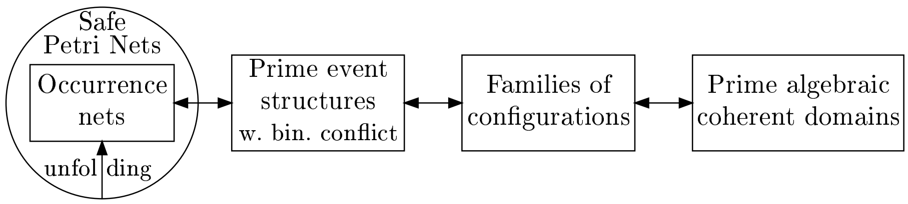
Petri Nets, Configuration Structures and Higher Dimensional Automata*

R.J. van Glabbeek**

Computer Science Department, Stanford University
Stanford, CA 94305-9045, USA.
`http://theory.stanford.edu/~rvg/`
`rvg@cs.stanford.edu`

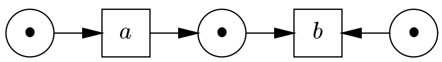
In this talk, translations between several models of concurrent systems are reviewed c.q. proposed. The models considered capture causality, branching time, and their interplay, and these features are preserved by the translations. To the extent that the models are intertranslatable, this yields support for the point of view that they are all different representations of the same phenomena. The translations can then be applied to reformulate any issue that arises in the context of one model into one expressed in another model, which might be more suitable for analysing that issue. To the extent that the models are not intertranslatable, my investigations are aimed at classifying them w.r.t. their expressiveness in modelling phenomena in concurrency. The results are summarised in the figure at the end of this paper.

Starting point is the work of NIELSEN, PLOTKIN & WINSKEL [17], in which safe Petri nets are translated, through the intermediate stages of occurrence nets, prime event structures with a binary conflict relation, and their families of configurations, into a class of Scott domains.



1 From nets to configurations

In VAN GLABBEEK & PLOTKIN [10] extensions of the translations above to unsafe Petri nets have been studied. For this purpose two different schools of thought in interpreting the causal behaviour of nets needed to be distinguished, which we called the *individual* and *collective token* philosophy. Their difference is illustrated by the following net. According to the individual token philosophy, A

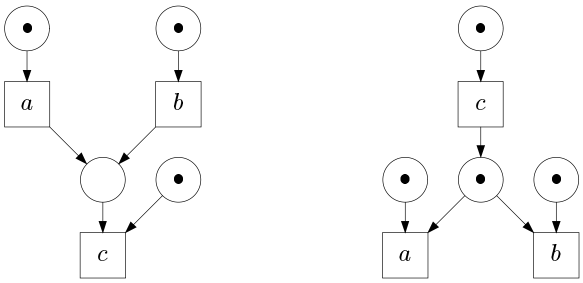
A: 

has an execution in which the action b causally depends on a, whereas according to the collective token philosophy, a and b are always causally independent.

* To appear in Proceedings CONCUR '99.
** This work was supported by ONR under grant number N00014-92-J-1974.

In Meseguer, Montanari & Sassone [15], the unfolding from [17], translating safe nets into the subclass of occurrence nets, is extended to arbitrary nets, while preserving the individual token interpretation. It follows that under this interpretation prime event structures are expressive enough to represent all processes expressible by Petri nets.

Under the collective token interpretation there turn out to be nets whose causal behaviour cannot be faithfully represented by a prime event structure. Representative examples are the two nets below, modelling what I often call



disjunctive causality and *resolvable conflict*, respectively.

In Winskel [25] a more general notion of event structure is proposed, extending the prime event structures with a binary conflict relation from [17], along with matching generalisations of the families of configurations and Scott domains. These event structures capture disjunctive causality, but not resolvable conflict.

The families of configurations of Winskel's event structures were introduced merely to facilitate the construction of the Scott domains associated to these event structures. In Van Glabbeek & Goltz [9] we found it convenient to use such families as a model of concurrency in its own right. In this context the families were called *configuration structures*. A configuration structure can be given by a set of *events*, modelling occurrences of actions the represented system may perform, possibly a *labelling function*, associating actions to events, and a collection of sets of events, the *configurations*, modelling the states of the represented system, and satisfying a number of closure conditions. A configuration represents the state in which the events it contains have occurred. The closure conditions ensure that each configuration structure can be regarded as the family of configurations of an event structure.

In [10] we proposed to drop the closure conditions, thereby obtaining a more general model of concurrency, capturing both disjunctive causality and resolvable conflict. The resulting configuration structures are, up to isomorphism, the *extensional Chu spaces* of Gupta & Pratt [13], but equipped with a slightly different computational interpretation. Through suitable translations we showed that these configuration structures are equally expressive as general Petri nets without self-loops. Such nets are called *pure*. To this end we defined a *1-occurrence net* to be a Petri net in which each transition can fire at most once, and we showed how any (pure) Petri net can be converted into a (pure) 1-occurrence net, using a construction we called *1-unfolding*. We argued that this conversion preserves essential features of the represented system like causality and branching time. It may convert a finite net into an infinite one, however. The translations between pure 1-occurrence nets and configuration structures take the transitions of the net to be the events of the configuration structure and vice versa; this way a configuration structure can be fully recovered from its Petri net representation. Our translations also extend the correspondence between flow nets and flow event structures proposed in BOUDOL [3].

ST-configuration structures are a further generalisation of configuration structures in which the configurations may contain certain events 'partially' (in case they are currently being executed). They are (a mild generalisation of) what are called *local event structures* in HOOGERS, KLEIJN & THIAGARAJAN [14]. In forthcoming work, Gordon Plotkin and I extend the translations between pure nets and configuration structures to translations between arbitrary Petri nets and ST-configuration structures, thus showing that also these models are equally expressive. The same was done, using a different construction, for general Petri nets without autoconcurrency in [14]. We also propose a matching generalisation of the model of event structures.

2 Scott domains versus process graphs

In [17] a "curious mismatch" is observed between the domains that result from translating nets or event structures, and the ones originally studied by SCOTT [22]. Although mathematically of the same nature, a domain that arises through the translations of [17] represents a single concurrent system, namely the same one represented by the Petri net or event structure it originated from. In domain theory, on the other hand, processes show up at best as the *elements* of a domain. Thus the use of domains to represent concurrent systems is novel in [17].

In most models of concurrency, attention is restricted to *discrete* processes, i.e. processes that can perform only finitely many actions in a finite time. Petri nets are commonly interpreted to represent discrete processes—this comes with the common definitions of the *firing rule*. On prime event structures the axiom of *finite causes* restricts attention to the structures representing discrete systems, and in Winskel's general event structures discreteness is obtained by the way the notion of a configuration of an event structure is defined. A Scott domain is a partially ordered set, satisfying certain conditions. The *finite* elements in such a domain are the ones that dominate only finitely many other elements. A discrete Scott domain (resulting from translating a discrete event structure) has the property that its infinitary part is redundant, in the sense that it can be recovered in full from its finitary part (the partial suborder of its finite elements). The finitary part of a domain can, without loss of information, be trivially represented, and is often displayed, as an unlabelled rooted graph. Therefore I argue that the correspondence between event structures and domains proposed in [17], and generalised to all event structures in [25], can equivalently, or maybe better, be regarded as a correspondence between event structures and a class of unlabelled transition systems or *process graphs*. As remarked in [7], this correspondence can trivially be extended to *labelled* event structures and transitions systems; the latter are easier to label than domains. It follows immediately that process graphs, or labelled transition systems, are at least as capable of expressing causality as labelled event structures.

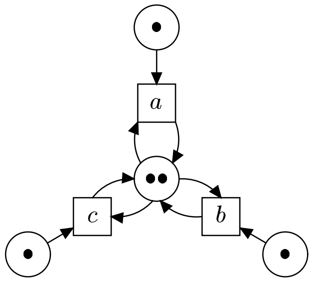
The computational interpretation of domains, inherited from that of event structures, naturally applies to the process graphs corresponding with those domains. These graphs capture causality through confluence of squares of transitions. This computational interpretation can be extended to process graphs that do not correspond to event structures or Scott domains. It can be seen as an enrichment of the classical interpretation of process graphs. Just like any process graph can be unfolded into a tree, while preserving its interleaving interpretation, I propose a causality respecting unfolding of arbitrary process graphs into so-called *history preserving* ones, which preserves transition squares. History preserving process graphs generalise the Scott domains originating from the general event structures of [25]. They can also model phenomena like resolvable conflict that are not expressible by these event structures.

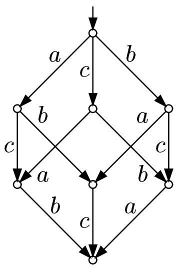
Several brands of transition systems enriched with some auxiliary structure to capture causality have been proposed as models of concurrency, cf. the *asynchronous transition systems* of SHIELDS [23] and BEDNARCZYK [2], the *behaviour structures* of RABINOVICH & TRAKHTENBROT [21], the *concurrent transition systems* of STARK [24] and DROSTE, [5] and the *transition systems with independence* of NIELSEN & WINSKEL [26]. In each of these cases the added structure does not fundamentally increase their expressiveness: after a suitable behaviour-preserving unfolding, the causalities expressed by this added structure are completely determined by the underlying transition system, which always forms a history preserving process graph.

Event automata, studied by PINNA & POIGNÉ [19], fit between configuration structures and ST-configuration structures. Through appropriate translations these can be shown to be equally expressive as the so-called *configuration-deterministic* process graphs. Graphs which are not configuration deterministic do not correspond to nets or event-oriented models. Interestingly, translating back and forth between event automata and process graphs may repeatedly increase the number of events of the event automaton representation of the system in question, namely by spitting events into subevents that occur in disconnected parts of the system representation. Hence these translations cannot be expressed as reflexions or coreflexions in a suitable categorical framework.

3 Higher dimensional automata

The concurrent interpretation of process graphs allows one to think of squares and cubes as being "filled in". PRATT [20] proposes a *geometric model of concurrency*, refining this approach by not necessarily filling in *all* squares and cubes, but explicitly filling in only those that one wants to represent concurrency. Alternative formalisations of this idea appear in VAN GLABBEEK [8], GOUBAULT & JENSEN [11] and CATTANI & SASSONE [4]. Although the resulting model of *higher dimensional automata* is more complicated than that of plain automata or process graphs, it is more expressive as well. The Petri net below, for in-

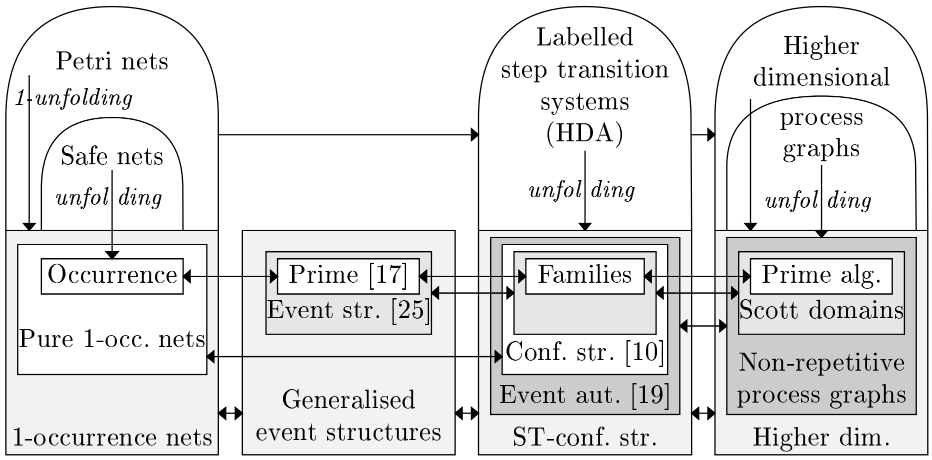




stance, is expressible by a higher dimensional automaton in the form of a cube, as displayed above, of which all 6 sides are filled in (representing the 6 possible concurrent firings of two transitions, either before or after the third one fires), but the interior is not. The causal behaviour of this system cannot be represented by a process graph, and hence neither by an event automaton, nor by a pure Petri net. Process graphs and the mentioned transition systems with extra structure to capture causality can be regarded as one- and two-dimensional automata, respectively.

A representation of higher dimensional automata in which the names of both events and actions are incorporated, is given by *labelled step transition systems* (LSTSs), see also Badouel [1]. These naturally unfold into (alternative representations of) ST-configuration structures. Petri nets translate to LSTSs by taking their marking graphs; LSTSs translate to higher dimensional automata as in [8] by forgetting event names (but remembering their labels).

Ehrenfeucht & Rozenberg [6] characterised which process graphs can be obtained as the marking graphs of a safe nets. Likewise, Mukund [16] characterised which LSTSs can be obtained as the step marking graphs of general Petri nets. Both papers also yield translations back from (step) transition systems to nets, but only for systems in the characterised class. More general translations from LSTSs to nets can be obtained through unfolding.



References

1. E. Badouel (1996): *Splitting of actions, higher-dimensional automata, and net synthesis.* Technical Report RR-3490, Inria, France.
2. M. Bednarczyk (1987): *Categories of asynchronous systems.* PhD thesis, Computer Science, University of Sussex, Brighton.
3. G. Boudol (1990): *Flow event structures and flow nets.* In I. Guessarian, editor: *Semantics of Systems of Concurrent Processes, Proceedings LITP Spring School on Theoretical Computer Science,* La Roche Posay, France, LNCS 469, Springer, pp. 62–95.
4. G.L. Cattani & V. Sassone (1996): *Higher dimensional transition systems.* In Proceedings 11^{th} Annual IEEE Symposium on *Logic in Computer Science* (LICS 96), New Brunswick, USA, IEEE Computer Society Press, pp. 55–62.
5. M. Droste (1992): *Concurrent automata and domains. International Journal of Foundations of Computer Science* 3(4), pp. 389–418.
6. A. Ehrenfeucht & G. Rozenberg (1990): *Partial 2-structures. Acta Informatica* 27(4), pp. 315–368.
7. R.J. van Glabbeek (1988): *An operational non-interleaved process graph semantics of CCSP* (abstract). In E.-R. Olderog, U. Goltz & R.J. van Glabbeek, editors: *Combining compositionality and concurrency, summary of a GMD-workshop, Königswinter, March 1988*, Arbeitspapiere der GMD 320, pp. 18–19.
8. R.J. van Glabbeek (1991): *Bisimulations for higher dimensional automata.* Email message, July 7, '91. Available at `http://theory.stanford.edu/~rvg/hda`.
9. R.J. van Glabbeek & U. Goltz (1990): *Refinement of actions in causality based models.* In J.W. de Bakker, W.P. de Roever & G. Rozenberg, editors: *Proceedings REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness,* Mook, The Netherlands, May/June 1989, LNCS 430, Springer, pp. 267–300.
10. R.J. van Glabbeek & G.D. Plotkin (1995): *Configuration structures (extended abstract).* In D. Kozen, editor: Proceedings 10^{th} Annual IEEE Symposium on *Logic in Computer Science* (LICS 95), San Diego, USA, IEEE Computer Society Press, pp. 199–209.
11. E. Goubault & T. Jensen (1992): *Homology of higher dimensional automata.* In W.R. Cleaveland, editor: *Proceedings CONCUR 92,* Stony Brook, NY, USA, LNCS 630, Springer, pp. 254–268.
12. J. Gunawardena (1992): *Causal automata. Theoretical Computer Science* 101, pp. 265–288.
13. V. Gupta & V.R. Pratt (1993): *Gates accept concurrent behavior.* In *Proc. 34th Ann. IEEE Symp. on Foundations of Comp. Sci.*, pp. 62–71. More material on Chu spaces can be found at `http://boole.stanford.edu/chuguide.html`.
14. P.W. Hoogers, H.C.M. Kleijn & P.S. Thiagarajan (1993): *Local event structures and Petri nets.* In E. Best, editor: *Proceedings CONCUR 93,* Hildesheim, Germany, LNCS 715, Springer, pp. 462–476.
15. J. Meseguer, U. Montanari & V. Sassone (1992): *On the semantics of Petri nets.* In W.R. Cleaveland, editor: *Proceedings CONCUR 92,* Stony Brook, NY, USA, LNCS 630, Springer, pp. 286–301.
16. M. Mukund (1992): *Petri nets and step transition systems. International Journal of Foundations of Computer Science* 3(4), pp. 443–478.
17. M. Nielsen, G.D. Plotkin & G. Winskel (1981): *Petri nets, event structures and domains, part I. Theoretical Computer Science* 13(1), pp. 85–108.

18. M. Nielsen, G. Rozenberg & P.S. Thiagarajan (1992): *Elementary transition systems.* *Theoretical Computer Science* 96, pp. 3–33.
19. G.M. Pinna & A. Poigné (1995): *On the nature of events: another perspective in concurrency.* *Theoretical Computer Science* 138(2), pp. 425–454.
20. V.R. Pratt (1991): *Modeling concurrency with geometry.* In *Proc. 18th Ann. ACM Symposium on Principles of Programming Languages*, pp. 311–322.
21. A. Rabinovich & B.A. Trakhtenbrot (1988): *Behavior structures and nets.* *Fundamenta Informaticae* 11(4), pp. 357–404.
22. D. Scott (1970): *Outline of a mathematical theory of computation.* In *Proceedings of the* 4^{th} *Annual Princeton Conference on Information Sciences and Systems*, pp. 169–176.
23. M.W. Shields (1985): *Concurrent machines.* *The Computer Journal* 28(5), pp. 449–465.
24. E.W. Stark (1989): *Concurrent transition systems.* *Theoretical Computer Science* 64, pp. 221–269.
25. G. Winskel (1987): *Event structures.* In W. Brauer, W. Reisig & G. Rozenberg, editors: *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course*, Bad Honnef, September 1986, LNCS 255, Springer, pp. 325–392.
26. G. Winskel & M. Nielsen (1995): *Models for concurrency.* In S. Abramsky, D.M. Gabbay & T.S.E. Maibaum, editors: *Handbook of Logic in Computer Science*, volume 4: Semantic Modelling, chapter 1. Oxford University Press.