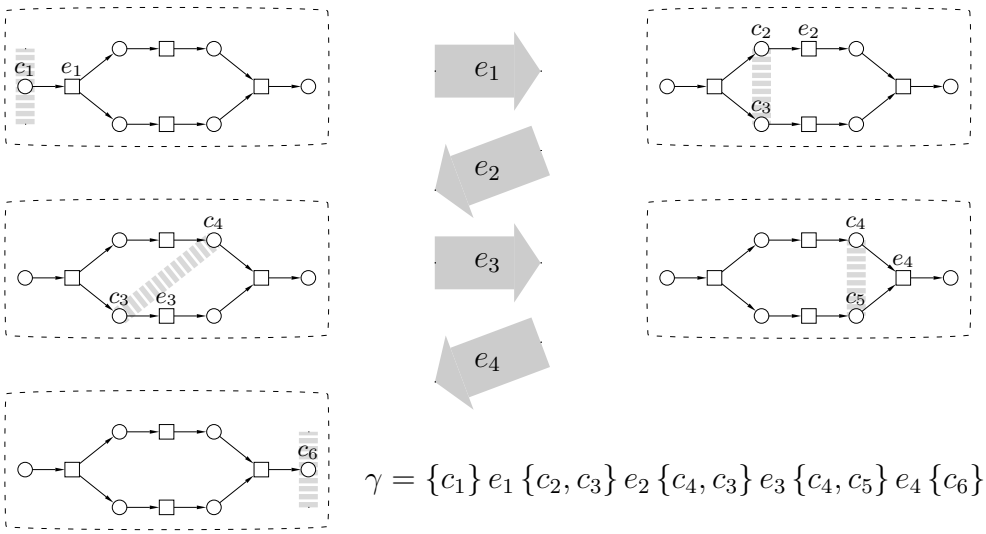


Fig. 3. A possible sequential execution γ leading from the initial to final cut of the occurrence net of Figure 1. In this, and other similar pictures, we indicate cuts (global states) by wide dashed lines, and the wide arrows indicate the order of execution together with the labels of the event(s) being executed. At each stage of the execution only the relevant conditions and events are labelled with their identities.

Definition 3 (step execution of ON). *A* step execution *of the occurrence net* ON *as in Definition 1 is a sequence* $\chi \stackrel{\text{df}}{=} D_0\, G_1\, D_1 \ldots G_n\, D_n$ *($n \geq 0$), where each D_i is a set of conditions and each G_i is a possibly empty set of events (called a* step*), such that $D_0 = Init_{\text{ON}}$ and, for every $i \leq n$, we have $pre(G_i) \subseteq D_{i-1}$ and $D_i = (D_{i-1} \setminus pre(G_i)) \cup post(G_i)$. We also say that the step execution χ* leads *to the cut D_n, and that D_n is* reachable. ◇

As for sequential executions, it is required of step executions that at each stage all pre-conditions of executed events hold; after the execution they cease to hold, and all post-conditions of executed events begin to hold.

For the occurrence net depicted in Figure 1, $\chi = \{c_1\}\,\{e_1\}\,\{c_2, c_3\}\,\{e_3\}\,\{c_4, c_3\}$ as well as $\chi' = \{c_1\}\,\{e_1\}\,\{c_2, c_3\}\,\{e_2, e_3\}\,\{c_4, c_5\}\,\{e_4\}\,\{c_6\}$ are two of its possible step executions.

For the basic model of occurrence nets, the sequential and step executions are broadly speaking equivalent. In particular, a version Theorem 1 holds also for step executions.

Theorem 2 (see [3]). *Given the step execution as in Definition 3, we have that: each D_i is a cut of* ON*; no event occurs more than once; and $D_n = Fin_{\text{ON}}$* iff *each event of E occurs in the execution. Moreover, each cut of* ON *can be reached from the initial cut through some step execution, and there is a step execution involving all the events in E.* ◇

Note that for extended notions of occurrence nets, such as those employing activator arcs discussed in [13] but not here, the sequential and step execution semantics may not be equivalent, e.g., there may exist global states reachable through step executions which are impossible to reach through sequential executions.

The next result can be understood as a statement of a consistency between the causality captured by the flow relation and the temporal ordering of events involved in a step execution.

Proposition 1 (see [3]). *Given the step execution as in Definition 3 and $i \geq j$, $(G_i \times G_j) \cap F^+ = \varnothing$.* $\diamond$

The above result states that causal predecessors of an event can never be executed after (or together with) that event.

We end this section introducing two kinds of structures present in occurrence nets which will prove useful in the rest of this paper.

Definition 4 (phases and blocks of ON). *Let* ON *be the occurrence net as in Definition 1.*

- *A* phase *of* ON *is a non-empty set π of conditions such that: the set Min_π of minimal (w.r.t. F^+) conditions of π is a cut; the set Max_π of maximal (w.r.t. F^+) conditions of π is a cut; and π comprises all conditions c of* ON *for which there are $b \in Min_\pi$ and $d \in Max_\pi$ satisfying $(b, c) \in F^*$ and $(c, d) \in F^*$.*
- *A* phase decomposition *of* ON *is a sequence $\pi_1 \ldots \pi_m$ ($m \geq 1$) of phases of the occurrence net such that $Init_{\text{ON}} = Min_{\pi_1}$, $Max_{\pi_i} = Min_{\pi_{i+1}}$ (for $i \leq m-1$) and $Max_{\pi_m} = Fin_{\text{ON}}$.*
- *A* block *of* ON *is a non-empty set B of nodes such that $B \cap C = pre(B \cap E) \cap post(B \cap E)$ and $(pre(B) \setminus B) \times (post(B) \setminus B) \subseteq Prec^+_{\text{ON}}$.* $\diamond$

Phases will be used to represent a succession of system modifications (to be discussed in Section 4) in the evolution of systems. Each phase is a fragment of an evolution beginning with a global state and ending with a global state which follows it in the causal sense, including all the conditions occurring in-between these global states. A phase decomposition is then a sequence of phases arranged back-to-back so that whenever one phase ends, the successive one begins. Since the decomposition starts with the initial state and ends with the final one, each condition in ON belongs to at least one phase in such a decomposition. Blocks represent contiguous fragments of activity within the behaviour represented by an occurrence net. Intuitively, a block is a set of nodes in which conditions are internal to the block (i.e., each condition has a predecessor and successor events within the block), and all external preconditions of events in the block causally precede all external postconditions of events in the block. The latter requirement is motivated by our intention to use blocks as representations of atomic actions (intuitively, all inputs to such an action must be present before outputs are produced).

For the occurrence net depicted in Figure 1, one of the possible phase decompositions is $\pi_1\pi_2\pi_3 = \{c_1, c_2, c_3\}\,\{c_2, c_3, c_4, c_5\}\,\{c_4, c_5, c_6\}$, and it is illustrated in Figure 4(a). In the diagram, as in Figure 3, the wide dashed lines indicate the global states (cuts) where the three component phases begin and end. Note that we allow phases where the initial and final cuts are the same, and so $\pi'_1\pi'_2\pi'_3 = \{c_1, c_2, c_3, c_5\}\,\{c_4, c_5\}\,\{c_4, c_5, c_6\}$ is also a possible phase decomposition of the occurrence net of Figure 1 even though $Min_{\{c_4,c_5\}} = Max_{\{c_4,c_5\}} = \{c_4, c_5\}$. Among the blocks of the same occurrence net one can find, for example, $B = \{e_2, e_3, e_4, c_4, c_5\}$ and $B' = \{e_1, e_2, c_2\}$. Figure 4(b) uses shadowing to show the nodes of B.

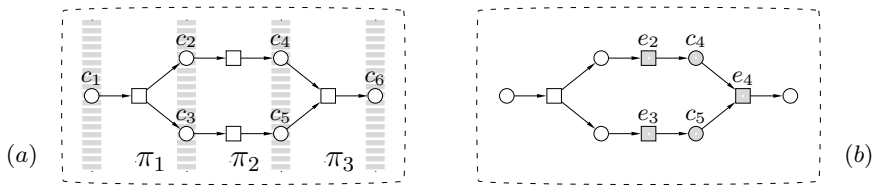


Fig. 4. A phase decomposition (a), and a block (b) of the occurrence net shown in Figure 1.

In this section we introduced basic notions concerning occurrence nets and recalled some fundamental results about their behaviour which we will investigate in the extended model described in subsequent

sections of this paper. These detail a number of different ways in which multiple occurrence nets (ONs) can be related together in order to construct *structured occurrence nets* (SONs).

3 Communication

We now outline the first of several ways of structuring occurrence nets, something that can be done either by defining one or more relations between a set of hitherto separate occurrence nets, or by adding structure to an existing occurrence net, i.e., by replacing it by an equivalent set of related smaller occurrence nets. In subsequent figures we follow a convention that conditions and events of different systems are identified by shading them differently; there are some obvious rules about legal such labellings (e.g., that they partition the nodes into disjoint sets). A further graphical convention is 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.

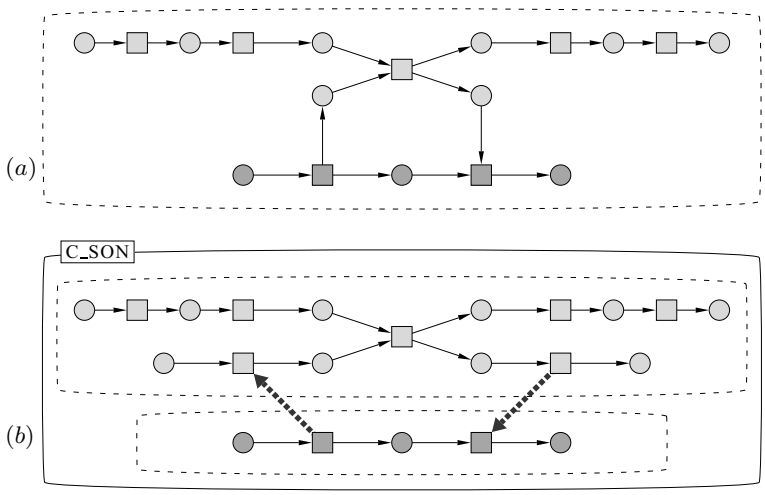


Fig. 5. Part (a) portrays the activity of two systems, one of which is exhibiting asynchronous behaviour, in a single occurrence net; whereas (b) portrays an essentially equivalent structured occurrence net, in which the activities of these two systems are shown in separate (communicating) occurrence nets.

Our first method of structuring captures *communication*, i.e., a situation in which separate occurrence nets proceed concurrently and (occasionally) communicate with each other — see, for example, Figure 5(b), in which thick dashed arcs are used to represent communications so as to distinguish them from the interactions represented in conventional occurrence nets by causal arcs. Note that another distinction is that interactions link conditions to events and events to conditions (as was shown in Figure 1), whereas communications link events — of separate occurrence nets — directly.

In practice, when structuring a complex occurrence net into a set of simpler communicating occurrence nets, it is sometimes necessary to use synchronous communications. Hence, as shown in Figure 6, we allow for the use of two types of communication: thick dashed directed arcs indicate, for example, that an event in one occurrence net is a causal predecessor of an event in another occurrence net (i.e., information flow between the two systems was unidirectional) or that the two events can be executed synchronously (i.e., the activities of the two systems and the interactions between them could be executed synchronously), whereas

undirected such arcs indicate that the two events have been executed synchronously (i.e., information flow was bidirectional). In practice, interactions and communications of all the kinds described above can occur in the same overall structured occurrence net provided that a simple acyclicity constraint — similar to that used for ordinary occurrence nets — is satisfied.

As an example of such structuring, Figure 5(a) shows a single occurrence net that is recording the interactions between two systems, the upper of which is itself exhibiting asynchronous behaviour, and Figure 5(b) shows a possible structuring of Figure 5(a) into a structured occurrence net composed of two separate (communicating) occurrence nets, each portraying the activity of a single system. Because of our rule that communications relate events directly, the upper occurrence net in Figure 5(b) has had to be augmented with additional events and implicit conditions. Note that in Figure 5(b) the thick dashed arcs are abstractions of the details that correspond to such communications when one describes such a structured occurrence net by a single conventional occurrence net. The rules governing such abstractions, and the rationale for our introducing synchronous as well as causal communications, should become clearer later on when we discuss temporal abstraction (illustrated in Figures 20 and 21).

Definition 5 (communication SON). *Let* $\mathrm{ON}_i \stackrel{\mathrm{df}}{=} (C_i, E_i, F_i)$ *for* $i = 1, \ldots, k$ *be occurrence nets (*$k \geq 1$*) with* disjoint *sets of nodes. We denote respectively by* $\mathbf{C}$*,* $\mathbf{E}$ *and* $\mathbf{F}$ *their conditions, events and arcs.*[3]
Let κ *and* σ *be two relations (*σ *being symmetric) comprising pairs of events coming from different* ON_i*'s. For every event* $e \in \mathbf{E}$*, the sets* $Pre(e)$ *and* $Post(e)$ *respectively comprise all conditions* $c \in \mathbf{C}$ *satisfying* $(c, e) \in \mathbf{F} \circ (\kappa \cup \sigma)^*$ *and* $(e, c) \in (\kappa \cup \sigma)^* \circ \mathbf{F}$.
A communication structured occurrence net *is a tuple* $\mathrm{C\_SON} \stackrel{\mathrm{df}}{=} (\mathrm{ON}_1, \ldots, \mathrm{ON}_k, \kappa, \sigma)$ *such that the relation* $Prec_{\mathrm{C\_SON}} \stackrel{\mathrm{df}}{=} \bigcup_{e \in \mathbf{E}} Pre(e) \times Post(e)$ *is acyclic.* ◇

The above definition takes a number of disconnected occurrence nets and joins them by specifying direct causal relationships between events (i.e., κ specifies asynchronous communication, and σ specifies synchronous communication). Intuitively, if $(e, f) \in \kappa$ then e cannot happen after f, and if $(e, f) \in \sigma$ then e and f must happen synchronously. To ensure that the resulting causal dependencies remain consistent, we require the acyclicity of the relation $Prec_{\mathrm{C\_SON}}$ which captures causal relationships in C_SON, including those implied by the internal structure of the component occurrence nets. Capturing such dependencies is achieved since $(c, e) \in Pre(e)$ means that condition c must have held before the execution of event e and (perhaps indirectly) caused e. This extends the definition of $pre(e)$ (i.e., the set of those conditions which directly caused event e) and, in fact, it is always the case that $pre(e) \subseteq Pre(e)$. $Post(e)$ extends the definition of $post(e)$ in a similar way. Therefore, $(c, c') \in Pre(e) \times Post(e)$ intuitively means that there is a causal chain passing through event e originating at condition c and ending at condition c' (this idea is further developed in the Appendix).

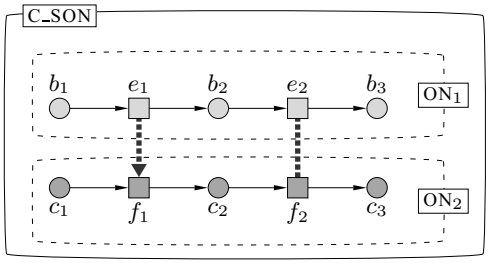


Fig. 6. A communication structured occurrence net composed out of two occurrence nets.

[3] These notations will also be used in indexed or primed form, e.g., $\mathbf{F}_j$ and $\mathbf{C}'$.

Figure 6 shows a communication structured occurrence net C_SON composed of two occurrence nets,

$$\text{ON}_1 = (\{b_1, b_2, b_3\}, \{e_1, e_2\}, \{(b_1, e_1), (e_1, b_2), (b_2, e_2), (e_2, b_3)\})$$
$$\text{ON}_2 = (\{c_1, c_2, c_3\}, \{f_1, f_2\}, \{(c_1, f_1), (f_1, c_2), (c_2, f_2), (f_2, c_3)\}) ,$$

as well as two relations between the events of the two component occurrence nets, asynchronous communication $\kappa = \{(e_1, f_1)\}$ and synchronous communication relation $\sigma = \{(e_2, f_2), (f_2, e_2)\}$. Thus, in any execution consistent with the causal relationships captured by C_SON, e_1 will never be executed after f_1 (though the two events can be executed simultaneously), whereas e_2 and f_2 will always be executed simultaneously. We further have that:

$$\begin{array}{lll} Pre(e_1) = \{b_1\} & Pre(e_2) = \{b_2, c_2\} & Pre(f_1) = \{b_1, c_1\} \\ Pre(f_2) = \{b_2, c_2\} & Post(e_1) = \{b_2, c_2\} & Post(e_2) = \{b_3, c_3\} \\ Post(f_1) = \{c_2\} & Post(f_2) = \{c_3, b_3\} , & \end{array}$$

and so the causality relationship between the conditions of C_SON is given by the acyclic relation $Prec_{\text{C\_SON}} = \{(b_1, b_2), (b_1, c_2), (b_2, b_3), (b_2, c_3), (c_1, c_2), (c_2, b_3), (c_2, c_3)\}$, shown in Figure 7.

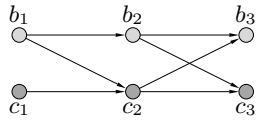


Fig. 7. Causal relationships for the communication structured occurrence net of Figure 6.

For the communication structured occurrence net as in Definition 5, cuts and step executions need to be re-defined. A *cut* of C_SON is a maximal (w.r.t. set inclusion) set of conditions $Cut \subseteq \mathbf{C}$ such that no two conditions in Cut are related by $Prec^+_{\text{C\_SON}}$. The initial cut of C_SON, $Init_{\text{C\_SON}}$, is the union of the initial cuts of all the ON_i's, and the final cut, $Fin_{\text{C\_SON}}$, is the union of the final cuts of all the ON_i's.

There are four cuts of the communication structured occurrence net of Figure 6, namely $Cut_1 = \{b_1, c_1\}$ (initial), $Cut_2 = \{b_2, c_1\}$, $Cut_3 = \{b_2, c_2\}$ and $Cut_4 = \{b_3, c_3\}$ (final).

Our first result, Proposition 2, amounts to saying that a global state of a communication structured occurrence net is made up of local states of the component occurrence nets. This, and other similar results, are formulated using the idea of *projecting* cuts, steps, and step executions of a structured occurrence net onto the component occurrence nets. In technical terms, the projection is achieved by deleting from cuts and steps all those conditions or events which do not belong to the occurrence net onto which projection is being performed.

Proposition 2. *If Cut is a cut of the* C_SON *as in Definition 5, then $Cut \cap C_i$ is a cut of* ON_i*, for every $i \leq k$.* ◇

Thus what the above result is saying is that projecting cuts of a communication structured occurrence net produces valid cuts of the component occurrence nets.

We now re-define the notion of a step execution.

Definition 6 (step execution of communication SON). *A* step execution *of the* C_SON *as in Definition 5 is a sequence $\chi \stackrel{\text{df}}{=} D_0\, G_1\, D_1 \ldots G_n\, D_n$ ($n \geq 0$), where each $D_i \subseteq \mathbf{C}$ is a set of conditions and each $G_i \subseteq \mathbf{E}$ is a set of events, such that $D_0 = Init_{\text{C\_SON}}$ and, for every $i \leq n$:*

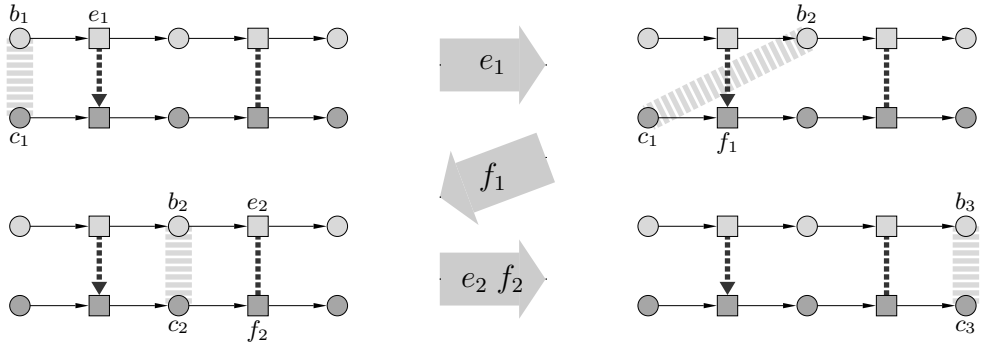
- *$pre(G_i) \subseteq D_{i-1}$ and $D_i = (D_{i-1} \setminus pre(G_i)) \cup post(G_i)$;*

– $(e, f) \in \kappa \cup \sigma$ *and* $f \in G_i$ *implies* $e \in \bigcup_{j \leq i} G_j$.

We also say that the step execution χ leads *to the cut* D_n*, and that* D_n *is* reachable. ◇

There are clear similarities between this definition and Definition 3 describing valid step executions of an occurrence net. The only new requirement is that all events which causally precede or are synchronous with a given event must have already been executed or are executed in the same step as that event. In particular, this means that if $(e, f) \in \sigma$ and $f \in G_i$ then also $e \in G_i$ as σ is symmetric and so $(f, e) \in \sigma$.

For the communication structured occurrence net depicted in Figure 6, a possible step execution is $\chi = \{b_1, c_1\}\,\{e_1\}\,\{b_2, c_1\}\,\{f_1\}\,\{b_2, c_2\}\,\{e_2, f_2\}\,\{b_3, c_3\}$. Figure 8 shows its graphical representation. Another step execution, one in which events e_1 and f_1 occur in the same step, is $\chi' = \{b_1, c_1\}\,\{e_1, f_1\}\,\{b_2, c_2\}$.



$$\chi = \{b_1, c_1\}\,\{e_1\}\,\{b_2, c_1\}\,\{f_1\}\,\{b_2, c_2\}\,\{e_2, f_2\}\,\{b_3, c_3\}$$

Fig. 8. A possible step execution χ leading from the initial to final cut of the communication structured occurrence net of Figure 6.

Our aim now is to re-establish the basic behavioural properties of occurrence nets within the domain of communication structured occurrence nets. First, however, we capture a consistency between the individual and interactive views of step executions.

Theorem 3. *Given the step execution as in Definition 6, for every* $j \leq k$*, 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 ON_j. ◇

In other words, projecting a step execution of a communication structured occurrence net onto a component occurrence net always results in a valid step execution of the occurrence net. Such a result testifies that step executions of a communication structured occurrence net are consistent with step executions of the component occurrence nets.

For the communication structured occurrence net depicted in Figure 6, projecting the step execution χ illustrated in Figure 8 on the two component occurrence nets gives the following individual views of the execution:

$$\mathrm{ON}_1 : \{b_1\}\,\{e_1\}\,\{b_2\}\,\varnothing\,\{b_2\}\,\{e_2\}\,\{b_3\}$$
$$\mathrm{ON}_2 : \{c_1\}\,\varnothing\,\{c_1\}\,\{f_1\}\,\{c_2\}\,\{f_2\}\,\{c_3\}\;.$$

Clearly, both projections are valid step executions of the respective occurrence nets.

The last result in this section is a re-statement of Theorem 2 formulated for occurrence nets, and its intuitive meaning is similar to that of the earlier result.

Theorem 4. *Given a step execution as in Definition 6, we have that: each D_i is a cut of* C_SON*; no event occurs more than once; and $D_n = Fin_{\text{C\_SON}}$* iff *each event of* $\mathbf{E}$ *occurs in the execution. Moreover, each cut of* C_SON *can be reached from the initial cut through some step execution, and there is a step execution involving all the events in* $\mathbf{E}$*.* ◇

Remark 1. To conclude the discussion of communication structured occurrence nets we make three observations:

- In general, any attempt to translate communication structured occurrence nets into behaviourally equivalent occurrence nets will fail because, e.g., the possibility to execute two events in a single step in an occurrence net always implies the possibility to execute the events in either order, whereas the same does not hold for a communication structured occurrence net. (In other words, communication structured occurrence nets are more expressive than occurrence nets.) Consider, for example, the communication structured occurrence net of Figure 6 which generates the step execution $\{b_1, c_1\}\{e_1, f_1\}\{b_2, c_2\}$. In any occurrence net which is capable of generating a step execution of the form $D\{e_1, f_1\}D'$, it is also possible to generate a step sequence of the form $D\{f_1\}D''\{e_1\}D'$. The latter, however, is impossible to realise by the communication structured occurrence net in Figure 6 as $(e_1, f_1) \in \kappa$, and so e_1 cannot be executed *after* f_1.
- It may happen that a cut of a component occurrence net ON_i cannot be obtained as a projection of any cut C_SON. Take, for example, the occurrence net ON in Figure 1 and use it to build a communication structured occurrence net C_SON depicted in Figure 9. It can be checked that two of the cuts of ON, $\{c_2, c_5\}$ and $\{c_3, c_4\}$, cannot be obtained by projecting any of the reachable cuts of C_SON. The reason is that both pairs (c_2, c_5) and (c_3, c_4) belong to $Prec_{\text{C\_SON}}$ and so no cut of C_SON can contain both c_2 and c_5 (nor both c_3 and c_4).

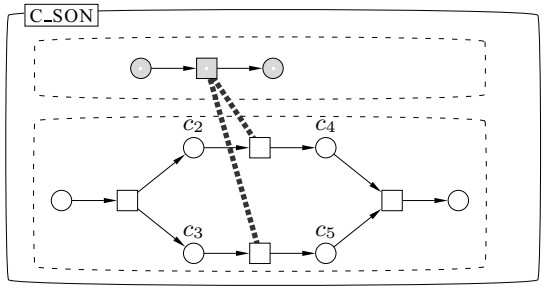


Fig. 9. A communication structured occurrence net where some of the cuts of a component occurrence net are not reachable.

- The synchronous communication relation σ is included in the definition of a communication structured occurrence net for convenience as it could be omitted after replacing κ by $\sigma \cup \kappa$. We will adopt this convention from now on and simply omit σ in formal definitions and results (but retain it in some of the examples). ◇

In this section we introduced a model of structured occurrence nets which captures communications between concurrently executed subsystems. We then demonstrated the soundness of this formalisation by

showing that the model satisfies key behavioural properties captured by Theorems 3 and 4. We now go on to describe various other forms of relation that can be used to construct structured occurrence nets.

4 Behavioural abstraction

Structures like that shown in Figures 5(b) and 6 capture communications between different systems but give no information about the evolution of individual systems. This orthogonal view is illustrated in Figure 10, where we have a two-level view of a system's history. (That it concerns a single system is indicated by the fact all conditions and events are similarly-shaded.)

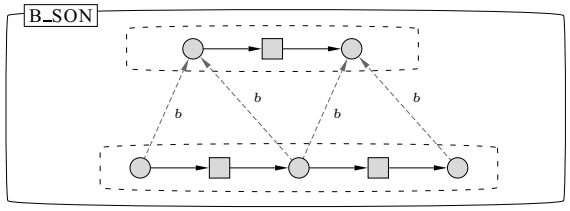


Fig. 10. Behavioural abstraction. The 'behaviour' relation is indicated by b-labelled edges.

The intended interpretation of Figure 10 is that the upper level provides a high-level view of a system which went through two successive versions which are represented by two conditions of the upper occurrence net (the event in the middle representing a version update). The lower occurrence net captures the behaviour of the system during the same period. Figure 10 also shows the 'behaviour' relation working across the two levels of description. The relation connects conditions in the lower part with those in the upper part which abstract them. We omit a formal definition of this two-level occurrence net as it is a special case of the construct introduced later in Definition 7.

In this section we aim at formalising the relationship which connects together different descriptions of the same system.

As already illustrated in Figure 10, any condition can be viewed either as a state (of some system), or as itself representing a system that presumably has its own states and events — just which is simply a matter of viewpoint. Moreover, as indicated in Figures 5(b) and 6, behaviours of different systems can interact with each other. In general, it is possible to have sets of related occurrence nets, some showing what has happened in terms of systems and their evolution, the other showing the behaviours of these systems. Thus the former can be viewed as the *behavioural abstraction* of the latter. What comes now is a combination of the structuring mechanisms that were illustrated in Figures 5(b), 6 and 10.

Figure 11 shows a simple example, involving the interacting activities of two systems (note that the same shadings are used for the higher- and lower-level view of each system). This picture gives no information about the evolution of the two systems — some such additional information is portrayed in the following figures. Moreover, the upper part of the picture does not provide any information about the interactions between the two systems (basically, all it says is that 'there are two systems').

More interesting is Figure 12(a) which shows the history of an online modification of two systems, i.e., one in which the modified systems carry on from the states that had been reached by the original systems — a possibility that is easy to imagine, though often difficult to achieve dependably, especially with software systems. In this case, the 'behaviour' relation is non-trivial as it identifies those parts of the behaviours which are pre- and post-modification ones. (Strictly speaking, Figure 12(a) is not an exact reflection of the formal definition as different occurrence nets are assumed to be disjoint, and so each of

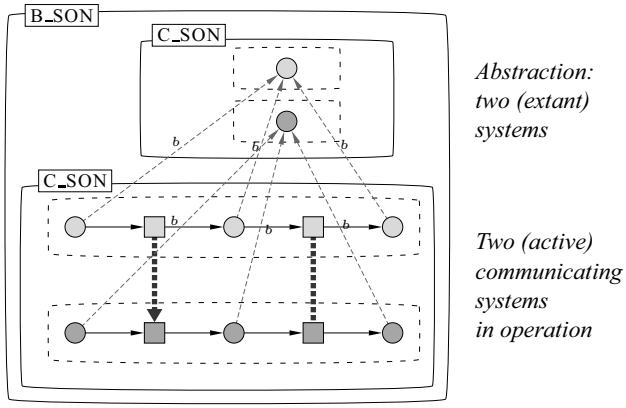


Fig. 11. Behavioural abstraction.

the pair of overlapping occurrence nets, showing the pre- and post-modification behaviours, is treated as a single occurrence net. However, one can portray a more abstract view of what is going on by showing two occurrence nets. Such a relationship can be deduced by looking at the nets of an upper level, and will be used below to identify the stages through which a system has passed during its execution.)

Another type of system modification ('offline modification') is shown in Figure 12(b). It again shows that the two systems have each suffered some sort of modification, i.e., have evolved, once — the 'abstracts' relations between the two levels show which state sequences are associated with the systems before they were modified, and which with the modified systems. Note that in this case the behaviour of each system is represented by two disjoint occurrence nets since the situation portrayed is that of modified systems restarting their activities from some given initial state, rather than continuing on from the state reached before the system modification took place. Thus standard occurrence net theory does not work as desired as it would consider these two parts as concurrent whereas, in fact, one is meant to precede the other. In the proposed structured view the upper part provides the necessary information for the desired sequencing of the occurrence nets in the lower part. Again, this is a feature which is due to the multi-level view of behaviours.

The last motivating example in this section, Figure 13, shows some of the earlier history of the two systems in Figure 11, i.e., that one system has spawned the other system, and after that both systems went through some independent further evolutions. Note that additional information could have been portrayed in the figures by showing relations, from the earlier versions of the two systems, to parts of the occurrence nets which recorded the behaviour that occurred when these earlier versions were active — but to avoid undue graphical complexity no attempt is made to show that here. (Indeed, it may happen that no records of the prior behaviour of the two systems are available.)

We will now formalise the concept of 'behavioural abstraction' outlined above. Below, we call an occurrence net ON *line-like* if $|Init_{\rm ON}| = 1$ and $|pre(e)| = |post(e)| = 1$, for every event e. Such an occurrence net can be represented as a single chain of alternating conditions and events, $\xi \stackrel{\rm df}{=} c_1 e_1 c_2 \ldots e_{m-1} c_m$ ($m \geq 1$), such that $pre(e_i) = \{c_i\}$ and $post(e_i) = \{c_{i+1}\}$, for every $i < m$.

The upper occurrence net of Figure 6 is line-like and can be represented as the chain of nodes $b_1 e_1 b_2 e_2 b_3$, whereas the occurrence net of Figure 1 is not line-like.

Definition 7 (behavioural SON). *Let* C_SON *be as in Definition 5, and let* $\text{C\_SON}' = (\text{ON}'_1, \ldots, \text{ON}'_l, \kappa')$ *be a communication structured occurrence net with each* ON'_i *being line-like. Moreover, let* $\beta \subseteq \mathbf{C} \times \mathbf{C}'$ *be a relation between the conditions of these communication structured occurrence nets such that, for every*

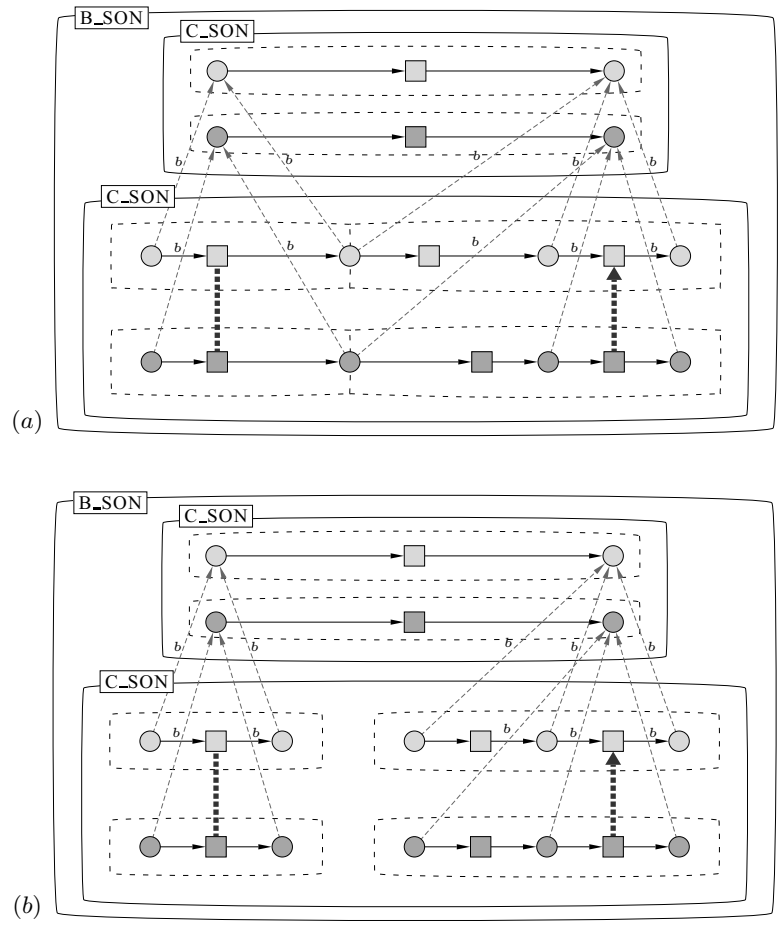


Fig. 12. System modifications: (a) online, with the modified systems carrying on from where they left off; and (b) offline, with the modified systems restarting from predefined initial states.

ON_i*, there is exactly one* ON'_j *satisfying* $\beta(C_i) \cap C'_j \neq \varnothing$.

A behavioural structured occurrence net *is a tuple* $\text{B\_SON} \stackrel{\text{df}}{=} (\text{C\_SON}, \text{C\_SON}', \beta)$ *such that:*

1. *For every line-like occurrence net in* $\text{C\_SON}'$ *represented as a chain of nodes* $\xi = c_1 e_1 c_2 \ldots e_{m-1} c_m$*, the sequence* $\pi_1 \pi_2 \ldots \pi_m \stackrel{\text{df}}{=} \beta^{-1}(c_1)\beta^{-1}(c_2)\ldots\beta^{-1}(c_m)$ *is a concatenation of phase decompositions of different occurrence nets in* C_SON*. We also denote, for all* c_j *and* e_i *occurring in the chain* ξ*,* $\Pi(c_j) \stackrel{\text{df}}{=} \pi_j$ *and*

$$before(e_i) \stackrel{\text{df}}{=} \begin{cases} Pre(pre(Max_{\pi_i})) \times Post(post(Min_{\pi_{i+1}})) \\ \quad \cup\ Pre(e_i) \times Post(post(Min_{\pi_{i+1}})) \\ \quad \cup\ Pre(pre(Max_{\pi_i})) \times Post(e_i) & \text{if } Max_{\pi_i} = Min_{\pi_{i+1}} \\ Max_{\pi_i} \times Min_{\pi_{i+1}} \\ \quad \cup\ Pre(e_i) \times Min_{\pi_{i+1}} \ \cup\ Max_{\pi_i} \times Post(e_i) & \text{if } Max_{\pi_i} \neq Min_{\pi_{i+1}}\ . \end{cases}$$

2. *The relation* $Prec_{\text{B\_SON}} \stackrel{\text{df}}{=} Prec_{\text{C\_SON}} \cup Prec_{\text{C\_SON}'} \cup \bigcup_{e \in \mathbf{E}'} before(e)$ *is acyclic.* ◇

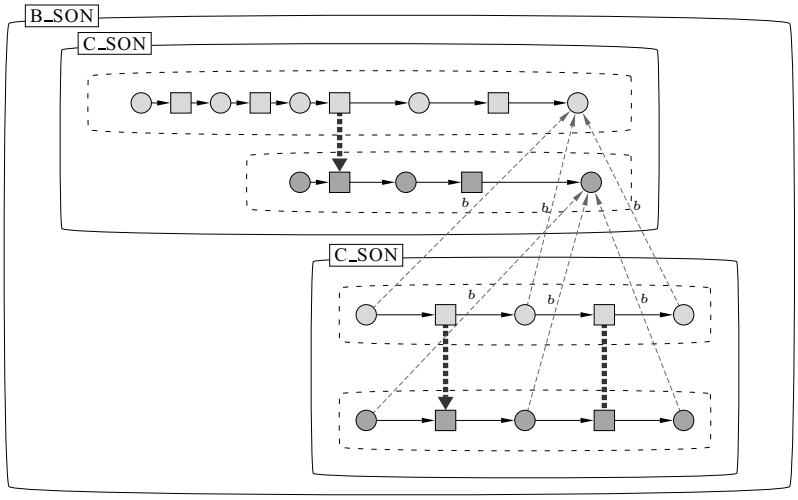


Fig. 13. System creation where the upper system has spawned the lower system.

The intuitive meaning of the above definition is as follows. (1) links together the two levels of system description represented by C_SON and C_SON′ by requiring that, in the abstracted view C_SON′, each condition represents a phase of one of the parts of the view C_SON being abstracted, i.e., a state of the system where activity is being modelled. What is more, (1) requires a consistency between the ordering of these conditions and the ordering of the corresponding phases, and so the succession of conditions in the abstracted view corresponds to a valid phase decomposition in C_SON. This necessarily implies some new causal dependencies between conditions coming from both levels of abstraction which are captured by the relations $before(e)$, where each e inducing such a relation is an event representing a progression from one phase of evolution to the next. In (2) it is required that the new dependencies, when added to those already present in C_SON and C_SON′, do not produce (causal) cycles.

Remark 2. Definition 7(1) implies that each condition of the abstraction corresponds to a phase of the abstracted system, which is not the case in Figure 13. This is not, however, a problem as for any such condition c we can introduce a very simple occurrence net with a single condition $\widehat{c}$ (and no events) and add $(\widehat{c}, c)$ to β without invalidating any of the relevant properties nor results. ◇

For the behavioural structured occurrence net B_SON depicted in Figure 14 we have the following causal relationships illustrated diagramatically in Figure 15:

$$
\begin{aligned}
Prec_{\text{C\_SON}} &= \{c_1, d_1\}\times\{c_2, d_2\} \cup \{c_3, d_3\}\times\{c_4, d_4\} \cup \{(c_4, c_5), (d_4, c_5), (d_4, d_5)\}\\
Prec_{\text{C\_SON}'} &= \{(b_1, b_2), (b_2, b_3), (b_4, b_5), (b_4, b_2)\}\\
before(e_1) &= \{(c_2, c_3), (b_1, c_3), (b_4, c_3), (c_2, b_2)\}\\
before(e_2) &= \{(c_3, c_5), (d_3, c_5), (b_2, c_5), (c_3, b_3), (d_3, b_3)\}\\
before(e_3) &= \{(d_2, d_3), (d_2, b_5), (d_2, b_2), (b_4, d_3)\}
\end{aligned}
$$

Let us check how Definition 7(1) applies to the line-like occurrence net ON'_1. The occurrence net can be represented as $\xi = b_1e_1b_2e_2b_3$, and we have $\beta^{-1}(b_1)\beta^{-1}(b_2)\beta^{-1}(b_3) = \pi_1\pi_2\pi_3 = \{c_1, c_2\}\{c_3, c_4\}\{c_4, c_5\}$. One can then observe that π_1 is a phase decomposition of ON_1, and $\pi_2\pi_3$ is a phase decomposition of ON_3. Moreover, $\Pi(b_2) = \{c_3, c_4\}$ and $\Pi(b_5) = \{d_3, d_4, d_5\}$ as well

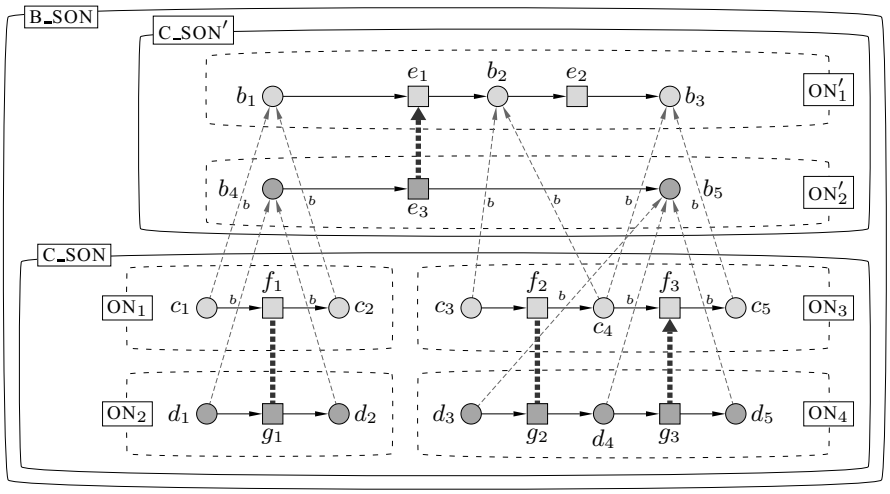


Fig. 14. A behavioural structured occurrence net.

as

$$\begin{aligned}
\mathit{before}(e_1) &= \mathit{Max}_{\Pi(b_1)} \times \mathit{Min}_{\Pi(b_2)} \cup \mathit{Pre}(e_1) \times \mathit{Min}_{\Pi(b_2)} \cup \mathit{Max}_{\Pi(b_1)} \times \mathit{Post}(e_1) \\
&= \mathit{Max}_{\{c_1,c_2\}} \times \mathit{Min}_{\{c_3,c_4\}} \cup \{b_1,b_4\} \times \mathit{Min}_{\{c_3,c_4\}} \cup \mathit{Max}_{\{c_1,c_2\}} \times \{b_2\} \\
&= \{c_2\} \times \{c_3\} \cup \{b_1,b_4\} \times \{c_3\} \cup \{c_2\} \times \{b_2\} \\
\mathit{before}(e_2) &= \mathit{Pre}(\mathit{pre}(\mathit{Max}_{\{c_3,c_4\}})) \times \mathit{Post}(\mathit{post}(\mathit{Max}_{\{c_4,c_5\}})) \\
&\quad \cup \mathit{Pre}(e_2) \times \mathit{Post}(\mathit{post}(\mathit{Max}_{\{c_4,c_5\}})) \\
&\quad \cup \mathit{Pre}(\mathit{pre}(\mathit{Max}_{\{c_3,c_4\}})) \times \mathit{Post}(e_2) \\
&= \mathit{Pre}(\mathit{pre}(\{c_4\})) \times \mathit{Post}(\mathit{post}(\{c_4\})) \\
&\quad \cup \{b_2\} \times \mathit{Post}(\mathit{post}(\{c_4\})) \cup \mathit{Pre}(\mathit{pre}(\{c_4\})) \times \{b_3\} \\
&= \mathit{Pre}(\{f_2\}) \times \mathit{Post}(\{f_3\}) \cup \{b_2\} \times \mathit{Post}(\{f_3\}) \cup \mathit{Pre}(\{f_2\}) \times \{b_3\} \\
&= \{c_3,d_3\} \times \{c_5\} \cup \{b_2\} \times \{c_5\} \cup \{c_3,d_3\} \times \{b_3\} \ .
\end{aligned}$$

We now introduce cuts and step executions for the behavioural structured occurrence net as in Definition 7. Following what can now be seen as an established pattern, a *cut* of B_SON is a maximal (w.r.t. set inclusion) set of conditions $\mathit{Cut} \subseteq \mathbf{C} \cup \mathbf{C}'$ such that no two conditions in Cut are related by $\mathit{Prec}^+_{\text{B\_SON}}$. The initial cut $\mathit{Init}_{\text{B\_SON}}$ of B_SON is the union of the initial cut of C_SON′ and the initial cuts of all the ON_i's such that $\beta(\mathit{Init}_{\text{ON}_i}) \subseteq \mathit{Init}_{\text{C\_SON}'}$. Similarly, the final cut $\mathit{Fin}_{\text{B\_SON}}$ of B_SON is the union of the final cut of C_SON′ and the final cuts of all the ON_i's such that $\beta(\mathit{Fin}_{\text{ON}_i}) \subseteq \mathit{Fin}_{\text{C\_SON}'}$.

For the behavioural structured occurrence net of Figure 14, there are altogether nine cuts:

$$\begin{aligned}
(\mathit{Init}_{\text{B_SON}} =)\ &\{b_1,b_4,c_1,d_1\}\ \{b_1,b_4,c_2,d_2\}\ \{b_1,b_5,c_2,d_3\} \\
&\{b_2,b_5,c_3,d_3\}\ \{b_2,b_5,c_4,d_4\}\ \{b_2,b_5,c_4,d_5\} \\
&\{b_3,b_5,c_4,d_4\}\ \{b_3,b_5,c_4,d_5\}\ \{b_3,b_5,c_5,d_5\}\ (= \mathit{Fin}_{\text{B_SON}})\ .
\end{aligned}$$

It is not difficult to observe that by projecting any of these cuts onto, say, C_SON′ yields $\{b_1,b_4\}$ or $\{b_1,b_5\}$ or $\{b_2,b_5\}$ or $\{b_3,b_5\}$, each of which is a valid cut of the communication structured occurrence net C_SON′.

What we just observed is in fact an instance of a general result akin to Proposition 2 which establishes a consistency between the cuts of a behavioural structured occurrence net and the cuts of the component (structured) occurrence nets.

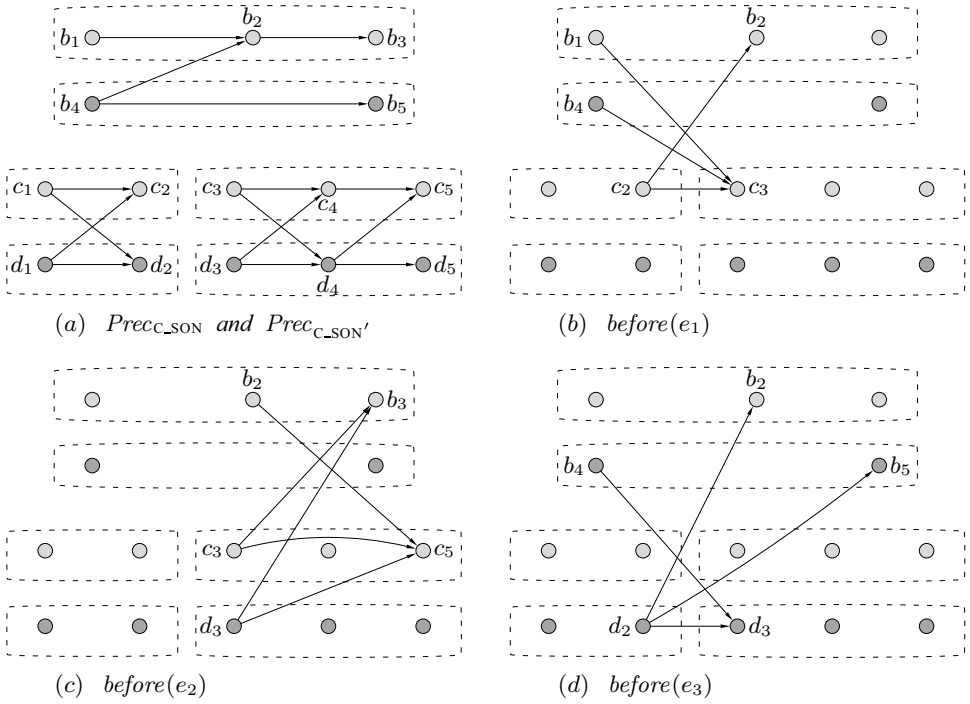


Fig. 15. Causal relationships induced by the component communication structured occurrence nets (a), and those induced by the events representing a progression from one phase of evolution to the next (b,c,d), for the behavioural structured occurrence net of Figure 14.

Proposition 3. *If Cut is a cut of the* B_SON *as in Definition 7, then*

1. *$Cut \cap \mathbf{C}'$ is a cut of* C_SON′.
2. *$Cut \cap C_i$ is either $\varnothing$ or a cut of* ON_i*, for every $i \leq k$.* ◇

Note that the empty set in the second projection is due, e.g., to the fact that some lower level subsystems may be inactive (active) in a global state represented by the cut, and become active (resp. inactive) after some system modification.

Next we describe valid executions of a behavioural structured occurrence net.

Definition 8 (step execution of behavioural SON). *A* step execution *of the* B_SON *as in Definition 7 is a sequence $\chi \stackrel{\text{df}}{=} D_0\, G_1\, D_1 \ldots G_n\, D_n$ ($n \geq 0$), where each $D_i \subseteq \mathbf{C} \cup \mathbf{C}'$ is a set of conditions and each $G_i \subseteq \mathbf{E} \cup \mathbf{E}'$ is a set of events, such that $D_0 = Init_{\text{B\_SON}}$ and, for every $i \leq n$:*

- $pre(G_i) \cup max_i \subseteq D_{i-1}$;
- $(e, f) \in \kappa \cup \kappa'$ *and* $f \in G_i$ *implies* $e \in \bigcup_{j \leq i} G_j$;
- $D_i = (D_{i-1} \setminus (pre(G_i) \cup max_i)) \cup post(G_i) \cup min_i$;

where $min_i \stackrel{\text{df}}{=} \bigcup\{Min_{\Pi(c')} \mid c' \in post(\mathbf{E}' \cap G_i)\}$ and $max_i \stackrel{\text{df}}{=} \bigcup\{Max_{\Pi(c')} \mid c' \in pre(\mathbf{E}' \cap G_i)\}$.
We also say that the step execution χ leads *to the cut D_n, and that D_n is* reachable. ◇

The above definition, which contains requirements present in the two previous definitions of step executions, take cares of the fact that moving between the phases (in the system being abstracted) and the corresponding events (in the abstracted system) has to be synchronised.

The following is a possible step execution for the net of Figure 14 leading from the initial to final cut:

$$\chi \;=\; \{b_1, b_4, c_1, d_1\}\,\{f_1, g_1\}\,\{b_1, b_4, c_2, d_2\}\,\{e_3\}\,\{b_1, b_5, c_2, d_3\}\,\{e_1\}\,\{b_2, b_5, c_3, d_3\}\\ \{f_2, g_2\}\,\{b_2, b_5, c_4, d_4\}\,\{e_2, g_3\}\,\{b_3, b_5, c_4, d_5\}\,\{f_3\}\,\{b_3, b_5, c_5, d_5\}\,.$$

This execution is illustrated graphically in Figure 16. We also observe that in this case, for example, $min_1 = max_1 = \varnothing$, $min_2 = \{d_2\}$ and $max_2 = \{d_3\}$.

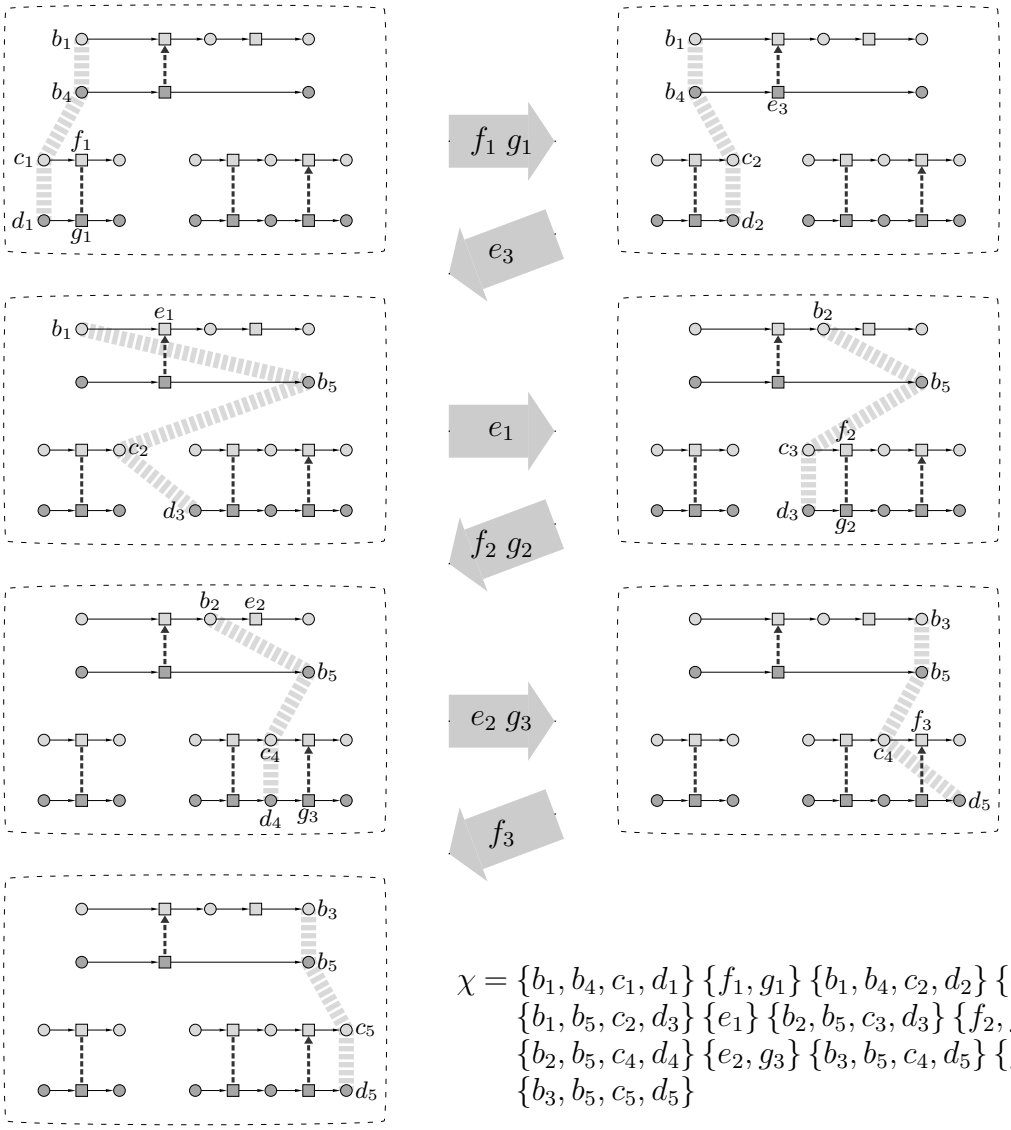


Fig. 16. A possible step execution χ leading from the initial to final cut of the behavioural structured occurrence net of Figure 14.

Our final aim in this section is to establish the consistency between the executions of a behavioural structured occurrence net, and the behaviours of the component (structured) occurrence nets.

Theorem 5. *Given the step execution as in Definition 8, the sequence*

$$D_0 \cap \mathbf{C}' \;\; G_1 \mathbf{E}' \;\; D_1 \cap \mathbf{C}' \;\; \ldots \;\; G_n \cap \mathbf{E}' \;\; D_n \cap \mathbf{C}'$$

is a step execution of ON'. *Moreover, for every* $j \leq k$, *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 either a sequence of empty sets, or a step execution of the occurrence net ON_j *possibly preceded and/or followed by a sequence of empty sets (in the former case, the first non-empty set is the initial cut of* ON_j, *and in the latter the final one).* ◇

Such a result can be seen as a direct counterpart of Theorem 3 ensuring that any behaviour of a behavioural structured occurrence net can be interpreted as a valid behaviour from the viewpoint of each component (structured) occurrence net.

Projecting the step execution χ illustrated in Figure 16 on the different (structured) occurrence nets making up the behavioural structured occurrence net in Figure 14 gives the following individual views of the execution:

$\mathrm{ON}_1 : \{c_1\}\,\{f_1\}\,\{c_2\}\,\varnothing\,\{c_2\}\,\varnothing\,\varnothing\,\varnothing\,\varnothing\,\varnothing\,\varnothing\,\varnothing\,\varnothing$
$\mathrm{ON}_2 : \{d_1\}\,\{g_1\}\,\{d_2\}\,\varnothing\,\varnothing\;\varnothing\,\varnothing\,\varnothing\,\varnothing\,\varnothing\,\varnothing\varnothing\,\varnothing$
$\mathrm{ON}_3 : \varnothing\,\varnothing\,\varnothing\,\varnothing\,\varnothing\,\varnothing\,\{c_3\}\,\{f_2\}\,\{c_4\}\,\varnothing\,\{c_4\}\,\{f_3\}\,\{c_5\}$
$\mathrm{ON}_4 : \varnothing\,\varnothing\,\varnothing\,\varnothing\,\{d_3\}\,\varnothing\,\{d_3\}\,\{g_2\}\,\{d_4\}\,\{g_3\}\,\{d_5\}\,\varnothing\,\{d_5\}$
$\mathrm{ON}' : \{b_1, b_4\}\,\varnothing\,\{b_1, b_4\}\,\{e_3\}\,\{b_1, b_5\}\,\{e_1\}\,\{b_2, b_5\}\,\varnothing\,\{b_2, b_5\}\,\{e_2\}\,\{b_3, b_5\}\,\varnothing\,\{b_3, b_5\}\,.$

The last result in this section is basically a re-statement of Theorem 4 for the case of behavioural structured occurrence nets.

Theorem 6. *Given the step execution as in Definition 8, we have that: each* D_i *is a cut of* B_SON*; no event occurs more than once; and* $D_n = \mathit{Fin}_{\mathrm{B\_SON}}$ iff *each event of* $\mathbf{E} \cup \mathbf{E}'$ *occurs in the execution. Moreover, each cut of* B_SON *can be reached from the initial cut through some step execution, and there is a step execution involving all the events in* $\mathbf{E} \cup \mathbf{E}'$. ◇

In this section we introduced structured occurrence nets capturing an abstraction mechanism between different representations of the same system. We have also demonstrated its soundness through showing that the resulting structured representation retains the desirable properties of the basic occurrence net model.

5 Spatial and Temporal Abstractions

What we will call *spatial abstraction* is based on the relation 'contains/is component of'. Figure 17 shows the behaviour of a system and of the three systems of which it is composed, and how its behaviour is related to that of these components. (This figure does not represent the matter of *how*, or indeed whether, the component systems are enabled to communicate, i.e., what design is used, or what connectors are involved.) Having identified such a set of systems, and hence the *containing* system which they make up, then each member of this set has the other members as its *environment*.

Definition 9 (spatial abstraction SON). *Let* C_SON *be as in Definition 5, let*

$$\mathrm{C\_SON}' = (\mathrm{ON}_1^1, \ldots, \mathrm{ON}_{m_1}^1, \ldots, \mathrm{ON}_1^k, \ldots, \mathrm{ON}_{m_k}^k, \kappa')$$

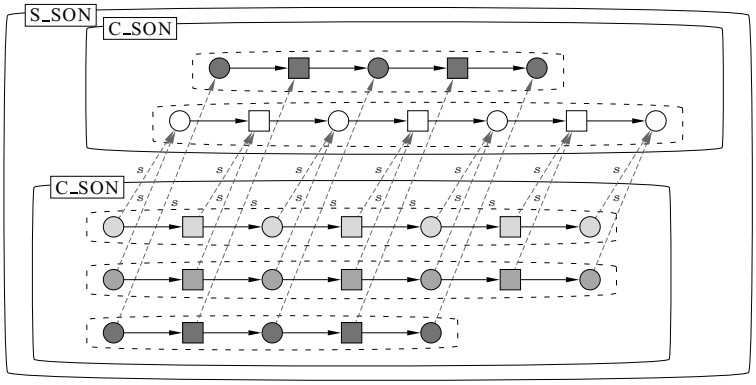


Fig. 17. System composition. The 'spatially-abstracts' relation is indicated by s-labelled edges.

be another communication structured occurrence net ($m_i \geq 1$ for $i \leq k$), and let $\mathbf{C}_i$ and $\mathbf{E}_i$ be respectively all the conditions and events in the occurrence nets $\text{ON}^i_1, \ldots, \text{ON}^i_{m_i}$, for $i \leq k$. Moreover, let $\varsigma : \mathbf{C}' \cup \mathbf{E}' \to \mathbf{C} \cup \mathbf{E}$ be a mapping from the nodes of C_SON′ *to the nodes of* C_SON.
A spatial abstraction structured occurrence net *is a tuple* $\text{S\_SON} \stackrel{\text{df}}{=} (\text{C\_SON}, \text{C\_SON}', \varsigma)$ *such that the following hold:*

1. $\varsigma(\mathbf{C}_i) = C_i$, $\varsigma(\mathbf{E}_i) = E_i$ *and* $\varsigma^{-1}(C_i \cup E_i) = \mathbf{C}_i \cup \mathbf{E}_i$, *for every* $i \leq k$;
2. $\varsigma^{-1}(pre(e)) = pre(\varsigma^{-1}(e))$ *and* $\varsigma^{-1}(post(e)) = post(\varsigma^{-1}(e))$, *for every* $e \in \mathbf{E}$;
3. *The tuple* $\text{C\_SON}'' \stackrel{\text{df}}{=} (\text{ON}_1, \ldots, \text{ON}_k, \text{ON}^1_1, \ldots, \text{ON}^1_{m_1}, \ldots, \text{ON}^k_1, \ldots, \text{ON}^k_{m_k}, \kappa \cup \kappa' \cup \kappa'')$, *where κ'' is given by $\{(e, \varsigma(e)), (\varsigma(e), e) \mid e \in \mathbf{E}'\}$, is a communication structured occurrence net.* ◇

The idea behind the above definition is that each occurrence net ON_i is a spatial abstraction of occurrence nets $\text{ON}^i_1, \ldots, \text{ON}^i_{m_i}$, and that this relationship is introduced through the mapping ς (see (1) above). Intuitively, each node x of ON_i is composed of all the nodes z of $\text{ON}^i_1, \ldots, \text{ON}^i_{m_i}$ such that $\varsigma(z) = x$. (2) ensures that the direct causalities between the conditions and events of the nets being abstracted are consistent with those present in the occurrence nets ON_i. Finally, (3) ensures a consistency (acyclicity) between causalities coming from the communication structured occurrence nets, C_SON and C_SON′, after taking into account the fact that each event e in the latter is a component of event $\varsigma(e)$ in the former and so the two events are implicitly synchronised.

Figure 18 shows a spatial abstraction structured occurrence net in which, for example, event e_2 is composed of events f_2 and f_4, and condition b_5 of conditions c_5 and c_8.

A *cut* of S_SON is simply a cut of C_SON″ introduced in Definition 9(3). Similarly, the initial and final cuts of S_SON, $Init_{\text{S\_SON}}$ and $Fin_{\text{S\_SON}}$, are the initial and final cuts of C_SON″, respectively.

The spatial abstraction structured occurrence net depicted in Figure 18 has four cuts:

$$\begin{aligned}(Init_{\text{S\_SON}} =)\ &\{b_1, b_3, c_1, c_3, c_6\}\ \{b_2, b_3, c_2, c_3, c_6\}\\ &\{b_2, b_4, c_2, c_4, c_7\}\ \{b_2, b_5, c_2, c_5, c_8\}\ (= Fin_{\text{S\_SON}})\ .\end{aligned}$$

By projecting any of these cuts onto C_SON we get $\{b_1, b_3\}$ or $\{b_2, b_3\}$ or $\{b_2, b_4\}$ or $\{b_2, b_5\}$, each of which is a valid cut of C_SON.

The next result characterises the cuts of a spatial abstraction structured occurrence net.

Proposition 4. *If Cut is a cut of the* S_SON *as in Definition 9, then*

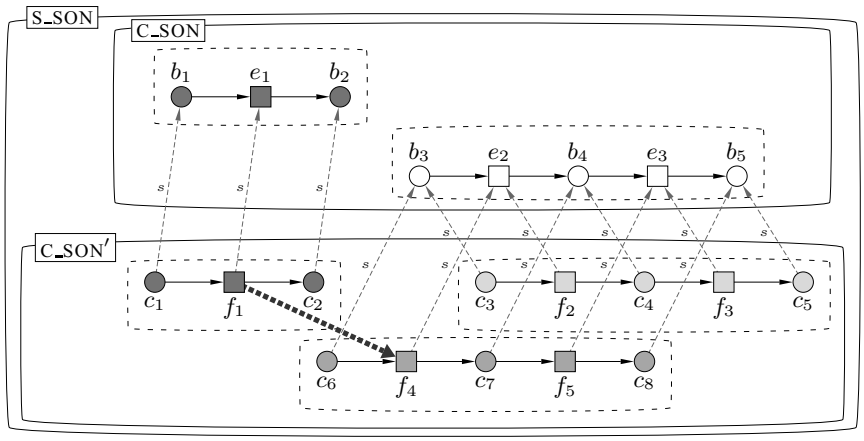


Fig. 18. A spatial abstraction structured occurrence net.

1. $Cut \cap \mathbf{C}$ *is a cut of* C_SON.
2. $Cut \cap \mathbf{C}'$ *is a cut of* C_SON′.
3. $c \in Cut$ iff $\varsigma(c) \in Cut$, *for every* $c \in \mathbf{C}'$. ◇

The first two parts establish a consistency between the cuts of S_SON and the cuts of C_SON or C_SON′. The third one is new and it can be seen as a formal capture of synchronisation between the corresponding conditions of C_SON and C_SON′.

For the spatial abstraction structured occurrence net depicted in Figure 18, we have already observed that all projections on C_SON yield valid cuts. Similarly, the projections on C_SON′ yield $\{c_1, c_3, c_6\}$ or $\{c_2, c_3, c_6\}$ or $\{c_2, c_4, c_7\}$ or $\{c_2, c_5, c_8\}$, each of which is its valid cut C_SON′. However, not all the cuts of C_SON′ can be obtained in this way, e.g., $\{c_1, c_4, c_6\}$. To illustrate Proposition 4(3), we note that for the cut $Cut = \{b_2, b_4, c_2, c_4, c_7\}$ we have the mappings $\varsigma^{-1}(b_2) = \{c_2\} \subseteq Cut$ and $\varsigma^{-1}(b_4) = \{c_4, c_7\} \subseteq Cut$.

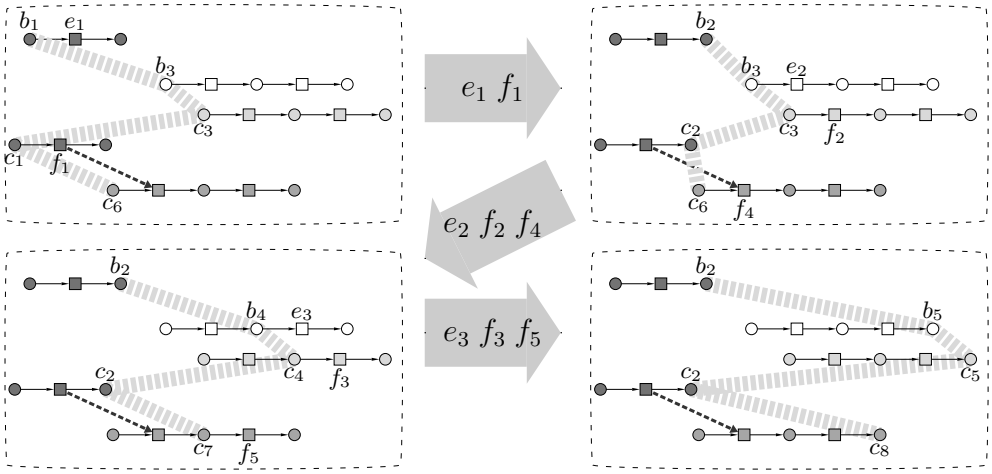
Defining possible executions of a spatial abstraction structured occurrence net incorporates a number of requirements used previously to define executions of other kinds of structured occurrence nets.

Definition 10 (step execution of spatial abstraction SON). *A* step execution *of the spatial structured occurrence net* S_SON *as in Definition 9 is a sequence* $\chi \stackrel{\text{df}}{=} D_0\, G_1\, D_1 \ldots G_n\, D_n$ *($n \geq 0$), where each $D_i \subseteq \mathbf{C} \cup \mathbf{C}'$ is a set of conditions and each $G_i \subseteq \mathbf{E} \cup \mathbf{E}'$ is a set of events, such that $D_0 = Init_{\text{S\_SON}}$ and, for every $i \leq n$:*

- $pre(G_i) \subseteq D_{i-1}$;
- $(e, f) \in \kappa \cup \kappa'$ *and* $f \in G_i$ *implies* $e \in \bigcup_{j \leq i} G_j$;
- $D_i = (D_{i-1} \setminus pre(G_i)) \cup post(G_i)$;
- $e \in G_i$ iff $\varsigma(e) \in G_i$, *for every* $e \in \mathbf{E}'$.

We also say that the step execution χ leads *to the cut D_n, and that D_n is* reachable. ◇

The only new requirement in the above definition (the last one) is that an event being abstracted and the event to which it is abstracted are always executed together.



$$\chi = \{b_1, b_3, c_1, c_3, c_6\}\,\{e_1, f_1\}\,\{b_2, b_3, c_2, c_3, c_6\}\,\{e_2, f_2, f_4\}$$
$$\{b_2, b_4, c_2, c_4, c_7\}\,\{e_3, f_3, f_5\}\,\{b_2, b_5, c_2, c_5, c_8\}$$

Fig. 19. A possible step execution χ leading from the initial to final cut of the spatial abstraction structured occurrence net of Figure 18.

The spatial abstraction structured occurrence net depicted in Figure 18 admits, e.g., the following two step executions, the first one (illustrated in Figure 19) leading from the initial to final cut:

$$\begin{aligned}\chi \;&= \{b_1, b_3, c_1, c_3, c_6\}\,\{e_1, f_1\}\,\{b_2, b_3, c_2, c_3, c_6\}\,\{e_2, f_2, f_4\}\\ &\qquad\qquad \{b_2, b_4, c_2, c_4, c_7\}\,\{e_3, f_3, f_5\}\,\{b_2, b_5, c_2, c_5, c_8\}\\ \chi' &= \{b_1, b_3, c_1, c_3, c_6\}\,\{e_1, f_1, e_2, f_2, f_4\}\,\{b_2, b_4, c_2, c_4, c_7\}\,.\end{aligned}$$

The next two results on spatial abstraction structured occurrence nets basically re-state similar results developed earlier for other kinds of structured occurrence nets.

Theorem 7. *Given the step execution as in Definition 10, the sequences*

$$D_0{\cap}\mathbf{C}\; G_1{\cap}\mathbf{E}\; D_1{\cap}\mathbf{C}\; \ldots\; G_n{\cap}\mathbf{E}\; D_n{\cap}\mathbf{C} \quad \textit{and} \quad D_0{\cap}\mathbf{C}'\; G_1{\cap}\mathbf{E}'\; D_1{\cap}\mathbf{C}'\; \ldots\; G_n{\cap}\mathbf{E}'\; D_n{\cap}\mathbf{C}'$$

are step executions of C_SON *and* C_SON′*, respectively.* ◇

Thus any behaviour of a spatial abstraction structured occurrence net can be interpreted as a valid behaviour from the viewpoint of both component communication structured occurrence nets.

Projecting the above step executions, χ and χ', of the S_SON depicted in Figure 18 on C_SON and C_SON′ gives the following individual views of the execution:

$$\begin{aligned}&\text{C\_SON} \;\&\; \chi \;: \{b_1, b_3\}\,\{e_1\}\,\{b_2, b_3\}\,\{e_2\}\,\{b_2, b_4\}\,\{e_3\}\,\{b_2, b_5\}\\ &\text{C\_SON} \;\&\; \chi' \;: \{b_1, b_3\}\,\{e_1, e_2\}\,\{b_2, b_4\}\\ &\text{C\_SON}' \;\&\; \chi \;: \{c_1, c_3, c_6\}\,\{f_1\}\,\{c_2, c_3, c_6\}\,\{f_2, f_4\}\,\{c_2, c_4, c_7\}\,\{f_3, f_5\}\,\{c_2, c_5, c_8\}\\ &\text{C\_SON}' \;\&\; \chi' : \{c_1, c_3, c_6\}\,\{f_1, f_2, f_4\}\,\{c_2, c_4, c_7\}\,.\end{aligned}$$

Each is a valid execution of the respective communication structure occurrence net.

Theorem 8. *Given the step execution as in Definition 10, we have that: each D_i is a cut of* S_SON*; no event occurs more than once; and $D_n = Fin_{\text{S\_SON}}$* iff *each event of $\mathbf{E} \cup \mathbf{E}'$ occurs in the execution. Moreover, each cut of* S_SON *can be reached from the initial cut through some step execution, and there is a step execution involving all the events in $\mathbf{E} \cup \mathbf{E}'$.* ◇

The above is, as indicated, a *spatial* abstraction — one can also have a *temporal* abstraction, as shown in Figure 20(a). When one so 'abbreviates' parts of an occurrence net one is in effect defining atomic actions, i.e., actions that appear to be instantaneous to their environment. The rules that enable one to make such abbreviations are non-trivial when multiple concurrent activities are shown in the net. These rules are best illustrated by an alternative representation for an occurrence net together with its abbreviations, namely a structured occurrence net in which each abbreviated section (or 'atomic' activity) of the net is shown surrounded by an enclosing 'event box'. Figure 20(b) shows this alternative representation of Figure 20(a), the top part of which can readily be recreated by 'collapsing' Figure 20(b)'s occurrence net, i.e., by replacing the enclosed sections by simple event symbols, as shown in Figure 20(c). This net collapsing operation is much trickier with occurrence nets that represent asynchronous activity since there is a need to avoid introducing cycles into what is meant to be an acyclic directed graph. (Hence the need, on occasion, to use synchronous system interactions.) This is the main subject of [4] and is illustrated in Figure 21.

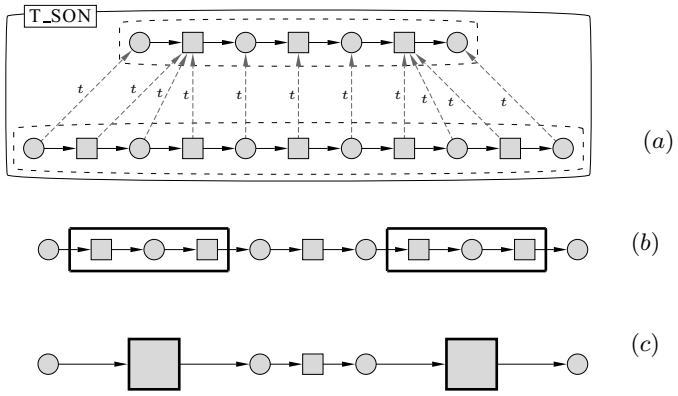


Fig. 20. System abbreviation. The 'temporally abstracts' relation is indicated by t-labelled edges (note that an edge from condition c to event e means that c is part of an atomic action which is represented by event e). (a) depicts the temporal abstraction structured occurrence net view of what has happened, whereas (b) delineates the 'collapsed' parts of behaviour, and (c) the result of such a collapsing.

Definition 11 (temporal abstraction SON). *Let* C_SON *be as in Definition 5, let* $\text{C\_SON}' = (\text{ON}'_1, \ldots, \text{ON}'_k, \kappa')$ *be another communication structured occurrence net, and let $\tau : \mathbf{C}' \cup \mathbf{E}' \to \mathbf{C} \cup \mathbf{E}$ be a mapping from the nodes of* C_SON′ *to the nodes of* C_SON.
A temporal abstraction structured occurrence net *is a tuple* $\text{T\_SON} \stackrel{\text{df}}{=} (\text{C\_SON}, \text{C\_SON}', \tau)$ *such that the following hold:*

1. $\tau(C'_i \cup E'_i) = C_i \cup E_i$, $\tau^{-1}(C_i) \subseteq C'_i$ *and* $\tau(E'_i) = E_i$, *for every* $i \leq k$;
2. $\tau^{-1}(e)$ *are disjoint blocks of the component occurrence nets of* C_SON′, *for all* $e \in \mathbf{E}$;
3. $|\tau^{-1}(c)| = 1$, *for every* $c \in \mathbf{C}$;
4. $\mathbf{F} = \{(x, y) \in (\mathbf{C} \times \mathbf{E}) \cup (\mathbf{E} \cup \mathbf{C}) \mid (\tau^{-1}(x) \times \tau^{-1}(y)) \cap \mathbf{F}' \neq \varnothing\}$;
5. $\kappa = \{(e, f) \in \mathbf{E} \times \mathbf{E} \mid (\tau^{-1}(e) \times \tau^{-1}(f)) \cap \kappa' \neq \varnothing\}$. ◇

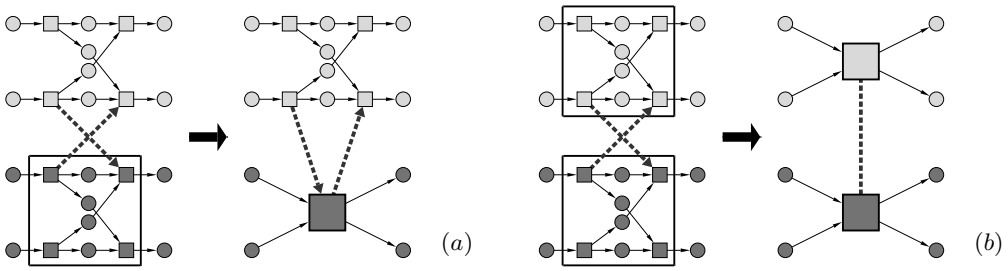


Fig. 21. Two valid collapsings which give rise to asynchronous (in (a)) and synchronous (in (b)) communication between abstract events.

The intuitive meaning of the above definition is that all the events and some of the conditions are grouped in C_SON′ into disjoint blocks (note that an individual transition constitutes a block), and then each such block (which, in general, contains both conditions and events) is collapsed into a single event of C_SON (1,2). The connectivity between new events and those conditions which survived collapsing is inherited from the events which have been collapsed (4), as is the causality relation between newly created events (5). Note that it is implicitly assumed that the conditions which survived collapsing are renamed to preserve disjointness of the two component communication structured occurrence nets (3).

The soundness of the temporal abstraction cannot be captured in the same way as for the previous structured occurrence nets as a single (instantaneous) collapsed event can correspond to a long sequence of sequentially executed events. In this case, therefore, we proceed by directly relating the cuts and step executions of the two communication structured occurrence nets.

For the temporal abstraction structured occurrence net of Figure 22, there are 13 cuts in C_SON′:

$$\{b_1,b_8\}\ \{b_2,b_3,b_8\}\ \{b_2,b_5,b_8\}\ \{b_2,b_7,b_8\}\ \{b_2,b_7,b_9\}\ \{b_3,b_4,b_8\}\ \{b_4,b_5,b_8\}$$
$$\{b_4,b_7,b_8\}\ \{b_4,b_7,b_9\}\ \{b_3,b_6,b_8\}\ \{b_5,b_6,b_8\}\ \{b_6,b_7,b_8\}\ \{b_6,b_7,b_9\}\,.$$

Four of them are mapped by τ into sets of conditions:

$$\{b_1,b_8\}\ \{b_4,b_5,b_8\}\ \{b_6,b_7,b_8\}\ \{b_6,b_7,b_9\}$$

and the result is in each case a cut of C_SON:

$$\{c_1,c_6\}\ \{c_2,c_3,c_6\}\ \{c_4,c_5,c_6\}\ \{c_4,c_5,c_7\}\,.$$

It turns out that the cuts of C_SON correspond directly to the cuts of C_SON′.

Proposition 5. *If Cut is a cut of* C_SON *then* $\tau^{-1}(Cut)$ *is a cut of* C_SON′. ◇

A practical way in which temporal abstraction might be used is to analyse the behaviour at the higher level of abstraction, which can be done more efficiently, and after finding a problem mapping it to a corresponding behaviour at the lower level (and possibly continuing the analysis there). To give a flavour of the kind of result which would provide an support for this approach, we have the following.

Theorem 9. *Let* T_SON *be the temporal abstraction structured occurrence net as in Definition 11, and* $D_0\,G_1\,D_1\ldots D_{n-1}\,G_n\,D_n$ *be a step execution of* C_SON. *Then there is a step execution of* C_SON′,

$$D'_0\,G_1^1\,D_1^1\,\ldots\,G_1^{m_1}\,D_1^{m_1}\ldots D_{n-1}^{m_{n-1}}\,G_n^1\,D_n^1\,\ldots\,G_n^{m_n}\,D_n^{m_n}\ ,$$

such that $D'_0=\tau^{-1}(D_0)$ *and* $D_i^{m_i}=\tau^{-1}(D_i)$ *and* $G_i^1\cup\ldots\cup G_i^{m_i}=\tau^{-1}(G_i)\cap\mathbf{E}'$, *for all* $i\le n$ *and* $j\le m_i$. ◇

The above result means that any step execution in the abstract level can be re-interpreted as a valid step execution in the lower level. Thus, for example, errors detected for the abstract representation may be used in the analysis of the original system.

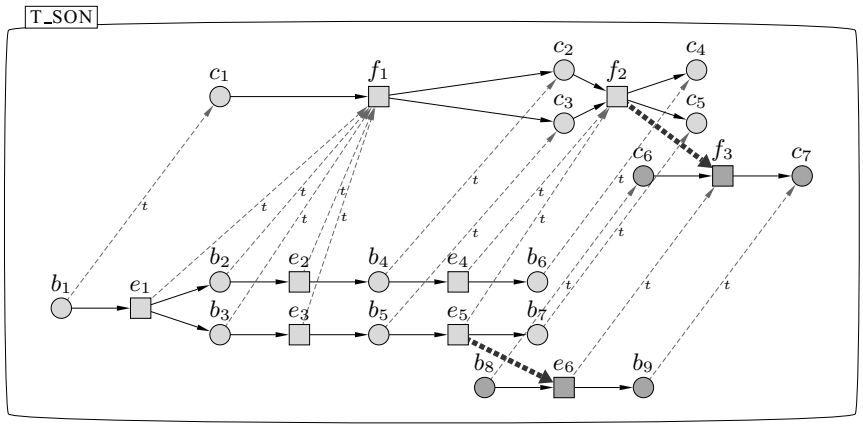


Fig. 22. A temporal abstraction structured occurrence net. For clarity, bounding boxes have been removed.

For the temporal abstraction structured occurrence net of Figure 22, consider the following step execution of C_SON:

$$\{c_1, c_6\}\,\{f_1\}\,\{c_2, c_3, c_6\}\,\{f_2, f_3\}\,\{c_4, c_5, c_7\}\,.$$

It corresponds, e.g., to following step execution of C_SON′:

$$\underbrace{\{b_1, b_8\}}_{\{c_1,c_6\}}\ \underbrace{\{e_1\}\,\{b_2, b_3, b_8\}\,\{e_2, e_3\}\,\{b_4, b_5, b_8\}}_{\{f_1\}\,\{c_2,c_3,c_6\}}\ \underbrace{\{e_4\}\,\{b_6, b_5, b_8\}\,\{e_5, e_6\}\,\{b_6, b_7, b_9\}}_{\{f_2,f_3\}\,\{c_4,c_5,c_7\}}$$

In this section we have presented composition and abbreviation, i.e., spatial and temporal abstraction, as though they are quite separate — in practice, it is likely that useful abstractions will result from successive applications of both spatial *and* temporal abstractions.

We envisage that abstraction mechanisms described in this section will be particularly useful in improving the efficiency of verification techniques based on structured occurrence nets. A possible step of achieving this would be to exploit the structuring of the execution histories allowing, e.g., to carry out separate checks on those fragments which correspond to abbreviated behaviours, and then feeding the results to the final stage of verification.

6 Information retention and judgement

We now introduce the idea of one occurrence net *retaining* information about another occurrence net that is representing the activity of some given system. This could be, for example, in order to provide fault tolerance in the form of 'recovery points', enabling the given system to fall back and restart in order that failure might be averted, or to enable the post hoc assessment of the given system's activities by some separate 'judgmental' system, tasked with trying to determine whether and why a system failed.

A simple example of state retention aimed at supporting recovery points is shown in Figure 23(a). The upper system is shown as initially acquiring (through its first event) information about an initial fragment

of the activity of the lower system, but then going on and retaining more information about this system. Intuitively, each event of the upper system is supposed to describe a recovery point given by the cut made out of the conditions of a cut in the lower system.

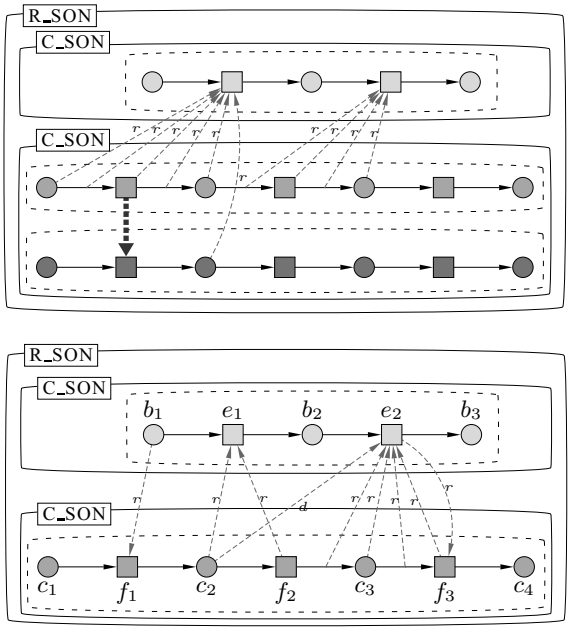


Fig. 23. State retention. The 'retains' and 'discards' relations are indicated by r-labelled and d-labelled edges, respectively. (a) shows how the upper systems accumulates retained information about the lower system, and (b) shows two systems which retain (and also discard) information about each other.

Definition 12 (state retention SON). *Let* C_SON$_1, \ldots,$ C_SON$_m$ *($m \geq 2$) be communication structured occurrence nets, with $\widehat{\mathbf{C}}$, $\widehat{\mathbf{E}}$, $\widehat{\mathbf{F}}$ and $\widehat{\kappa}$ respectively denoting all their conditions, events, flow arcs and causal arcs (collectively referred to as* elements*). Moreover, let ρ and δ be two disjoint relations included in $(\widehat{\mathbf{C}} \cup \widehat{\mathbf{E}} \cup \widehat{\mathbf{F}} \cup \widehat{\kappa}) \times \widehat{\mathbf{E}}$ which are assumed to relate elements coming from different* C_SON$_i$*'s.*
We will denote by κ the set of all pairs of events $(e, f) \in \widehat{\mathbf{E}} \times \widehat{\mathbf{E}}$ such that: $(e, f) \in \rho \cup \delta$ or there is a condition $c \in post(e)$ such that $(c, f) \in \rho \cup \delta$ or there is an arc z adjacent to e satisfying $(z, f) \in \rho \cup \delta$. Then, for every event $e \in \widehat{\mathbf{E}}$, the sets $\widehat{Pre}(e)$ and $\widehat{Post}(e)$ respectively comprise all conditions c satisfying $(c, e) \in \widehat{\mathbf{F}} \circ (\widehat{\kappa} \cup \kappa)^*$ and $(e, c) \in (\widehat{\kappa} \cup \kappa)^* \circ \widehat{\mathbf{F}}$. Also, for every event e in $\widehat{\mathbf{E}}$,

$$info(e) \stackrel{\text{df}}{=} \{z \mid (z, e) \in \rho\} \cup \bigcup \{info(f) \mid (f, e) \in \widehat{\mathbf{F}} \circ \widehat{\mathbf{F}}\} \setminus \{z \mid (z, e) \in \delta\} \ .$$

A state retention structured occurrence net *is a tuple* R_SON $\stackrel{\text{df}}{=}$ (C_SON$_1, \ldots,$ C_SON$_m, \rho, \delta$) *such that:*

1. *The relation $Prec_{\text{R\_SON}} \stackrel{\text{df}}{=} \bigcup_{e \in \widehat{\mathbf{E}}} \widehat{Pre}(e) \times \widehat{Post}(e)$ is acyclic.*
2. *If $(x, y) \in info(e)$ then $x, y \in info(e)$.*
3. *If $(z, e) \in \delta$ then $z \in info(f)$ for some f satisfying $(f, e) \in \widehat{\mathbf{F}} \circ \widehat{\mathbf{F}}$.* $\diamond$

In the above definition, each communication structured occurrence net C_SON$_i$ represents a system's evolution for which some information is being retained during the evolutions of other systems. The relation

ρ formally captures this, and $(z, e) \in \rho$ means that the information about z has been acquired during the execution of e. This information does not need to be complete (indeed, there may even be 'gaps' in the retained information, as in the example in Figure 23(a)), and the only requirement is that having information about an arc implies that the arc's endpoints are also known (2). This 'knowing' is assumed to be cumulative, i.e., if f is a predecessor of e then all information acquired during the activation of f is retained and available at e as well — see the definition of $info(e)$, which is a well-defined notion as $\widehat{\mathbf{F}}$ is a finite partial order relation. Another relation, δ, is used to model the 'deletion' of previously acquired knowledge introduced, as illustrated with d-labelled edges in Figure 23. It is assumed that one cannot at the same time retain and delete the same item of information (i.e., the relations ρ and δ are disjoint), and one only can delete an item of information if that is already known (3). Note that deleting an information about a node implies that the knowledge about adjacent arcs disappears as well — see the definition of $info(e)$ and (2).

Also, it is worth stressing that we do not assume that if z preceded z' in C_SON$_i$ then the information about the former was necessarily acquired before z'. However we still have a general acyclicity requirement (1).

Figure 24 shows the relation κ and $Prec_{\text{R\_SON}}$ for the state retention structured occurrence net of Figure 23(b). Moreover, we have the following:

$$\begin{array}{ll} info(f_1) = \{b_1\} & info(e_1) = \{c_2, f_2\} \\ info(f_2) = \{b_1\} & info(e_2) = \{c_3, f_2, f_3, (f_2, c_3), (c_3, f_3)\} \\ info(f_3) = \{b_1, e_2\}\ . & \end{array}$$

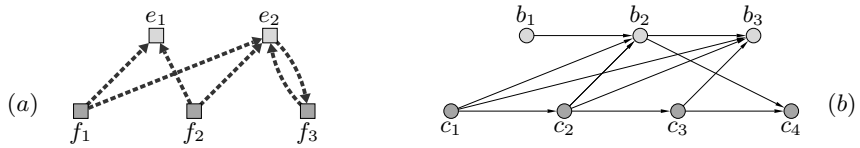


Fig. 24. (a) shows the auxiliary relation κ, and (b) the causality relation $Prec_{\text{R\_SON}}$ for the state retention structured occurrence net of Figure 23(b).

A *cut* of R_SON is a maximal (w.r.t. set inclusion) set of conditions $Cut \subseteq \widehat{\mathbf{C}}$ such that no two conditions in Cut are related by $Prec^{+}_{\text{R\_SON}}$. The initial cut of R_SON, $Init_{\text{R\_SON}}$, is the union of the initial cuts of all the C_SON$_i$'s, and the final cut, $Fin_{\text{R\_SON}}$, is the union of the final cuts of all the C_SON$_i$'s.

There are 5 cuts of the state retention structured occurrence net of Figure 23(b) (see also Figure 24(b)), namely $Cut_1 = \{b_1, c_1\}$ (initial), $Cut_2 = \{b_1, c_2\}$, $Cut_3 = \{b_1, c_3\}$, $Cut_5 = \{b_2, c_3\}$ and $Cut_5 = \{b_3, c_4\}$ (final).

Proposition 6. *If Cut is a cut of* R_SON*, then* $Cut \cap \mathbf{C}_i$ *is a cut of* C_SON$_i$*, for every* $i \leq m$. ◇

The above result, as on the previous occasions, provides a formal capture of consistency between the global and local views on a system's state. Next, for the last time, we re-define the notion of a step execution.

Definition 13 (step execution of state retention SON). *A* step execution *of the* R_SON *as in Definition 5 is a sequence* $\chi \stackrel{\text{df}}{=} D_0\, G_1\, D_1 \ldots G_n\, D_n$ *($n \geq 0$), where each* $D_i \subseteq \widehat{\mathbf{C}}$ *is a set of conditions and each* $G_i \subseteq \widehat{\mathbf{E}}$ *is a set of events, such that* $D_0 = Init_{\text{R\_SON}}$ *and, for every* $i \leq n$*:*

- $pre(G_i) \subseteq D_{i-1}$ *and* $D_i = (D_{i-1} \setminus pre(G_i)) \cup post(G_i)$*;*
- $(e, f) \in \widehat{\kappa} \cup \kappa$ *and* $f \in G_i$ *implies* $e \in \bigcup_{j \leq i} G_j$*.*

We also say that the step execution χ leads *to the cut* D_n*, and that* D_n *is* reachable. ◇

Figure 25 illustrates a step execution $\chi = \{b_1, c_1\}\,\{f_1\}\,\{b_1, c_2\}\,\{e_1, f_2\}\,\{b_2, c_3\}$ of the state retention structured occurrence net of Figure 23(b).

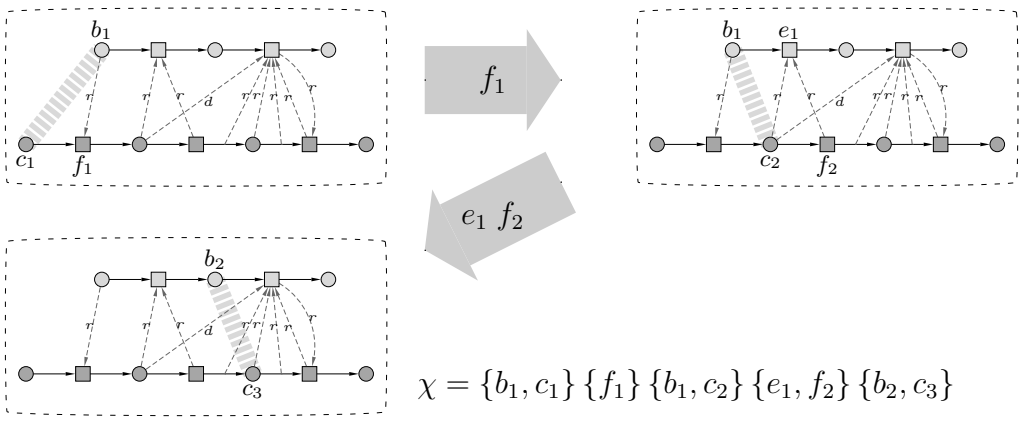


Fig. 25. A possible step execution χ of the state retention structured occurrence net of Figure 23(b).

We end with two results testifying to the soundness of the notion of a state retention structured occurrence net, formulated similarly as before in this paper.

Theorem 10. *Given the step execution as in Definition 13, each sequence*

$$D_0 \cap \mathbf{C}_i \; G_1 \cap \mathbf{E}_i \; D_1 \cap \mathbf{C}_i \; \ldots \; G_n \cap \mathbf{E}_i \; D_n \cap \mathbf{C}_i$$

is a step execution of C_SON_i. ◇

Theorem 11. *Given the step execution as in Definition 13, we have that: each* D_i *is a cut of* R_SON*; no event occurs more than once; and* $D_n = \mathit{Fin}_{\text{R\_SON}}$ iff *each event of* $\widehat{\mathbf{E}}$ *occurs in the execution. Moreover, each cut of* R_SON *can be reached from the initial cut through some step execution, and there is a step execution involving all the events in* $\widehat{\mathbf{E}}$. ◇

As already indicated, the notion of a 'failure' event involves, in principle, three systems — the given (possibly failing) system, its environment, and a judging system. This judging system may interact directly and immediately with the given system, in which case it is part of the system's environment, e.g., in VLSI an on-chip testing facility [14]; another example, in a very different world, is a football referee! Alternatively the judging system may be deployed after the fact using an occurrence net that represents how the failing event occurred. Figure 26 is an attempt to portray this. It deliberately represents a situation in which a judgement system has obtained and retained only incomplete evidence of the states and events, and even the causal relationships between conditions and events, of two interacting systems (each of which constitutes the other's environment).

The judgment system is shown as having gradually accumulated information about the two systems, and then used this information (together with any other relevant information that is available concerning, e.g., specifications, contracts and laws) to help reach a judgement as to whether a particular event was erroneous, and a failure has occurred. Such 'retained' information might have been obtained directly by observation of an actual system and its environment, or may be little more than guesswork about the given system's presumed activity.

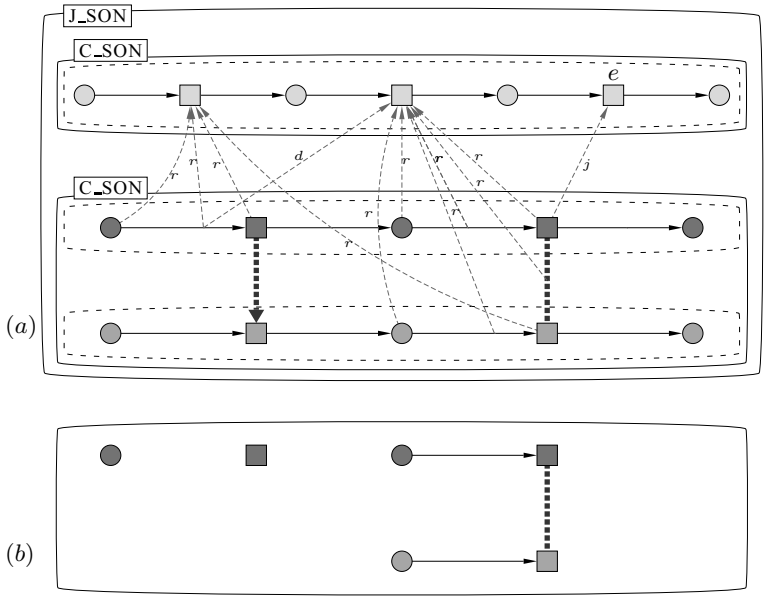


Fig. 26. (a) shows post-hoc judgement involving a judgemental system (upper part) and an active system together with its environment (lower part). The 'retains', 'deletes' and 'judges' relations are indicated by r-labelled, d-labelled and j-labelled edges, respectively. (b) shows the part of the active system behaviour given by $info(e)$ on the basis of which judgement was made.

Definition 14 (judgement SON). *Let* R_SON *be the state retention structured occurrence net as in Definition 12, and* ι *be a relation comprising pairs of events coming from different* C_SON$_i$*'s. Moreover, for every event* e*, let* $\widetilde{Pre}(e)$ *and* $\widetilde{Post}(e)$ *be sets respectively comprising all conditions* c *satisfying* $(c,e) \in \widehat{\mathbf{F}} \circ (\widehat{\kappa} \cup \kappa \cup \iota)^*$ *and* $(e,c) \in (\widehat{\kappa} \cup \kappa \cup \iota)^* \circ \widehat{\mathbf{F}}$.
A judgement structured occurrence net *is a tuple* J_SON $\stackrel{\text{df}}{=} (\text{C\_SON}_1, \ldots, \text{C\_SON}_m, \rho, \delta, \iota)$ *such that:*

1. *The relation* $Prec_{\text{J\_SON}} \stackrel{\text{df}}{=} \bigcup_{e \in \widehat{\mathbf{E}}} \widetilde{Pre}(e) \times \widetilde{Post}(e)$ *is acyclic.*
2. *If* $(e,f) \in \iota$ *then* $e \in info(f)$. ◇

The relationship $(e,f) \in \iota$ represents an act of judging an event e through the event f of the judging system. For this to be valid, e must be known at f, and so we assume that $e \in info(f)$ using notation introduced in Definition 12.

Formal definitions of cuts and step executions as well as the basic properties of judgement structured occurrence nets are very similar to those already presented for retention structured occurrence nets, and are therefore omitted.

Retracing the 'fault-error-failure' chain, after a judgment has been made that a particular event needs to be regarded as a failure, involves following causal arrows *in either direction* within a given occurrence net, and following relations so as to move from one occurrence net to another. Thus one could retrace (i) the source and/or consequence of an interaction between systems, (ii) from a system to some guilty component(s), (iii) from a component to the system(s) built from it, or (iv) from a given system to the system(s) that created or modified it. All this tracing activity could be undertaken by some tracing system (perhaps a part of the judgement system) using whatever evidence is available (e.g., a retained occurrence net which is alleged to record what happened). This tracing system (just like a judgment system) can of course itself fail (in the eyes of some other judgment system)! The actual implementation of such tracing

in situations of ongoing activity, and of potential further failures, e.g., such as interfering with witnesses and the jury (in a judicial context), involves problems such as those addressed by the *chase protocols* [20].

7 Applications of Structured Occurrence Nets

Structured occurrence nets have a number of different potential applications. For example, they could we believe be used to extend the capabilities of (i) existing occurrence net based model-checking approaches to system evaluation [10], and (ii) existing tools for the automated synthesis of systems (e.g., VLSI designs) from exemplar occurrence nets [12]. Such applications involve fully-detailed structured occurrence nets (produced either from actual systems, or from models of intended systems). However, we first discuss the use of structured occurrence nets that were created after the fact from whatever evidence was available (and that as a consequence are likely to be far from complete) to assist the task of analyzing (accidental or malicious) failures in large complex evolving systems, possibly involving software, hardware and people.

One can envisage a given judge, having identified some system event as a failure, analysing a structured occurrence net, i.e., a set of related occurrence nets (dealing with the various abstractions of the various relevant systems), in an attempt to identify (i) the fault(s) that should be blamed for the failure, and/or (ii) the erroneous states that could and should be corrected or compensated for. Unless we assume that the occurrence nets are recorded correctly and completely as an automated by-product of system activity, in undertaking such a task it may well prove appropriate during such an analysis to correct or add to the occurrence nets, both individually and as a set, based on additional evidence or assumptions about what occurred.

Different judges (even different automated judgement systems) could of course, even if they identify the same failure event, come to different decisions regarding what actually happened and in determining the related faults and errors — possibly because they use different additional information (e.g., assumptions and information relating to system designs and specifications) to augment the information provided by the occurrence nets themselves. The result of such analyses could be thought of as involving the marking-up of the set of occurrence nets so as to indicate a four-way classification of all their places, namely as ‘Erroneous’, ‘Correct’, ‘Undecided’, and ‘Not considered’.

As indicated earlier, the production of such a classification is likely to involve repeated partial traversals of the occurrence nets, following causal arrows backwards within a given occurrence net in a search for causes and forwards in a search for consequences. In addition it will involve following relations so as to move from one occurrence net to another. A simplistic example of this is the recognition that a given system’s behaviour had, after a period of correct operation, started to exhibit a succession of errors might lead to investigating the related occurrence net representing the system’s evolution to determine if it had suffered a modification at the relevant time.

This way of describing the failure analysis task using occurrence nets might be regarded as essentially metaphorical, i.e., essentially just as a way of describing (semi)-formally what is currently often done by expert investigators in the aftermath of a major system failure. However, at the other extreme one can imagine attempting to automate the recording and analysis of actual occurrence nets — indeed one could argue that this is likely to be a necessary function of any complex system that really merited the currently fashionable appellations ‘self-healing’ and ‘autonomic’. The more likely, and practical, possibility — one that we plan to investigate — is the provision of computer assistance for the tasks of representing, checking the legality of, and performing analyses of, structured occurrence nets. This is because the task of analysing and/or deriving the scenarios depicted by structured occurrence nets will, in real life, be too complex to be undertaken as a simple paper and pencil exercise. The main reason is that the systems we primarily aim at are (highly) concurrent — thus automated analyses of their behaviour suffer from the so-called ‘state explosion problem’. In a nutshell, even the most basic problems are then of non-polynomial complexity and so perhaps the only way to deal with them is to use highly optimised automated tools. This work could build on work in, e.g., [6, 10–12, 19], on the *unfoldings* of Petri nets introduced in [18]. For

example, Theorems 4, 6, 8 and 11 establish that the reachable global states of structured occurrence nets are precisely all their cuts, and so one might try to solve the corresponding complex reachability problem using a model checker based on a SAT-solver or integer programming, e.g., as in [5, 10, 12]. Another possibility follows from Theorem 9 which can offer a reduction of the complexity of model checking as any potentially inadmissible step execution in the abstract level can be re-interpreted as a valid step execution in the lower level. Other results which hint at the possibility of using the structuring information in order to alleviate the verification effort are Propositions 2 3, 4, 5 and 6 and Theorems 3, 5, 7 and 10.

Turning to the other types of application that we envisage, the utilisation of structured occurrence nets for system evaluation and synthesis is a more straightforward extension of existing research, and of existing proven tools. They could be used as a way of modelling complex system behaviour *prior* to system deployment, so as to facilitate the use of some form of automated model-checking in order to verify at least some aspects of the design of the system(s). Alternatively such automated model-checking might be used to assist analysis of the automatically-generated records of actual failures of complex systems. Such work could take good advantage of recent work on the model-checking of designs, originally expressed in the pi-calculus, work which involves the automated generation and analysis of occurrence nets [11].

System synthesis is yet another very different avenue that could be usefully explored. This would involve using structured occurrence nets which have been shown to exhibit desirable behaviour, including automated tolerance and/or diagnosis of faults, as an aid to designing systems that are guaranteed to exhibit such behaviour when deployed. We have in fact, with colleagues, already shown that it is possible to synthesise asynchronous VLSI sub-systems via the use of formal representations based on occurrence nets [12], but such designs are much less complex than those that we have had in mind while developing the concept of structured occurrence nets. The use of structured occurrence nets, in particular those consisting of a set of interacting occurrence nets, as input to an enhanced synthesiser is in effect a means of allowing the user to exercise a degree of control over the synthesis process. In effect the structuring is being used to constrain the synthesiser's search space. By such means the user could cooperate with the synthesiser so as to produce solutions to problems of a complexity that exceeds the practical limits of existing synthesis tools [12].

8 Concluding Remarks

A major aim of the present paper has been to introduce, and motivate the further study of, the concept that we term structured occurrence nets, a concept that we claim could serve as a basis for possible improved techniques of failure prevention and analysis of complex evolving systems. This is because the various types of abstractions that the concept of a structured occurrence nets make use of are all ones that we suggest could facilitate the task of understanding complex systems and their possible or actual failures, and that of the analysis of the cause(s) of such failures. These abstractions would in most cases be a natural consequence of the way the system(s) have been conceived and perceived, rather than abstractions that have to be generated after the fact, during analysis.

In principle, a single huge conventional (i.e., unstructured) occurrence net could be used to represent the activity of a large evolving set of complex systems. However, by retaining structuring which matches the actual or perceived reality of a set of distinct systems and their largely distinct activities, the challenge of creating, understanding and validating the various systems' evolution and behaviour (and their failures) is greatly reduced. In particular, we believe it will prove possible to design automated SON analysis tools that take advantage of all the retained structure, and as a result are capable of dealing with much more complex system activities than could be handled by existing tools for analysing conventional occurrence nets. (Such tools could we hope be built as extensions of existing tools for supporting the analysis of conventional unstructured occurrence nets.)

Perhaps the most straightforward use to which structured occurrence nets could be put is for analysing fully detailed algorithmic models of a set of systems. This is because such models could be used to generate

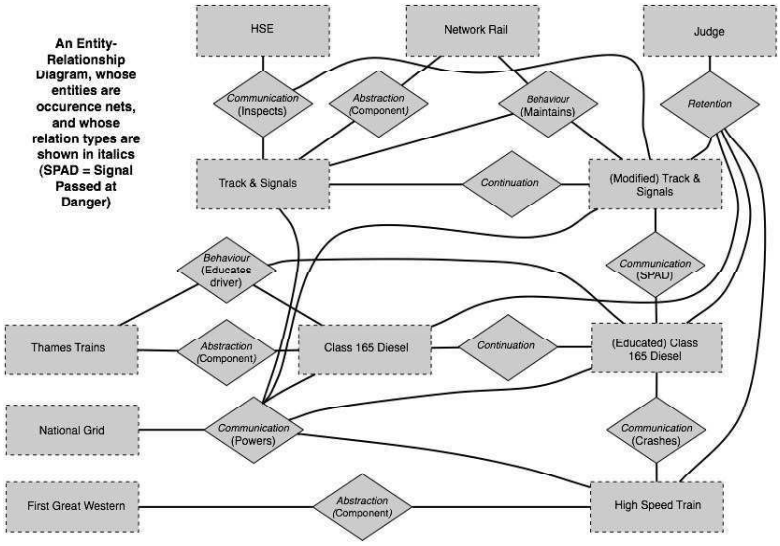


Fig. 27. Entity-Relationship diagram for the Ladbroke Grove Train crash.

structured occurrence sets that would be known to be complete and accurate (as opposed to being in large part the fruit of speculation and guesswork, which may well be the case for forensic investigations and safety analyses). Such detailed structured nets can then be used for an enhanced version of conventional model-checking [5] in order to establish various formal properties of the set of systems, taking advantage of the structuring to enable more complex systems to be analysed than would be feasible with conventional techniques.

Of the various possible types of use of structured occurrence net that we have identified — post hoc analysis of partially-understood systems, a priori analysis of detailed models of fully specified systems, and synthesis of systems from exemplar occurrence nets — we have, ahead of the availability of any tool support, so far initiated exploration of just the first. We have been working on a sketch of a structured representation of the various activities and mistakes which led up to the tragic Ladbroke Grove Train crash [24]. Figure 27 is an example result of this exploration — it uses the conventional Entity-Relationship graphical notation, the entities in fact being individual (un-detailed) occurrence nets, representing the existence of information about the activities of each of the trains that collided, the organisations responsible for train maintenance and inspection, the organisation that designed the signalling system, the organisation that was supposed to educate new drivers about the detailed location and operation of the signals, etc. In doing so, we made use of such SON relationships as communication, abstraction and behaviour. However, in our view tool support is needed in order for us to take such experiments on much further, and our main next priority is to explore the provision of such support.

Such a tool for recording, analysing and manipulating structured occurrence nets is best regarded as constituting a somewhat general purpose infrastructure, which would actually be used via a particular specialised application interface. The infrastructure tool would embody fairly general facilities for assessing and reporting on the completeness and consistency of a given structured occurrence net, using algorithms based on the various theorems and propositions given in earlier sections of this paper. The application interface could be the means by which for example (i) a structured occurrence net is constructed, and (ii) any additional information required to identify states and events that should be classified as erroneous is provided.

A possible example of a specialised application interface would be one supporting the performing of forensic analyses of extensive automatically-recorded audit trails of network traffic obtained from or about computers that are suspected of having been involved in cybercrime [7]. A related, but in practice very different, potential application concerns the evaluation of evidence from a major (conventional) crime. The aim of such evaluation is to formulate plausible scenarios worthy of further police investigation, an application for which a 'knowledge-based' modelling technique has been developed [9]. Somewhat similar in intent, though designed for aircraft accident analyses rather than criminal investigations, and very different technically, is the 'Why-Because' causal analysis scheme [15]. Our speculation is that at least in theory, and perhaps in practice, all these could benefit from the use of infrastructural support for structured occurrence nets.

Clearly, much remains to be done to explore how best to exploit the concept of structured occurrence nets, and to determine the adequacy of the set of relations that we have described in this paper. (We are considering further relations, in particular that of 'alternate', a relation which would be used when the need is to document and explore speculative alternative activities that might have led to some given situation, but have deferred discussion of this to a subsequent paper. Similarly, we have deferred discussion of how one might best extend the formalism developed here to deal with recursively defined structured occurrence nets, a challenging task for which the present paper provides the necessary groundwork.) However, we hope that the present account has demonstrated that our plans have a solid formal basis, a basis which can usefully be exploited through the provision of automated support and analysis tools, and has adequately explained why we believe such tool building activities are now justified.

Acknowledgment We are indebted to the anonymous referees whose suggestions have led to significant improvements in the content and presentation of our paper.

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. J.Esparza and K.Heljanko (2008). *Unfoldings: A Partial-Order Approach to Model Checking*. Springer.
6. J.Esparza, S.Römer and W.Vogler (2002). An Improvement of McMillan's Unfolding Algorithm. *Formal Methods in System Design* 20, 285–310.
7. P.Gladyshev and A.Patel (2005). Formalising Event Time Bounding in Digital Investigations. *International Journal of Digital Evidence* 4.
8. 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.
9. J.Keppens and B.Schafer (2006). Knowledge Based Crime Scenario Modelling. *Expert Syst. Appl.* 30, 203–222.
10. V.Khomenko and M.Koutny (2007). Verification of Bounded Petri Nets Using Integer Programming. *Formal Methods in System Design* 30, 143–176.
11. V.Khomenko, M.Koutny and A.Niaouris (2006). Applying Petri Net Unfoldings for Verification of Mobile Systems. Report CS-TR 953, Newcastle University.
12. 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.
13. H.C.M.Kleijn and M.Koutny (2004). Process Semantics of General Inhibitor Nets. *Information and Computation* 190, 18-69.
14. D.Koppad, D.Sokolov, A.Bystrov and A.Yakovlev (2006). Online Testing by Protocol Decomposition. Proc. of *IOLTS'06*, IEEE CS Press, 263–268.

15. P.B.Ladkin (2000). Causal Reasoning about Aircraft Accidents. Proc. of *SAFECOMP 2000*, LNCS 1943, 344–360.
16. S.Lenk (1994). Extended Timing Diagrams as a Specification Language. Proc. of *European Design Automation*, IEEE Computer Society Press, 28–33.
17. S.Mauw (1996). The Formalization of Message Sequence Charts. *Computer Networks and ISDN Systems* 28, 1643–1657.
18. K.L.McMillan (1995). A Technique of State Space Search Based on Unfoldings. *Formal Methods in System Design* 6, 45-65.
19. S.Melzer and S.Römer (1997). Deadlock Checking Using Net Unfoldings. Proc. of *CAV'97*, LNCS 1254, 352–363.
20. P.M.Merlin and B.Randell (1978). State Restoration in Distributed Systems. Proc. of *FTCS-8*, IEEE Computer Society Press, 129–134.
21. B.Randell and M.Koutny (2007). Failures: Their Definition, Modelling and Analysis. Proc. of *ICTAC'07*, LNCS 4711, 260–274.
22. 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.
23. F.J.Thayer, J.C.Herzog and J.D.Guttman (1999). Strand Spaces: Proving Security Protocols Correct. *Journal of Computer Security* 7, 191–230.
24. http://www.rail-reg.gov.uk/upload/pdf/incident-ladbrokegrove-ladbroke-optim.pdf

Appendix: additional definitions, results and all proofs

In the appendix we present proofs of all the results included in the main body of the paper, and of with a number of auxiliary propositions which have been omitted from the main body in order to highlight what we consider to be the key properties of structured occurrence nets. Moreover, as in Section 2, we reproduce some of the existing results on occurrence nets. First, however, we recall some standard mathematical notions and notations used throughout the paper.

The composition of two binary relations, R and R', is denoted by $R \circ R' \stackrel{\text{df}}{=} \{(x,y) \mid \exists z : (x,z) \in R \wedge (z,y) \in R'\}$, the transitive closure of R by R^+ and the reflexive transitive closure by R^*. The restriction of $R \subseteq X \times Y$ to a subset Z of $X \times Y$ is denoted by $R|_Z$. $R \subseteq X \times X$ is a partial order relation if R^+ is irreflexive. The disjoint set union is denoted by $\uplus$.

A Occurrence nets (Section 2)

A cut Cut of an occurrence net ON divides it into two subnets which overlap along this cut (and so they share exactly the conditions in Cut), $preon_{\text{ON}}(Cut) \stackrel{\text{df}}{=} (C', E', F')$ and $poston_{\text{ON}}(Cut) \stackrel{\text{df}}{=} (C'', E'', F'')$, given by:

$$\begin{array}{ll} C' \stackrel{\text{df}}{=} \{d \in C \mid \exists c \in Cut : (d,c) \in F^*\} & C'' \stackrel{\text{df}}{=} \{d \in C \mid \exists c \in Cut : (c,d) \in F^*\} \\ E' \stackrel{\text{df}}{=} \{e \in E \mid \exists c \in Cut : (e,c) \in F^*\} & E'' \stackrel{\text{df}}{=} \{e \in E \mid \exists c \in Cut : (c,e) \in F^*\} \\ F' \stackrel{\text{df}}{=} F|_{(C' \times E') \cup (E' \times C')} & F'' \stackrel{\text{df}}{=} F|_{(C'' \times E'') \cup (E'' \times C'')}\ . \end{array}$$

Intuitively, the subnet $preon_{\text{ON}}(Cut)$ is the part of ON which has been executed to reach Cut, and $poston_{\text{ON}}(Cut)$ that which can still be executed after Cut, both including Cut.

Proposition 7 (see [3]). *Let* ON$'$ *and* ON$''$ *be respectively the subnets* $preon_{\text{ON}}(Cut)$ *and* $poston_{\text{ON}}(Cut)$ *defined above.*

1. ON$'$ *and* ON$''$ *are occurrence nets such that:* $C = C' \cup C''$, $C' \cap C'' = Cut$, $E = E' \uplus E''$ *and* $F = F' \uplus F''$.
2. $Init_{\text{ON}'} = Init_{\text{ON}}$, $Fin_{\text{ON}'} = Cut = Init_{\text{ON}''}$ *and* $Fin_{\text{ON}''} = Fin_{\text{ON}}$.

3. *Given step executions, χ' and χ'', of respectively,* ON′ *and* ON″, *χ' is a step execution of* ON *and, moreover, if χ' leads to Cut then $\chi = \chi' \oslash \widehat{\chi}''$ is a step execution of* ON. ◇

Proposition 8 (see [3]). *Consider the step execution as in Definition 3, $e \in E$ and $l \leq n$.*

1. *If $e \in G_l$ and $(f, e) \in F \circ F$, then $f \in G_i$ for some $i < l$.*
2. *If $c \in post(e)$ and $c \in D_l$, then $e \in G_i$, for some $i \leq l$.*
3. *If $c \in pre(e)$ and $e \in D_l$, then $c \notin G_i$, for all $i \geq l$.* ◇

Proposition 9. *If B is the block as in Definition 4 and $c \in B \cap C$, then there are $b \in pre(B) \setminus B$ and $d \in post(B) \setminus B$ such that $(b, c) \in F^+$ and $(c, d) \in F^+$.*

Proof. By Definition 4 and the acyclicity of ON, there are events $e, f \in B$ such that $(e, c), (c, f) \in F^+$ and $pre(e) \setminus B \neq \varnothing \neq post(f) \setminus B$. It then suffices to take any $b \in pre(e) \setminus B$ and $d \in post(f) \setminus B$. □

B Communication (Section 3)

Throughout this section, we assume that the C_SON is as in Definition 5.

A useful characterisation of cuts of C_SON can be obtained using the notion of a *causal chain* of C_SON which is any sequence of its nodes

$$\lambda \stackrel{\text{df}}{=} c_0\ e_1^1 e_2^1 \ldots e_{k_1}^1\ c_1\ e_1^2 e_2^2 \ldots e_{k_2}^2\ c_2\ \ldots\ c_{m-1}\ e_1^m e_2^m \ldots e_{k_m}^m\ c_m \qquad (*)$$

where $m, k_1, \ldots, k_m \geq 1$, and all the c_i's are conditions and e_i^j's events of C_SON such that for all $i \leq m$ and $j < k_i$: $(c_{i-1}, e_1^i), (e_{k_i}^i, c_i,) \in \mathbf{F}$ and $(e_j^i, e_{j+1}^i) \in \sigma \cup \kappa$. We also say that λ is a causal chain *from* c_0 *to* c_m. Note that

$$(c_i, c_{i+1}) \in Prec_{\text{C\_SON}}, \text{ for every } i < m. \qquad (\dagger)$$

And, conversely, if c and c' are conditions, then through a straightforward construction one can see that

$$(c, c') \in Prec^+_{\text{C\_SON}} \text{ implies that there is a causal chain from } c \text{ to } c'. \qquad (\ddagger)$$

In what follows, we will call any non-empty subsequence of λ a causal chain from its first to the last node (both of which can be events).

Two of causal chains for the communication structured occurrence net in Figure 6 are: $b_1\ e_1\ f_1\ c_2\ f_2\ e_2\ b_3$ and $e_1\ f_1\ c_2\ f_2\ e_2\ f_2\ e_2\ f_2\ e_2\ b_3$.

Proposition 10. *A set Cut is a cut of* C_SON iff *it is a maximal (w.r.t. set inclusion) subset of $\mathbf{C}$ such that there is no causal chain beginning and ending with a condition in Cut.*

Proof. Follows from (†) and (‡). □

Proposition 11. *In a causal chain of* C_SON*: (i) no condition occurs more than once; and (ii) between two occurrences of an event e there can only occur events and no conditions.*

Proof. (i) A repetition of a condition would contradict the acyclicity of $Prec_{\text{C\_SON}}$ and (†).
(ii) Suppose that λ is a causal chain with a sub-sequence $e\lambda' c\lambda'' e$, where c is a condition. Then $c\lambda'' e\lambda' c$ is a causal chain, contradicting (i). □

Proof. (of Proposition 2)
Suppose that $C \stackrel{\text{df}}{=} Cut \cap C_i$ is not a cut. Then, since C is a set of concurrent conditions of ON_i as $Prec_{\text{ON}_i} \subseteq Prec_{\text{C\_SON}}$, there is a condition $c \in C_i \setminus C = C_i \setminus Cut$ which is concurrent with all the conditions in C. Since $c \notin Cut$, we have that

$$(\{c\} \times Cut) \cap Prec^+_{\text{C\_SON}} \neq \varnothing \text{ or } (Cut \times \{c\}) \cap Prec^+_{\text{C\_SON}} \neq \varnothing \ .$$

Without loss of generality, we may assume that the former holds. Then, by (‡), the set Λ of causal chains from c to Cut (i.e., to a condition in Cut) is non-empty.

Let $\lambda \in \Lambda$. We first observe that not all events (and so also conditions) in λ belong to the occurrence net ON_i as c is concurrent with all the conditions in C. Hence we can represent λ as $c\phi f g \theta d$ where the sub-chain ϕ does not contain events outside E_i, $f \in E_i$, $g \notin E_i$ and $d \in Cut$. We will denote by h_λ the number of events in ϕ. Moreover, since ϕ is of the form $e_1 c_1 e_2 c_2 \ldots e_{h_\lambda} c_{h_\lambda}$, where the e_i's are events and c_i's conditions in ON_i, it follows from Proposition 11(i) that $h_\lambda < |C_i|$ (note that $c\phi$ is a causal chain). Hence there is a causal chain $\widehat{\lambda} \in \Lambda$ with the highest $h_{\widehat{\lambda}}$. Let $\widehat{\lambda} = ce_1 c_1 e_2 c_2 \ldots e_{h_{\widehat{\lambda}}} c_{h_{\widehat{\lambda}}} f g \theta d$, and consider any condition $c' \in post(f) \subseteq C_i$, which means that $\widetilde{\lambda} = ce_1 c_1 e_2 c_2 \ldots e_{h_{\widehat{\lambda}}} c_{h_{\widehat{\lambda}}} f$ is a causal chain.

Clearly, $c' \notin Cut$ since c is concurrent with all the conditions in C, and $c' \neq c$, by Proposition 11(i) and $\widetilde{\lambda}$ being a causal chain. We then observe that c' is not in the $Prec^+_{\text{C\_SON}}$ relation with any condition in Cut. Indeed, if $(c'', c') \in Prec^+_{\text{C\_SON}}$ and $c'' \in Cut$ then, by the fact that $|pre(c')| = 1$ and using the event f, we have $(c'', d) \in Prec^+_{\text{C\_SON}}$ a contradiction with the definition of a cut. Moreover, if $(c', c'') \in Prec^+_{\text{C\_SON}}$ and $c'' \in Cut$ then, by $|post(c')| = 1$, we can find a new causal chain $\lambda' = ce_1 c_1 e_2 c_2 \ldots e_{h_{\widehat{\lambda}}} c_{h_{\widehat{\lambda}}} f c' f' \theta' c''$ in Λ, where $f' \in E_i$, for which $h_{\lambda'}$ is greater than $h_{\widehat{\lambda}}$, contradicting the choice of $\widehat{\lambda}$.

Hence Cut is not a minimal set of conditions such that no two conditions in Cut are related by $Prec^+_{\text{C\_SON}}$, a contradiction. □

Our investigation of the dynamic behaviour of C_SON starts by showing that its events cannot be completely blocked right at the beginning.

Proposition 12. *If $\mathbf{E} \neq \varnothing$ then there is at least one step execution involving a non-empty set of events.*

Proof. For the causal chain λ as in (*) and event e^i_j occurring in it, let $ind_\lambda(e^i_j) \stackrel{\text{df}}{=} i$. By Proposition 11(ii), this notion is well-defined as $e^i_j = e^{i'}_{j'}$ implies $i = i'$. Now, for any event e, let Λ_e be the set of all causal chains beginning with a condition in $Init_{\text{C\_SON}}$ and containing e. Clearly, $\Lambda_e \neq \varnothing$ as we can always find at least one suitable causal chain with all the elements in the component occurrence net to which e belongs. Then, for every e, let $ind(e) \stackrel{\text{df}}{=} \max\{ind_\lambda(e) \mid \lambda \in \Lambda_e\}$. That $ind(e)$ is a well-defined integer follows from Proposition 11(i) which implies that $ind_\lambda(e) \leq |\mathbf{C}|$, for every $\lambda \in \Lambda_e$.

Let G be the (non-empty) set of all events e for which $ind(e)$ has the minimal value among all the events of C_SON. One can easily see that: (i) $e \in G$ implies that there is no event f such that $(f, e) \in \mathbf{F} \circ \mathbf{F}$ and so $pre(e) \subseteq Init_{\text{C\_SON}}$; and (ii) if $(f, e) \in \kappa \cup \sigma$ then $f \in G$. Hence $\chi = D\, G\, D'$, where $D = Init_{\text{C\_SON}}$ and $D' \stackrel{\text{df}}{=} (Init_{\text{C\_SON}} \setminus pre(G)) \cup post(G)$, is a step execution of C_SON involving a non-empty set of events. □

Proof. (of Theorem 3)
Follows directly from the definitions, $Init_{\text{ON}_i} = Init_{\text{C\_SON}} \cap C_i$ and the disjointness of the component occurrence nets. □

Proposition 13. *Given the step execution as in Definition 6 and the causal chain as in (*), if $e^i_j \in G_l$ ($l \leq n$) then each event $e^i_{j'}$ with $j' < j$ belongs to some step $G_{l'}$ with $l' \leq l$, each event $e^{i'}_{j'}$ with $i' < i$ belongs to some step $G_{l'}$ with $l' < l$.*

Proof. The result follows from the following two observations. First, if $j = 1$, $i' = i - 1$ and $j' = k_{i-1}$ then both e^i_j and $e^{i'}_{j'}$ belong to the same component occurrence net and $(e^i_j, e^{i'}_{j'}) \in F \circ F$, and so by Theorem 3 and Proposition 8(1), $e^{i'}_{j'}$ belongs to some $G_{l'}$ with $l' < l$. Second, if $j' = j - 1$ then, by Definition 6, $e^i_{j'}$ belongs to some $G_{l'}$ with $l' \leq l$. □

As in the case of an occurrence net, a cut Cut of C_SON divides it into two parts. We define

$$precson_{\text{C\_SON}}(Cut) \stackrel{\text{df}}{=} (\text{ON}'_1, \ldots, \text{ON}'_k, \kappa', \sigma') \text{ and } postcson_{\text{C\_SON}}(Cut) \stackrel{\text{df}}{=} (\text{ON}''_1, \ldots, \text{ON}''_k, \kappa'', \sigma'')$$

where, for $i \leq k$, the component occurrence nets are defined by:

$$\text{ON}'_i \stackrel{\text{df}}{=} preon_{\text{ON}_i}(Cut \cap C_i) \text{ and } \text{ON}''_i \stackrel{\text{df}}{=} poston_{\text{ON}_i}(Cut \cap C_i)\ ,$$

and the relations capturing communication are given by restricting κ and σ to the events occurring in $\text{ON}'_1, \ldots, \text{ON}'_k$ and $\text{ON}''_1, \ldots, \text{ON}''_k$, i.e., $\kappa' \stackrel{\text{df}}{=} \kappa|_{\mathbf{E}' \times \mathbf{E}'}$, $\kappa'' \stackrel{\text{df}}{=} \kappa|_{\mathbf{E}'' \times \mathbf{E}''}$, $\sigma' \stackrel{\text{df}}{=} \sigma|_{\mathbf{E}' \times \mathbf{E}'}$ and $\sigma'' \stackrel{\text{df}}{=} \sigma|_{\mathbf{E}'' \times \mathbf{E}''}$ where, according to our notational conventions, $\mathbf{E}'$ and $\mathbf{E}''$ are events in $\text{ON}'_1, \ldots, \text{ON}'_k$ and $\text{ON}''_1, \ldots, \text{ON}''_k$, respectively. Similar notations are used for sets of conditions and flow relations. Note that the occurrence nets in ON'_i and ON''_i are well-defined due to Proposition 2 which guarantees that $Cut \cap C_i$ is a cut of ON_i.

Proposition 14. *Let* C_SON′ *and* C_SON″ *be the tuples* $precson_{\text{C\_SON}}(Cut)$ *and* $postcson_{\text{C\_SON}}(Cut)$*, respectively, defined above.*

1. C_SON′ *and* C_SON″ *are communication structured occurrence nets with disjoint sets of events. Moreover,* $\mathbf{C} = \mathbf{C}' \cup \mathbf{C}''$ *and* $\mathbf{C}' \cap \mathbf{C}'' = Cut$.
2. $Init_{\text{C\_SON}'} = Init_{\text{C\_SON}}$, $Fin_{\text{C\_SON}'} = Cut = Init_{\text{C\_SON}''}$ *and* $Fin_{\text{C\_SON}''} = Fin_{\text{C\_SON}}$.
3. $(\mathbf{E}'' \times \mathbf{E}') \cap \kappa = \varnothing$ *and so* $\kappa' \cup \kappa'' = \kappa \setminus (\mathbf{E}' \times \mathbf{E}'')$.
4. $(\mathbf{E}'' \times \mathbf{E}') \cap \sigma = \varnothing$ *and so* $\sigma' \cup \sigma'' = \sigma \setminus (\mathbf{E}' \times \mathbf{E}'')$.
5. *Given step executions,* χ' *and* χ''*, of respectively,* C_SON′ *and* C_SON″*,* χ' *is a step execution of* C_SON *and, moreover, if* χ' *leads to* Cut *then* $\chi = \chi' \varnothing \widehat{\chi}''$ *is a step execution of* C_SON.

Proof. (1,2) Follow from the definitions and Propositions 7 and 2.
(3,4) Follow from the fact that otherwise we would have had a causal chain in C_SON starting and ending at condition in Cut, contradicting Proposition 10.
(5) Follows from Proposition 7 and parts (1–4). □

Proof. (of Theorem 4)
To show that D_i is a cut of C_SON, we first observe that, by Theorems 2 and 3, for every $l \leq k$, $D_i \cap C_l$ is a cut of ON_l. Thus D_i cannot be extended by any new condition which is not in the $Prec^+_{\text{C\_SON}}$ relation with the conditions in D_i. Hence, by Proposition 10, to show that D_i is a cut of C_SON it suffices to demonstrate that there is no causal chain $\lambda = ce\theta fd$ such that $c, d \in D_i$. In order to obtain a contradiction, suppose that such a λ does exist. By $d \in D_i$ and $d \in post(f)$ and Propositions 8(2) and Theorem 3, we have $f \in G_h$ for some $h \leq i$. Hence, by Proposition 13, we have that $e \in G_m$ for some $m \leq h$ (note that we allow $e = f$). Thus, by $c \in pre(e)$ and $m \leq i$ and Proposition 8(3) and Theorem 3, $c \notin D_i$, a contradiction. Hence D_i is a cut of C_SON.

The next parts, i.e., that no event occurs more than once and $D_n = Fin_{\text{C\_SON}}$ *iff* each event of $\mathbf{E}$ occurs in the execution, follow directly from Theorems 2 and 3.

In the next stage, proceeding by induction on the number of events in a communication structured occurrence net, we show that one can find a step execution involving all the events. In the base case, there are no events and so there is nothing to show as $\chi = Init_{\text{C\_SON}}$ is a valid step execution. In the induction step, from Proposition 12 it follows that we can find a step execution χ with a non-empty set of events. Let Cut be the resulting cut (note that we have shown that Cut is indeed a cut in the first part of this proof). Now, we can take $precson_{\text{C\_SON}}(Cut)$ and $postcson_{\text{C\_SON}}(Cut)$, and find by induction a step

execution χ' involving all the events of $poston_{\text{C\_SON}}(Cut)$. We then observe that χ is a step execution of $precson_{\text{C\_SON}}(Cut)$ leading to its final cut and so, by what we have already shown, χ involves all the events of $precson_{\text{C\_SON}}(Cut)$. By Proposition 14(5), we then get that $\chi\widehat{\chi}$ is a step execution of C_SON involving all the events of **E**.

Finally, let Cut be a cut of C_SON. We can take $preon_{\text{C\_SON}}(Cut)$. By what we have just shown, there is a step execution χ of $preon_{\text{C\_SON}}(Cut)$ leading to Cut (follows from Theorems 2 and 3). Thus, by Proposition 14(5), χ is also a step execution of C_SON leading to Cut. □

C Behavioural abstraction (Section 4)

Throughout this section, we assume that the B_SON is as in Definition 7.

We first observe that some of the κ-relationships in C_SON may be redundant, and some potential κ-relationships are impossible. Take, for instance, the B_SON in Figure 14. Adding a new arc (f_3, f_1) to κ would not be correct (i.e., the result would not be a behavioural structured occurrence net). Indeed, by $(c_2, c_3) \in before(e_1)$ and $(c_3, c_4) \in Prec^+_{\text{ON}_3}$ we have that $(c_2, c_4) \in Prec^+_{\text{B\_SON}}$. Moreover, $(c_4, c_2) \in pre(f_3) \times post(f_1)$, and so adding (f_3, f_1) to κ would produce a causal cycle. Redundancy can be illustrated by observing that adding an arc (g_1, g_3) to κ would result in a behavioural structured occurrence net B_SON$'$ with the same behaviour as B_SON. Indeed, all the new arc would contribute are the causal relationships $Pre(g_1) \times Post(g_3) = \{c_1, d_1\} \times \{c_5, d_5\}$. But these are already present in $Prec^+_{\text{B\_SON}}$, on account of $Prec^+_{\text{C\_SON}}$ and $before(e_3)$, and so $Prec^+_{\text{B\_SON}} = Prec^+_{\text{B\_SON}'}$. Hence both B_SON and B_SON$'$ would have the same cuts. Moreover, the only possible impact on step executions would be that $g_3 \in G_i$ implies $g_1 \in \bigcup_{j \leq i} G_j$. Yet this holds anyway for B_SON as $(d_2, d_4) \in Prec^+_{\text{B\_SON}}$, $g_1 \in pre(d_2)$ and $g_3 \in post(d_4)$.

More generally, let us consider Definition 7(1) and assume that $\pi_1 \dots \pi_m$ is a concatenation of $h \geq 2$ phase decompositions of occurrence nets $\text{ON}_1, \dots, \text{ON}_h$ (in that order). Then there are integers $1 = m_1 < m_2 < \dots < m_h < m_{h+1} = m + 1$ such that $\pi_{m_j} \pi_{m_j+1} \dots \pi_{m_{j+1}-1}$ is a phase decomposition of $\text{ON}_j = (C_j, E_j, F_j)$. For every $j < h$, we will denote $\text{ON}_j \rightsquigarrow_{e_{m_{j+1}-1}} \text{ON}_{j+1}$ (or simply $\text{ON}_j \rightsquigarrow \text{ON}_{j+1}$) and call $e_{m_{j+1}-1}$ an *off-line modification* event. Events appearing in $\xi = c_1 e_1 c_2 e_2 \dots e_{m-1} c_m$ different from $e_{m_2-1}, \dots, e_{m_h-1}$ will be called *on-line modification* events.

Proposition 15. *Given the above assumptions, the following hold.*

1. *For every $j < h$, $C_j \times C_{j+1} \subseteq F^*_j|_{C_j \times C_j} \circ (Fin_{\text{ON}_j} \times Init_{\text{ON}_{j+1}}) \circ F^*_{j+1}|_{C_{j+1} \times C_{j+1}} \subseteq Prec^+_{\text{B\_SON}}$.*
2. *The occurrence nets $\text{ON}_1, \dots, \text{ON}_h$ are all distinct.*
3. *If $j < l$ then $(E_l \times E_j) \cap \kappa = \varnothing$.*
4. *If B_SON$'$ is B_SON with κ replaced by $\widetilde{\kappa} \stackrel{\text{df}}{=} \kappa \setminus \bigcup_{j<l} E_j \times E_l$, then B_SON$'$ is a behavioural structured occurrence net with exactly the same cuts and step executions as B_SON.*

Proof. (1) First, we observe that for every condition c of any occurrence net ON there are conditions $b \in Init_{\text{ON}}$ and $d \in Fin_{\text{ON}}$ such that $(b, c) \in F^*$ and $(c, d) \in F^*$. Second, we observe that $Max_{\pi_{m_{j+1}-1}} = Fin_{\text{ON}_j}$ and $Min_{\pi_{m_{j+1}}} = Init_{\text{ON}_{j+1}}$ and $Max_{\pi_{m_{j+1}-1}} \times Min_{\pi_{m_{j+1}}} \subseteq before(e_{m_{j+1}-1})$.

(2) Follows from part (1) and the acyclicity of $Prec^+_{\text{B\_SON}}$.

(3) Suppose that $(e, f) \in (E_l \times E_j) \cap \kappa$. Take any $c \in pre(e)$ and $d \in post(f)$. Clearly, $(c, d) \in Prec^+_{\text{B\_SON}}$. On the other hand, by part (1), $(d, c) \in Prec^+_{\text{B\_SON}}$, contradicting the acyclicity of $Prec^+_{\text{B\_SON}}$.

(4) B_SON$'$ satisfies Definition 7(2) as $Prec_{\text{B\_SON}'} \subseteq Prec_{\text{B\_SON}}$ and $Prec_{\text{B\_SON}}$ is acyclic. Hence $Prec_{\text{B\_SON}'}$ is a behavioural structured occurrence net and, clearly, $Prec^+_{\text{B\_SON}'} \subseteq Prec^+_{\text{B\_SON}}$. What is more, $Prec^+_{\text{B\_SON}} \subseteq Prec^+_{\text{B\_SON}'}$. Indeed, suppose that $(e, f) \in \kappa \cap (\bigcup_{j<l} E_j \times E_l)$. Such an arc is used in B_SON to construct causal chains defined similarly as in the case of communication structured occurrence nets. Although in B_SON$'$ the arc (e, f) is not present, we can simulate it by other causal relationships, as follows. Let

$c \in post(e)$ and $d \in pre(f)$. By part (1), and the way B_SON was modified, we have $(c,d) \in C_j \times C_{j+1} \subseteq F^*_j|_{C_j \times C_j} \circ (Fin_{\mathrm{ON}_j} \times Init_{\mathrm{ON}_{j+1}}) \circ F^*_{j+1}|_{C_{j+1} \times C_{j+1}}$. Hence any causal chain involving $(e,f) \in \kappa$ can be simulated by a causal chain without $(e,f) \in \kappa$, and so $Prec^+_{\text{B\_SON}} \subseteq Prec^+_{\text{B\_SON}'}$.

We have shown that $Prec^+_{\text{B\_SON}} = Prec^+_{\text{B\_SON}'}$. Hence B_SON$'$ and B_SON have exactly the same cuts. Moreover, we observe that for the pair (e,f) considered above and the step execution of B_SON$'$ we have that $f \in G_i$ implies necessarily $e \in G_l$ for $l < i$ which in turn is also guaranteed to hold in B_SON. Hence B_SON$'$ and B_SON have exactly the same step executions. □

From Proposition 15(3,4), it follows that when proving Proposition 3 and Theorems 5 and 6 for B_SON, we can assume that:

$$(E_l \times E_j) \cap \kappa = \varnothing, \text{ for all } j, l \leq h. \qquad (\S)$$

The idea behind proving Proposition 3 and Theorems 5 and 6 is to construct a communication structure occurrence net $\widehat{\text{C\_SON}}$ behaviourally equivalent to B_SON, and then to use the results already established for the $\widehat{\text{C\_SON}}$ to derive similar results for B_SON. The construction first converts all on-line modification events to equivalent off-line modification events. Next, for a behavioural structured occurrence net without on-line modifications, one constructs a communication structured occurrence net with the same behaviour. This is done by connecting each pair of occurrence nets satisfying ON $\rightsquigarrow_e$ ON$'$ using a fresh event $\widehat{e}$. The final result is illustrated in Figure 28 for the behavioural structured occurrence net of Figure 14.

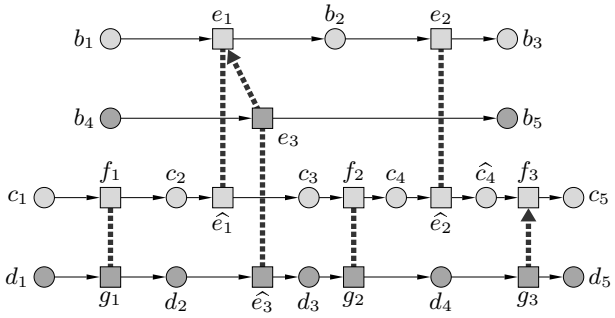


Fig. 28. An illustration to the construction of $\widehat{\text{C\_SON}}$ used to prove properties of the B_SON of Figure 14.

The way to convert an on-line modification event e_i as in Definition 7(1) into an off-line modification event is as follows. Consider $D = Max_{\pi_i} = Max_{\pi_{i+1}}$ which is a cut a component occurrence net ON. We can take ON$' = preon_{\mathrm{ON}}(D)$ and ON$''$ which is obtained from $poston_{\mathrm{ON}}(D)$ by replacing each $c \in D$ by a fresh condition $\widehat{c}$ (thus $\widehat{D}$ is a set of fresh conditions). We then replace ON with ON$'$ and ON$''$. This, in effect, splits ON along D into two disjoint occurrence nets (the renaming of the initial conditions of $poston_{\mathrm{ON}}(D)$ is needed to ensure the disjointness of the component occurrence nets). Moreover, we replace in β each pair (c, c_j) satisfying $c \in D$ and $j > i$ by $(\widehat{c}, c_j)$. The result of both modifications is denoted by $\overline{\text{B\_SON}}$.

To demonstrate that $\overline{\text{B\_SON}}$ is a behavioural structured occurrence net, all we need to do is check that the relation $Prec_{\overline{\text{B\_SON}}}$ is acyclic. It follows directly from the definitions that:

$$\begin{aligned} Prec_{\overline{\text{B\_SON}}} = \phi(Prec_{\text{C\_SON}'} \cup Prec_{\text{C\_SON}}) \cup \psi(Prec_{\text{C\_SON}'} \cup Prec_{\text{C\_SON}}) \ \cup \\ (D \times \widehat{D}) \cup (Pre(e_i) \times \widehat{D}) \cup (D \times Post(e_i)) \ \cup \\ \textstyle\bigcup_{e \in \mathbf{E}' \setminus \{e_i\}} \phi(before(e)) \cup \psi(before(e)) \ , \end{aligned} \qquad (\sharp)$$

where: $\phi(R)$ removes from R all pairs of the form (c,x) with $c \in D$; and $\psi(R)$ removes from R all pairs of the form (x,c) with $c \in D$, and then replaces all pairs of the form (c,x) with $c \in D$ by $(\widehat{c},x)$. This

and the fact that $Prec_{\text{B\_SON}}$ is acyclic implies that $Prec_{\overline{\text{B\_SON}}}$ is also acyclic. The reason is that one could transform any cycle of $Prec_{\overline{\text{B\_SON}}}$ into a cycle of $Prec_{\text{B\_SON}}$. Indeed, a cycle of $Prec_{\overline{\text{B\_SON}}}$ would have to use arcs $(b,d) \in (D \times \widehat{D}) \cup (Pre(e_i) \times \widehat{D}) \cup (D \times Post(e_i))$, each of which can be simulated in $Prec_{\text{B\_SON}}$, in the following way. For such b and d there would have to be conditions b' and d' (not belonging to $\widehat{D}$) directly preceding (after ignoring the events) and following b and d in the cycle, respectively. We would then have that $(b',d') \in before(e_i)$ and use this relationship to construct a causal cycle in $Prec_{\text{B\_SON}}$.
It follows from (♯) that if Cut is a cut of $\overline{\text{B\_SON}}$ then $Cut \cap D = \varnothing$ or $Cut \cap \widehat{D} = \varnothing$. Moreover, if we replace each $\widehat{c} \in \widehat{D} \cap Cut$ by c then the result is a cut of B_SON. And, conversely, any cut Cut of B_SON can be transformed into a cut of $\overline{\text{B\_SON}}$ by replacing each $c \in D$ with $\widehat{c}$, provided that $c_j \in Cut$ for some $j > i$.
We then observe that any step execution of B_SON can be transformed into a valid step execution of $\overline{\text{B\_SON}}$ by changing each cut of B_SON in the way just described. And, conversely, each step execution of $\overline{\text{B\_SON}}$ can be transformed into a valid step execution of B_SON in the way just described. Moreover, $Init_{\text{B\_SON}} = Init_{\overline{\text{B\_SON}}}$ and $Fin_{\text{B\_SON}} = Fin_{\overline{\text{B\_SON}}}$. Thus we can claim Proposition 3 and Theorems 5 and 6 for B_SON if they hold for $\overline{\text{B\_SON}}$. Similar reasoning can be applied to the remaining on-line modification events, and so at the end we face the task of proving Proposition 3 and Theorems 5 and 6 for B_SON without on-line any modification events.

We construct $\widehat{\text{C\_SON}}$ given a B_SON without on-line modification events and satisfying (§). The key is to build, given the occurrence nets $\text{ON}_1, \ldots, \text{ON}_m$ as at the beginning of this section (note that $h = m$ since B_SON does not have any on-line modification events), a single occurrence net:

$$\widehat{\text{ON}} \stackrel{\text{df}}{=} \Big(\bigcup_{i=1}^{m} C_i, \bigcup_{i=1}^{m} E_i \cup \{\widehat{e_1}, \ldots, \widehat{e_{m-1}}\}, \bigcup_{i=1}^{m} F_i \cup \bigcup_{i=1}^{m-1} ((Fin_{\text{ON}_i} \times \{\widehat{e_i}\}) \cup (\{\widehat{e_i}\} \times Init_{\text{ON}_{i+1}})) \Big) ,$$

where $\widehat{e_1}, \ldots, \widehat{e_{m-1}}$ are fresh events. Note that $\widehat{\text{ON}_i}$ is an occurrence net due to Proposition 15(1), and the fact that $\rightsquigarrow$ is a total order for the occurrence nets $\text{ON}_1, \ldots, \text{ON}_m$. Then we define

$$\widehat{\text{C\_SON}} \stackrel{\text{df}}{=} \big(\text{ON}'_1, \ldots, \text{ON}'_l, \widehat{\text{ON}_1}, \ldots, \widehat{\text{ON}_l}, \kappa \cup \kappa' \cup \bigcup_{e \in \mathbf{E}} \{(e,\widehat{e}), (\widehat{e},e)\} \big) .$$

To start with, we observe that, due to (§), $\widehat{\kappa}$ relates events coming from different occurrence nets. We then observe that $Prec_{\text{B\_SON}} = Prec_{\widehat{\text{C\_SON}}}$ which follows from $Pre(\widehat{e_i}) = Pre(e_i) \cup Max_{\pi_i}$ and $Post(\widehat{e_i}) = Post(e_i) \cup Min_{\pi_{i+1}}$. From (§§) it follows that B_SON and $\widehat{\text{C\_SON}}$ have the same cuts. Moreover, $Init_{\text{B\_SON}} = Init_{\widehat{\text{C\_SON}}}$ and $Fin_{\text{B\_SON}} = Fin_{\widehat{\text{C\_SON}}}$.
We then observe that any step execution of B_SON can be transformed into a valid step execution of $\widehat{\text{C\_SON}}$ by adding to each step G_i all the events $\widehat{e}$ such that $e \in G_i \cap \mathbf{E}'$. And, conversely, each step execution of $\widehat{\text{C\_SON}}$ can be transformed into a valid step execution of B_SON by deleting all the events $\widehat{e}$. Then we can claim Proposition 3 and Theorems 5 and 6 for B_SON on the basis of Proposition 2 and Theorems 3 and 4 which hold for $\widehat{\text{C\_SON}}$.

D Spatial and temporal abstractions (Section 5)

Below we assume that the S_SON is as in Definition 9.

Proof. (of Proposition 4)
(1) Suppose that $C = Cut \cap \mathbf{C}$ is not a cut. Since $Prec_{\text{C\_SON}} \subseteq Prec_{\text{C\_SON}''}$ and C_SON$''$ is a communication structured occurrence net, it follows that $(C \times C) \cap Prec^{+}_{\text{C\_SON}} = \varnothing$. Hence, without loss of generality, the set Z comprising all $c \in \mathbf{C} \setminus C$ such that $(C \times \{c\} \cup C \times \{c\}) \cap Prec^{+}_{\text{C\_SON}} = \varnothing$ and there is a causal chain from Cut to c in C_SON$''$, is non-empty. Since each occurrence net is acyclic, there is $c \in Z$ such that there is no $c' \in Z$ satisfying $(c',c) \in \mathbf{F} \circ \mathbf{F}$. Let $\lambda = d\phi ec$ be a causal chain from Cut to c, and take

any $b \in pre(e)$. Clearly, $b \notin C$ as $(b,c) \in \mathbf{F} \circ \mathbf{F}$. We then consider two cases.
Case 1: $(C \times \{b\} \cup C \times \{b\}) \cap Prec^+_{\text{C\_SON}} = \varnothing$. Then, since $b \notin Z$ and by the choice of c, there must be a causal chain λ' from b to Cut in C_SON$''$. By $|post(b)| = 1$, λ' is of the form $be\psi d'$, where $d' \in Cut$. Hence $d\phi e\psi d'$ is a causal chain from Cut to Cut, a contradiction.
Case 2: $(C \times \{b\} \cup C \times \{b\}) \cap Prec^+_{\text{C\_SON}} \neq \varnothing$. Then $C \times \{b\} \neq \varnothing$ is impossible as $C \times \{c\} = \varnothing$. Hence there is a causal chain from b to C in C_SON. We then proceed similarly as in Case 1.

(2) We proceed similarly as for part (1).

(3) Suppose that $c \in Cut$ but $\varsigma(c) \notin Cut$. Then, without loss of generality, there is a causal chain $\lambda = \varsigma(c)e\phi$ from $\varsigma(c)$ to Cut. Hence $c \in \varsigma^{-1}(pre(e))$, and by Definition 9(2), we have that $c \in pre(\varsigma^{-1}(e))$. Consequently, there is event f such that $c \in pre(f)$ and $\varsigma(f) = e$. Thus $(f,e) \in \kappa''$, and so $cfe\phi$ is a causal chain from c to Cut, a contradiction. The proof in the other direction is similar. □

Proof. (of Theorem 7)
Follows from an observation that, by Definition 10, the step executions of S_SON are exactly the step executions of C_SON$''$. The result then follows from Proposition 4 and Theorem 3. □

Proof. (of Theorem 8)
Follows from an observation that, by Definition 10, the step executions of S_SON are exactly the step executions of C_SON$''$. The result then follows from Proposition 4 and Theorem 4. □

We now move on to dealing with the temporal abstraction structured occurrence nets, assuming that T_SON is as in Definition 11. First, we prove two auxiliary results directly relating to Definition 11. Below, for each condition $c \in \mathbf{C}$ we denote by $\tau^{-1}(c)$ the unique condition $d \in \mathbf{C}'$ satisfying $\tau(d) = c$ (see Definition 11(3)).

Proposition 16. *If* $(c,d) \in Prec^+_{\text{C\_SON}'}$ *and* $\tau(c), \tau(d) \in \mathbf{C}$*, then* $(\tau(c), \tau(d)) \in Prec^+_{\text{C\_SON}}$.

Proof. Follows from Definition 11(4,5). □

Proposition 17. *If* $i \leq k$ *and* $(c,d) \in Prec^+_{\text{ON}_i}$*, then* $(\tau^{-1}(c), \tau^{-1}(d)) \in Prec^+_{\text{ON}'_i}$.

Proof. It suffices to show the result assuming that $(c,e) \in F_i$ and $(e,d) \in F_i$, for some event $e \in E_i$. Let B be the block of ON$'_i$ such that $B = \tau^{-1}(e)$. By Definition 11(4), there are events $e', e'' \in B$ such that $(\tau^{-1}(c), e') \in F'_i$ and $(e'', \tau^{-1}(d)) \in F'_i$. Hence $\tau^{-1}(c) \in pre(B) \setminus B$ and $\tau^{-1}(d) \in post(B) \setminus B$, and so, by Definition 4, $(\tau^{-1}(c), \tau^{-1}(d)) \in Prec^+_{\text{ON}'_i}$. □

Proof. (of Proposition 5)
Let $i \leq k$. By Proposition 2, $C = Cut \cap C_i$ is a cut of ON$_i$. Suppose that $C' = \tau^{-1}(C)$ is not a cut of ON$'_i$. By Proposition 16, C' is a set of mutually concurrent conditions of ON$'_i$. Hence there must exist $c \in C'_i \setminus C'$ which is concurrent with all the conditions of C'. We consider two cases.
Case 1: $d = \tau(c) \in C_i$. Then, since $d \notin C$ and C is a cut of ON$_i$, we can assume, without loss of generality, that $(d,b) \in Prec^+_{\text{ON}_i}$ for some $b \in C$. Hence, by Proposition 17, and Definition 11(3), $(c, \tau^{-1}(b)) \in Prec^+_{\text{ON}'_i}$. This and $\tau^{-1}(b) \in C'$ produces a contradiction with c being concurrent with all the conditions of C'.
Case 2: $e = \tau(c) \in E_i$. Let B be the block of ON$'_i$ such that $B = \tau^{-1}(e)$. By Proposition 9, there are $b \in pre(B) \setminus B$ and $d \in post(B) \setminus B$ such that $(b,c) \in (F'_i)^+$ and $(c,d) \in (F'_i)^+$. Clearly, $b, d \notin C'$ and so neither $\widehat{b} = \tau(b) \in pre(e)$ nor $\widehat{d} = \tau(d) \in post(e)$ belongs to C. Hence, since C is a cut of ON$_i$, neither $\widehat{b}$ nor $\widehat{d}$ is concurrent with all the conditions of C.
Now consider $\widehat{d}$ and suppose that $(\widehat{d}, d') \in Prec^+_{\text{ON}_i}$ for some $d' \in C$. Then, by $(c,d) \in (F'_i)^+$ and Proposition 17, we have that $(c, \tau^{-1}(d')) \in Prec^+_{\text{ON}'_i}$, producing a contradiction as $\tau^{-1}(d') \in C'$. Hence it must be the case that $(d', \widehat{d}) \in Prec^+_{\text{ON}_i}$, for some $d' \in C$. Thus, by $|pre(\widehat{d})| = 1$, there is $\widetilde{d} \in pre(e)$ such

that $(d', \widetilde{d}) \in Prec^+_{\mathrm{ON}_i}$. By Proposition 17, we have $(\tau^{-1}(d'), \tau^{-1}(\widetilde{d})) \in Prec^+_{\mathrm{ON}'_i}$ as well as $\tau^{-1}(d') \in C'$ and $\tau^{-1}(\widetilde{d}) \in pre(B) \setminus B$.
Proceeding in a similar way, we can show that there are $b' \in C'$ and $b'' \in post(B) \setminus B$ such that $(b'', b') \in Prec^+_{\mathrm{ON}'_i}$. Since, by the definition of a block, we have that $(\tau^{-1}(\widetilde{d}), b'') \in (F'_i)^+$, it follows that $(\tau^{-1}(d'), b') \in Prec^+_{\mathrm{ON}'_i}$, producing a contradiction with C' being a set of concurrent conditions.

We have shown that $\tau^{-1}(Cut \cap C_i)$ is a cut of ON_i, for every $i \leq k$. Hence, adding any new conditions to $\tau^{-1}(Cut)$ would necessarily create a causal chain between conditions of $\tau^{-1}(Cut)$. Moreover, by Proposition 16, there are no causal chains between the conditions of $\tau^{-1}(Cut)$, and we can conclude that $\tau^{-1}(Cut)$ is a cut of C_SON′. □

Proof. (of Theorem 9)
Let $i \leq n$. From Theorem 4 it follows that D_{i-1} and D_i are cuts of C_SON. Hence, by Proposition 5, $\tau^{-1}(D_{i-1})$ and $\tau^{-1}(D_i)$ are cuts of C_SON′. Also, one can easily see that $(D_i \times D_{i-1}) \cap \mathbf{F}^+ = \varnothing$. By Proposition 16, we have that $(\tau^{-1}(D_i) \times \tau^{-1}(D_{i-1})) \cap \mathbf{F}'^+ = \varnothing$. Moreover, one can show that

$$\tau^{-1}(G_i) \cap \mathbf{E}' = \{e \in \mathbf{E}' \mid \exists c \in \tau^{-1}(D_{i-1}), d \in \tau^{-1}(D_i) : (c, e) \in \mathbf{F}'^+ \wedge (e, d) \in \mathbf{F}'^+\} .$$

Let $\text{C\_SON}'' = precson_{precson_{\text{C\_SON}'}(\tau^{-1}(D_{i-1}))}(\tau^{-1}(D_i))$. It is easily seen that C_SON″ is a communication structured occurrence net such that $Init_{\text{C\_SON}''} = \tau^{-1}(D_{i-1})$, $Fin_{\text{C\_SON}''} = \tau^{-1}(D_i)$ and $\mathbf{E}'' = \tau^{-1}(G_i) \cap \mathbf{E}'$. By Theorem 4 there is a step execution from $Init_{\text{C\_SON}''}$ to $Fin_{\text{C\_SON}''}$ involving all the events of $\mathbf{E}''$. The same step execution can be reproduced in C_SON′, and so, by repeating similar argument for all $i \leq n$, the required step execution can be constructed. □

E Information retention and judgement (Section 6)

Throughout this section, we assume that the R_SON is as in Definition 12.

Let $\text{C\_SON} = (\mathrm{ON}_1, \ldots, \mathrm{ON}_k, \kappa \cup \widehat{\kappa})$, where $\mathrm{ON}_1, \ldots, \mathrm{ON}_k$ are all the component occurrence nets of the C_SON_i's. Intuitively, C_SON is the union of the C_SON_i's linked together by $\widehat{\kappa}$ extended with the auxiliary relation κ. Since $Prec_{\text{C\_SON}} = Prec_{\text{R\_SON}}$, it follows from Definition 12(1) that C_SON is a communication structured occurrence net. Moreover, by $Init_{\text{R\_SON}} = Init_{\text{C\_SON}}$ and Definition 13, the step executions of R_SON are exactly the step execution of C_SON.

Proof. (of Proposition 6)
Let $C = Cut \cap \mathbf{C}_i$. Since $Prec_{\text{C\_SON}} = Prec_{\text{R\_SON}}$, we have that Cut is a cut of the communication structured occurrence net C_SON. Hence, by Proposition 2, $Cut \cap C_j$ is a cut of ON_j, for every $j \leq k$. Moreover, since $Prec_{\text{C\_SON}_i} \subseteq Prec_{\text{R\_SON}}$ there are no causal chains in C_SON_i between the conditions of C. Hence C is a cut of C_SON_i. □

Proof. (of Theorem 10)
Follows from the fact that Theorem 3 holds for C_SON, and the following straightforward observations made on the basis of what we already said about C_SON and its relationship with R_SON: $Init_{\text{R\_SON}} \cap \mathbf{C}_i = Init_{\text{C\_SON}_i}$; $\kappa_i \subseteq \kappa \cup \widehat{\kappa}$; $pre(G_j) \subseteq D_{j-1}$ implies $pre(G_j \cap \mathbf{E}_i) \subseteq D_{j-1} \cap \mathbf{C}_i$; and $D_j = (D_{j-1} \setminus pre(G_j)) \cup post(G_j)$ implies $D_j \cap \mathbf{C}_i = (D_{j-1} \cap \mathbf{C}_i \setminus pre(G_j \cap \mathbf{E}_i)) \cup post(G_j \cap \mathbf{E}_i)$. □

Proof. (of Theorem 11)
Follows from the fact that the step executions of R_SON are the same as the step executions of C_SON, $Init_{\text{R\_SON}} = Init_{\text{C\_SON}}$, $Fin_{\text{R\_SON}} = Fin_{\text{C\_SON}}$, and the fact that Theorem 4 holds for C_SON. □