Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 121 (2005) 143–155
www.elsevier.com/locate/entcs

Petri Nets With Persistence

Federico Crazzolara[1,2]

C&C Research Laboratories
NEC Europe, ltd.
Sankt Augustin, Germany

Glynn Winskel[3]

Computer Laboratory
University of Cambridge
Cambridge, England

Abstract

Persistence of information is common in modern computer systems. This paper describes how to extend Petri nets, a traditional model of concurrent and distributed computations, to take account of conditions that are persistent. We found use for this kind of nets in modelling untrustworthy networks on which messages are exchanged according to a security protocol. The paper explains a construction where persistent conditions are unfolded and a basic net is recovered. Conditions are given under which the unfolded net exhibits the same finite behaviours as the original net with persistence.

Keywords: Petri Nets, Persistent Conditions, Unfolding.

1 Introduction

Petri nets have been first introduced by C. A. Petri in the 60's, and today are a well known model for distributed and concurrent computations [1,5,9,11,12]. They are a so called "non-interleaving" model, where an event occurrence affects only a neighbourhood of events within a global state. Events that do

[1] Work done while at BRICS, Centre of the Danish National Research Foundation.
[2] Email: Federico.Crazzolara@ccrl-nece.de
[3] Email: Glynn.Winskel@cl.cam.ac.uk

1571-0661 © 2005 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2004.10.012

not affect part of the same neighbourhood are said to be independent and could potentially occur simultaneously.

Sometimes we have use for conditions which once established continue to hold and can be used repeatedly. This is true of assertions in traditional logic, for example, where once an assertion is established to be true it can be used again and again in the proof of further assertions. Similarly, if we are to use net events to represent rules of the kind we find in inductive definitions, we need conditions that persist. It can be convenient to use persistent conditions in modelling databases or shared memory systems where more than one user can read the same data. Petri nets with persistent conditions have been used to give semantics to a language specifically designed for modelling security protocols [3,4]. The model of the language uses a pool of persistent messages to which agents can send messages and from which messages can be read simultaneously by more than one agent. This model abstracts from real networks at a level which is convenient for studying security properties of protocols – some degree of hostility is already "built in" in the model, namely that messages can be duplicated and sent to wrong agents. A similar idea of messages that "persist" is used in strand spaces [10] and the inductive approach of Paulson [8] – both approaches have been successfully applied to the analysis of a number of security protocols.

When considering the finite behaviours of nets, it turns out that, under reasonable conditions, nets with persistent conditions are not more expressive than the simpler basic nets; we show how to unfold persistent conditions and yield a basic net out of a net with persistence. Nets with persistent conditions have arisen independently several times and have been studied for example in *contextual nets* [7] and as an extension of coloured nets with *test arcs* [2,6]. Even if nets with persistent conditions arose in previous studies we are not aware of a previous result that shows how to unfold them to basic nets. As shown in [3] our unfolding is useful in relating special purpose models of security protocols with more traditional models of concurrency.

2 General Petri nets

The explanation of general Petri nets involves a little algebra of multisets (or bags). It's convenient to also allow infinite multiplicities, so we adjoin an extra element ∞ to the natural numbers, though care must be taken to avoid subtracting ∞. ∞-Multisets support addition $+$ and multiset inclusion $\leq$, and even multiset subtraction $X - Y$ provided $Y \leq X$ and Y has no infinite multiplicities, in which case we call Y simply a multiset.

A *general Petri net* (often called a *place-transition system*) consists of

- a set of *conditions* (or *places*), P,
- a set of *events* (or *transitions*), T,
- a *precondition map* pre, which to each $t \in T$ assigns a multiset $pre(t)$ over P. It is traditional to write ${}^{\cdot}t$ for $pre(t)$.
- a *postcondition* map $post$ which to each $t \in T$ assigns an ∞-multiset $post(t)$ over P, traditionally written $t^{\cdot}$.
- a *capacity* function Cap which is an ∞-multiset over P, assigning a nonnegative number or ∞ to each condition p, bounding the multiplicity to which the condition can hold; a capacity of ∞ means the capacity is unbounded.

A state of a Petri net consists of a *marking* which is an ∞-multiset $\mathcal{M}$ over P bounded by the capacity function, *i.e.*

$$\mathcal{M} \leq Cap \ .$$

A marking captures a notion of distributed, global state.

Token game for general nets: Markings can change as events occur, precisely how being expressed by the transitions

$$\mathcal{M} \stackrel{t}{\rightarrow} \mathcal{M}'$$

events t determine between markings $\mathcal{M}$ and $\mathcal{M}'$. For markings $\mathcal{M}$, $\mathcal{M}'$ and $t \in T$, define

$$\mathcal{M} \stackrel{t}{\longrightarrow} \mathcal{M}' \text{ iff } {}^{\cdot}t \leq \mathcal{M} \text{ and } \mathcal{M}' = \mathcal{M} - {}^{\cdot}t + t^{\cdot} \ .$$

An event t is said to have *concession* (or be *enabled*) at a marking $\mathcal{M}$ iff its occurrence would lead to a marking, *i.e.*iff

$${}^{\cdot}t \leq \mathcal{M} \text{ and } \mathcal{M} - {}^{\cdot}t + t^{\cdot} \leq Cap \ .$$

There is a widely-used graphical notation for nets in which events are represented by squares, conditions by circles and the pre- and postcondition maps by directed arcs carrying numbers or ∞. A marking is represented by the presence of tokens on a condition, the number of tokens representing the multiplicity to which the condition holds. When an event with concession occurs tokens are removed from its preconditions and put on its postconditions with multiplicities according to the pre- and postcondition maps. Because of this presentation, the transition relation on Petri nets is described as the "token game".

3 Basic nets

We instantiate the definition of general Petri nets to an important case where in all the multisets the multiplicities are either 0 or 1, and so can be regarded as sets. In particular, we take the capacity function to assign 1 to every condition, so that markings become simply subsets of conditions. The general definition now specialises to the following.

A *basic Petri net* consists of

- a set of *conditions*, B,
- a set of *events*, E, and
- two maps: a *precondition* map $pre : E \to \mathcal{P}ow(B)$ and a *postcondition* map $post : E \to \mathcal{P}ow(B)$. We can still write ${}^{\bullet}e$ for the preconditions and $e^{\bullet}$ for the postconditions of $e \in E$ and we require ${}^{\bullet}e \cup e^{\bullet} \neq \emptyset$.

Now a *marking* consists of a subset of conditions, specifying those conditions which hold.

Token game for basic nets: Markings can change as events occur, precisely how being expressed by the transitions

$$\mathcal{M} \xrightarrow{e} \mathcal{M}'$$

events e determine between markings $\mathcal{M}, \mathcal{M}'$.

For $\mathcal{M}, \mathcal{M}' \subseteq B$ and $e \in E$, define

$$\mathcal{M} \xrightarrow{e} \mathcal{M}' \text{ iff } (1)\ {}^{\bullet}e \subseteq \mathcal{M} \ \&\ (\mathcal{M} \setminus {}^{\bullet}e) \cap e^{\bullet} = \emptyset \text{ (concession), and}$$
$$(2)\ \mathcal{M}' = (\mathcal{M} \setminus {}^{\bullet}e) \cup e^{\bullet}\ .$$

Property (1) expresses that the event e has *concession* at the marking $\mathcal{M}$. Returning to the definition of concession for general nets, of which it is an instance, it ensures that the event does not load another token on a condition that is already marked. Property (2) expresses in terms of sets the marking that results from the occurrence of an event. So, an occurrence of the event ends the holding of its preconditions and begins the holding of its postconditions. (It is possible for a condition to be both a precondition and a postcondition of the same event, in which case the event is imagined to end the precondition before immediately restarting it.)

There is *contact* at a marking $\mathcal{M}$ when for some event e

$${}^{\bullet}e \subseteq \mathcal{M} \ \&\ (\mathcal{M} \setminus {}^{\bullet}e) \cap e^{\bullet} \neq \emptyset.$$

The occurrence of an event is blocked through conditions, which the event should cause to hold, holding already. Blocking through contact is consistent with the understanding that the occurrence of an event should end the

holding of its preconditions and begin the holding of its postconditions; if the postconditions already hold, and are not also preconditions of the event, then they cannot begin to hold on the occurrence of the event. Avoiding contact ensures the freshness of names in the semantics of name creation.

Basic nets are important because they are related to many other models of concurrent computation, in particular, Mazurkiewicz trace languages (languages subject to trace equivalence determined by the independence of actions) and event structures (sets of events with extra relations of causality and conflict) – see [12].

4 Coloured nets

We briefly introduce coloured nets following Winskel [11]. Coloured nets have been first introduced by Jensen [5] and are a useful abbreviation of Petri-nets. We consider the case of coloured nets where all multisets have multiplicity either 0 or 1, and therefore can be regarded as sets (see for example [11] for a more general definition). This simpler case will be enough for the purposes of this paper.

Our description of coloured nets, like many others, uses the notion of places and transitions. These have a higher level nature and stand for sets of conditions and events. Colours are associated to places and transitions so that one can think of a condition as of a place in a certain colour and of an event as of a transition in a certain colour.

A *coloured net* consists of

- a set of *places*, P,
- a set of *transitions*, T,
- a colour function Δ which associates each place p with a set of colours $\Delta(p)$ and each transition t with a set of colours $\Delta(t)$,
- two maps: $pre, post : E \to \mathcal{P}ow(B)$, where

$$E = \{(t, c) \mid t \in T \text{ and } c \in \Delta(t)\}$$
$$B = \{(p, c) \mid p \in P \text{ and } c \in \Delta(p)\} \quad .$$

  If $e \in E$ then we abbreviate ${}^{\cdot}e = pre(e)$ and $e^{\cdot} = post(e)$. We require ${}^{\cdot}e \cup e^{\cdot} \neq \emptyset$ for all $e \in E$.

Markings for coloured nets consist of sets of conditions in B, which in this case are places possibly instantiated to a particular colour. The token game is the same as for basic nets:

Token game for coloured nets: For $\mathcal{M}, \mathcal{M}' \subseteq B$ and $e \in E$, define

$$\mathcal{M} \xrightarrow{e} \mathcal{M}' \text{ iff } (1)\ {}^{\cdot}e \subseteq \mathcal{M} \ \&\ (\mathcal{M} \setminus {}^{\cdot}e) \cap e^{\cdot} = \emptyset \text{ and } (2)\ \mathcal{M}' = (\mathcal{M} \setminus {}^{\cdot}e) \cup e^{\cdot}\ .$$

As we mentioned, coloured nets are an abbreviation for ordinary Petri-nets. As already shown in [11], the following proposition holds:

Proposition 4.1 *A coloured net* $(P, T, \Delta, pre, post)$ *determines a basic Petri-net* $(E, B, pre, post)$ *where*

$$E = \{(t, c) \mid t \in T \text{ and } c \in \Delta(t)\}$$
$$B = \{(p, c) \mid p \in P \text{ and } c \in \Delta(p)\}\ .$$

□

Thus a coloured net has exactly the same transitions as its basic Petri-net.

5 Nets with persistent conditions

Persistent conditions can be understood as an abbreviation for conditions within general nets which once they hold, do so with infinite multiplicity. Consequently any number of events can make use of them as preconditions but without their ever ceasing to hold. Such conditions, having unbounded capacity, can be postconditions of several events without there being conflict.

To be more precise, we modify the definition of basic net given above by allowing certain conditions to be *persistent*. A net with persistent conditions will still consist of events and conditions related by pre- and postcondition maps which to an event will assign a set of preconditions and a set of postconditions. But, now among the conditions are the persistent conditions forming a subset P. A marking of a net with persistent conditions will be simply a subset of conditions, of which some may be persistent.

A net with persistent conditions can be understood on its own terms, or as standing for a general net with the same sets for conditions and events. The general net's capacity function will be either 1 or ∞ on a condition, being ∞ precisely on the persistent conditions. When p is persistent, $p \in e^{\cdot}$ is interpreted in the general net as $(e^{\cdot})_p = \infty$, and $p \in {}^{\cdot}e$ as $({}^{\cdot}e)_p = 1$. A marking of a net with persistent conditions will correspond to a marking in the general Petri net in which those persistent conditions which hold do so with infinite multiplicity.

Graphically, we'll distinguish persistent conditions by drawing them as double circles:

Token game with persistent conditions: The token game is modified to

account for the subset of conditions P being persistent. Let $\mathcal{M}$ and $\mathcal{M}'$ be markings (*i.e.* subsets of conditions), and e an event. Define

$$\mathcal{M} \xrightarrow{e} \mathcal{M}' \text{ iff } {}^{\cdot}e \subseteq \mathcal{M} \;\&\; (\mathcal{M} \setminus ({}^{\cdot}e \cup P)) \cap e^{\cdot} = \emptyset \text{ (concession), and}$$
$$\mathcal{M}' = (\mathcal{M} \setminus {}^{\cdot}e) \cup e^{\cdot} \cup (\mathcal{M} \cap P) \; .$$

The token game fits our understanding of persistence, and specifically it matches the token game in its interpretation as a general net. In [4] and the thesis [3], these special contextual nets are used in modelling and analysing security protocols.

The token game of a net with persistent conditions could be generalised to transitions

$$\mathcal{M} \xrightarrow{A} \mathcal{M}'$$

where A is set or even an ∞-multiset of events. If one permits A to be a multiset then one might allow an event to occur in the net simultaneously more than once. In basic nets this would never be the case because all conditions are marked with multiplicity 1. In nets with persistent conditions however, one might want to permit simultaneous occurrences of those events whose pre- and postconditions are all persistent.

6 Nets with persistent conditions and basic nets

To unfold a net with persistent conditions to a basic net we first construct a coloured and then, through a well known unfolding, a basic net. A run of the net with persistent conditions relates to a run of the unfolded, basic net provided that any event that marks a persistent condition does not occur more than once in the run.

It is known how a coloured net with test arcs can be transformed to an equivalent coloured net without test arcs (see [2]). That construction, however, introduces infinite multiplicities in the unfolded net. Test arcs have a similar function to persistent conditions. If an event has a precondition connected with a test arc then it can occur only if the condition is marked; when the event fires, however, that condition is not consumed. If one tries to unfold nets with persistent condition as suggested in [2] then conditions with infinite capacity replace those with persistence. A general Petri-net is the result of the unfolding, as is the case for the unfolding of contextual nets shown in [7]. In this section we show a much stronger result; it tells how persistent conditions can be unfolded to yield a basic net where all multiplicities are either 0 or 1.

We make use of the following notation: Write

$${}^{\cdot}b = \{e \mid b \in e^{\cdot}\} \cup \{*\}$$

for all the events that mark the condition b. We add $*$ to indicate that b can be included in an initial marking and therefore need not be marked by an event. Write

$$b^{\cdot} = \{e \mid b \in {}^{\cdot}e\}$$

for the set of events that have b as one of their preconditions.

6.1 Unfolding persistent conditions

Consider a net with persistent conditions

$$(B, P, E, pre, post)$$

it unfolds to the coloured net

$$(B, E, \Delta, pre', post')$$

where Δ is a colouring function for events and conditions, and $pre', post'$ are pre- and postcondition maps from coloured events to sets of coloured conditions.

The colour of an event that has persistent preconditions is a tuple consisting of all its persistent preconditions and of a choice of events that marks them. We colour an event that does not have any persistent preconditions with the default colour δ. More precisely, for every event $e \in E$

$$\Delta(e) = \begin{cases} \{\delta\} & \text{if } {}^{\cdot}e \cap P = \emptyset \\ \prod_{b \in {}^{\cdot}e \cap P} (\{b\} \times {}^{\cdot}b) & \text{otherwise} \end{cases} .$$

The colour of a persistent condition consists of a set of pairs of events that can mark the condition with the events that require the condition marked. The colour of ordinary conditions instead is the default colour δ. More precisely, for every condition $b \in B$

$$\Delta(b) = \begin{cases} \{\delta\} & \text{if } b \in B \setminus P \\ {}^{\cdot}b \times b^{\cdot} & \text{otherwise} \end{cases} .$$

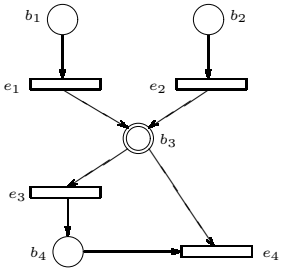
The precondition map of the coloured net is defined for pairs (e, c) of events e and colours $c \in \Delta(e)$ as follows:

$$pre'(e, c) = \{(b, \delta) \mid b \in {}^{\cdot}e \setminus P\} \ \cup \ \{(b, (e', e)) \mid (b, e') \in c\} .$$

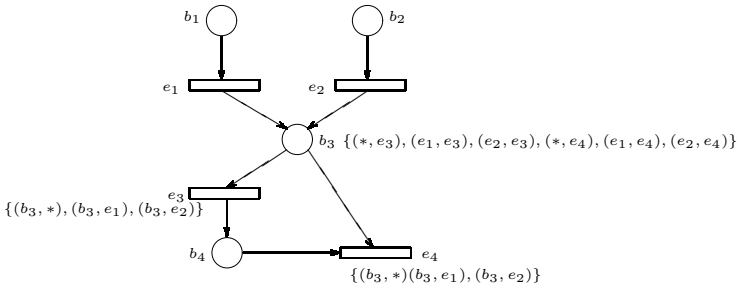
We use the shorthand notation $(b, e') \in c$ where $b \in B$ and $e' \in E$ when the colour c is a tuple and (b, e') is a component to which c projects. The postcondition map for pairs (e, c) where e is an event and $c \in \Delta(e)$ is defined as follows:

$$\begin{aligned} post'(e,c) = &\{(b,\delta) \mid b \in e^{\cdot} \setminus P\} \\ &\cup \{(b,(e,e')) \mid b \in e^{\cdot} \cap P \text{ and } e' \in b^{\cdot}\} \\ &\cup \{(b,(e',e)) \mid (b,e') \in c\} \ . \end{aligned}$$

As an example of the construction consider the following net with persistent conditions:



It unfolds to the following coloured net, where non-default colours are listed in curly brackets next to events and conditions:



As shown in Winskel's [11] a coloured net determines a Petri-net. In this case the coloured net obtained from a net with persistent conditions yields the basic net

$$(B', E', pre', post')$$

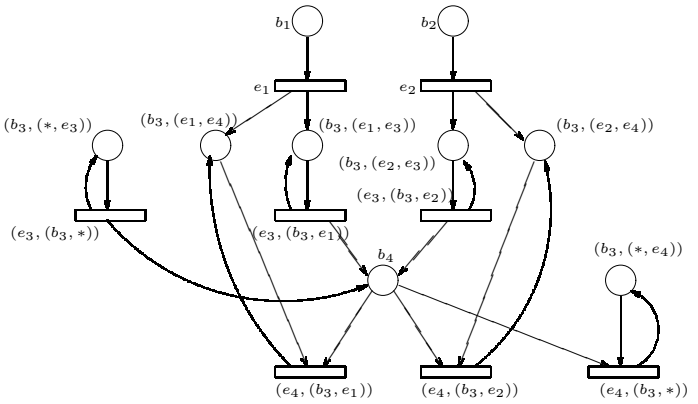
with conditions

$$B' = \bigcup_{b \in B} \{b\} \times \Delta(b)$$

and with events

$$E' = \bigcup_{e \in E} \{e\} \times \Delta(e) \quad .$$

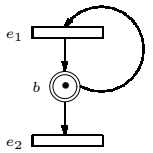
If one applies this unfolding to our example one obtains the basic net:



The construction that unfolds persistent conditions into ordinary conditions keeps track of how the conditions can be marked and of what events can consume a condition. Persistent conditions in a net can be part of the initial marking and therefore do not necessarily need an event to mark them. Conditions of the kind $(b, (*, e))$ are introduced so that the behaviour of a net that has the persistent condition b in its initial marking is still related to the unfolded net where the initial marking contains $(b, (*, e))$.

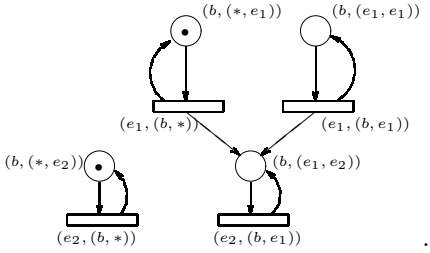
6.2 Relating nets with persistent conditions to basic nets

The construction that we described in the previous section, which shows how to unfold a net with persistent conditions into a basic net, is correct when the runs of the net it yields are substantially the same as the runs of the original net. Not all nets with persistence unfold well if one applies the described construction. Nets unfold well if they do not have runs where events with persistent postconditions appear more than once in the same run. The following example illustrates the reasons for this restriction. Consider the net:



.

The event e_1 can occur more than once in a run of the net, without the event e_2 necessarily having to occur. In the unfolded net, however, after the event $(e_1, (b, *))$ occurs once, it can not occur again.



.

Theorem 6.1 *Let N be a net with persistent conditions and N' be the basic net obtained from it by the described construction.*

(i) *To every run*

$$\mathcal{M}_0 \xrightarrow{e_1} \cdots \xrightarrow{e_{i-1}} \mathcal{M}_i \xrightarrow{e_i} \cdots$$

in N in which events that carry persistent postconditions occur at most once, there corresponds a run

$$\bar{\mathcal{M}}_0 \xrightarrow{(e_1,c_1)} \cdots \xrightarrow{(e_{i-1},c_{i-1})} \bar{\mathcal{M}}_i \xrightarrow{(e_i,c_i)} \cdots$$

in N' such that at every stage i in the run $c_i \in \Delta(e_i)$.

(ii) *To every run*

$$\bar{\mathcal{M}}_0 \xrightarrow{(e_1,c_1)} \cdots \xrightarrow{(e_{i-1},c_{i-1})} \bar{\mathcal{M}}_i \xrightarrow{(e_i,c_1)} \cdots$$

in N' there corresponds a run

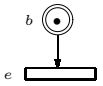
$$\mathcal{M}_0 \xrightarrow{e_1} \cdots \xrightarrow{e_{i-1}} \mathcal{M}_i \xrightarrow{e_i} \cdots$$

in N.

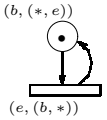
Proof. See [3]. □

Remark 6.2 For simplicity we described the relation between the runs of a net with persistent condition and the runs of the unfolded net only when one event at a time can fire. Petri-nets can describe true concurrent behaviour and

so one might want to generalise the above result to runs in which events can occur simultaneously. There is a restriction, however. In basic nets, an event can either happen or not happen, whereas in nets with persistent conditions an event may happen with some multiplicity. The following net, for example, permits the event e to happen with any multiplicity:



In its unfolding, however, the event e can happen any number of times but only once at a time:



One can therefore generalise the result in the previous section and relate transitions

$$\mathcal{M} \xrightarrow{A} \mathcal{M}'$$

in a Petri net with persistence to similar transitions in the unfolded basic net, provided A is a set and not a multiset.

References

[1] *Advanced course on Petri nets*, volume 254,255 of *LNCS*. Springer-Verlag, 1987.

[2] S. Christensen and N. D. Hansen. Coloured Petri Nets Extended with Place Capacities, Test Arcs and Inhibitor Arcs. In *Application and Theory of Petri nets, 14th International Conference*, volume 691 of *LNCS*, 1993.

[3] F. Crazzolara. Language, Semantics, and Methods for Security Protocols. PhD Thesis, BRICS, Computer Science Department, University of Aarhus, *BRICS Dissertation Series*, DS-03-4, May 2003.

[4] F. Crazzolara and G. Winskel. Events in security protocols. In *Proceedings of the Eight ACM Conference on Computer and Communications Security*, Philadelphia, November 2001. ACM Press.

[5] K. Jensen. Coloured Petri nets and the invariant method. *TCS*, 14, 1981.

[6] C. Lakos and S. Christensen. A General Systematic Approach to Arc Extensions for Coloured Petri Nets. In *Application and Theory of Petri nets, 15th International Conference*, volume 815 of *LNCS*, 1994.

[7] U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32(6), 1995.

[8] L. C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85–128, 1998.

[9] W. Reisig. *Petri Nets. An introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[10] J. Thayer, J. Herzog, and J. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*. IEEE Press, 1998.

[11] G. Winskel. Petri Nets, Algebras, Morphisms, and Compositionality. *Information and Computation*, 72:197–238, 1987.

[12] G. Winskel and M. Nielsen. *Handbook of Logic and the Foundations of Computer Science*, volume IV, chapter Models for concurrency. Oxford University Press, 1995.