Yale University
Department of Computer Science

Self-Stablizing Petri Nets

Gadi Taubenfeld

YALEU/DCS/TR-707
May 1989

This work was supported in part by the National Science Foundation under grant CCR-8405478, and by the Hebrew Technical Institute scholarship.

Self-Stablizing Petri Nets

Gadi Taubenfeld*
Computer Science Department, Yale University, New Haven, CT 06520

Abstract

The notion of self-stablization was first defined by Dijksra in 1974. In this short note we define, within the framework of Petri Nets, the notion of *self-stablizing petri nets* which are nets that can be recover from any possible failure. This notion is similar, but not identical, to Dijkstra original definition. We prove that not all nets can be transformed into self-stablizing nets. An example is given and future research directions are discussed.

Key words: petri nets, self-stabilization, failures.

1 Introduction

The design of fault tolerant concurrent system has been an active and creative field of research in the last several years. One fundamental notion in this field is the paradigm of self-stabilization which was defined by Dijkstra [Dij1]. A system which is composed of individual processes operating concurrently is said to be self-stabilizing with respect to some stable property p [1] if starting from any initial state the system is guarantee to converge within a *finite* number of steps to state in which p holds. Our intention is to formally define a similar notation within the framework of Petri Nets.

Self-stabilization is a most desired requirement for systems which are designed to operate in an environment where failures may occur, since systems which satisfy that requirement are highly robust. By a failure we understand an unexpected event such as lost or destruction of a message, lost of an information kept in the memory of a processor etc. An occurrence of a failure may leave the system in unpredictable state from which the system should continue operating. Since we view the state after a failure as a new initial state, it follows from the above description of self-stablizing systems, that no matter what may happen to the system as a result of a failure, (if the system is still operating then) it will always return to normal behavior within a finite number of steps after a failure occurs without the need for external interference.

As an example consider the case where two processors have to use a single common printer. In order to avoid the situation where both try to print simultaneously (which will

*Supported in part by the National Science Foundation under grant CCR-8405478, and by the Hebrew Technical Institute scholarship.

[1] A stable property is a property that once become true it remains true thereafter. The statements "the system is deadlocked" and "more than five messages have been received" are examples of stable properties.

result a mixed unuseful output), they participate in a mutual exclusion protocol in which the permission to use of the printer is a critical section. However, it is possible that as a result of a failure both processors will be in their critical sections trying to print simultaneously. A self-stabilizing mutual exclusion protocol would ensure that in such a case (where both processors have the permission to print) the problem will resolve after a finite number of attempts to use the printer.

Dijkstra's fundamental paper has inspired other researchers to explore this phenomenon. While some papers are concern with formal correctness proof and complexity issues of such systems [Dij2,Er], others include new design of self-stablizing systems [BGW,Go,Kr,La]. In this short note we define, within the framework of Petri Nets, the notion of *self-stablizing petri nets* which are nets that can be recover from any possible failure. This notion is similar (although not exactly identical) to Dijkstra's original notion of self-stablizing systems. We prove that there does not exist a complier which can translate any net to a similar net that is also self-stablizing. An example of a self-stablizing arbiter is given and future research directions are discussed.

2 Description of Petri Nets

In this section we give a brief and informal description of (super) Petri Nets (abbv. Nets) and of net languages. For general and formal definitions we refer the reader to [EY,YE].

A *net* is a 3-tuple $N = (P, T, V)$ where (1) P is a finite set of *places*; (2) T is a finite set of *transitions*; and (3) V is a function $V : (P \times T) \cup (T \times P) \rightarrow \{0, 1, I\}$. It is assumed that: $P \cap T = \emptyset$, $P \cup T \neq \emptyset$ and $V(T \times P) \subseteq \{0, 1\}$. If $V(x, y) \neq 0$ then we say that there is an arc from x to y. The symbol I indicating an '*Inhibiting*' arc. '*Inhibiting*' arcs are intended to model the fact that taking a move is enabled only when the value of a counter zero. A transition t can be *fired* if and only if there is no token in any place p where $V(p, t) = I$, and there is at least one token in any place p where $V(p, t) = 1$.

We assume that the reader is familiar with the graphical representation of nets and the concepts of marking, firing, firing sequence, and multiple-firing sequence. Let N be a net and M its initial marking. For a marked net $S = (N, M)$ we define $L(S)$ to be a (sequential) language over the alphabet T, consisting of all firing sequences of S. Similarly, we define $\pi(S)$ to be a (parallel) language over the alphabet $2^T - \{\emptyset\}$, consisting of all multiple-firing sequences of S.

For a marked net $S = (M, N)$, we use the notation $M[w > M'$ which means that M' is obtained from M by firing the sequence w, and say that M' is reachable from M via w. Also, we use the notation $M[w >$ which means that there exists some marking, say M', such that $M[w > M'$. We call the set of all the markings reachable from M, denoted by $[M]_N$, the set of the *legitimate* markings (we omit the subscript N when it is understood from the context). Finally, $\mathcal{M}(N)$ is used to denote the (possibly infinite) set of all possible markings of the net N.

3 Self-Stablizing Petri Nets

In this section we introduce the notion of a net that can be recover from any possible failure. A failure in the context of petri nets means that an arbitrary number of tokens can appear or disappear. That is, any possible marking of the net may result as a consequence of a failure. We emphasis that after a failure of a marked net $S = (M, N)$ the net may be marked with an unlegitimate marking which is a marking that is not reachable from the initial marking M. Informally, we say that S is a self-stablizing net if after the occurrence of a failure the net will always reach a legitimate marking in a finite numbers of steps assuming no other failure occurred in the meantime.

We say that a marking M' leads to $[M]$, to be denoted $M' \leadsto [M]$, iff there exists a non-negative integer k such that for every sequence w where $w \in T^*$ it is the case that $(M'[w > M'' \wedge |w| \geq k) \Rightarrow (M'' \in [M])$. That is, after a finite number of steps starting from M' the net always reaches a marking which belong to $[M]$. In the above definition w is a (single-) firing sequence (i.e., $w \in T^*$). However, allowing w to be a multiple-firing sequence (i.e., $w \in (2^T)^*$) would make no difference.

Definition: A marked net $S = (M, N)$ is *self-stablizing* iff $\forall M' \in \mathcal{M}(N) : M' \leadsto [M]$.

The notion of self-stablizing net can be strengthen by requiring that after a failure occurs the net will always return within a finite number of steps from an unlegitimate marking to the *initial* marking. In such a case we call the net a *strongly self-stablizing net.* On the other hand it can be weaken by adding the fairness requirement that any transition which is continuously enabled for some pre-determined (big enough) number of steps must be than fired. In such a case we call the net a *weak self-stablizing net.* Other fairness assumptions can be adopted instead of the previous one.

4 Example: An Arbiter

In this section we show how an arbiter which is not self-stablizing can be transformed into an arbiter which is self-stablizing. The net in Figure 1, which consists of the **thick** lines and circles, is the non self-stablizing arbiter, and the *dashed* lines describe what should be added to that arbiter in order to make it weak self-stablizing.

The construction of an arbiter is motivated by the need to synchronize processes that want to use shared resources. The arbiter in Figure 1, describes two processes each one of them continuously trying to enter its critical section. The correctness condition for the arbiter is that the two processes are never in their critical sections simultaneously. In terms petri nets, this means that markings in which there are tokens in both critical section 1 and critical section 2 are unlegitimate markings.

5 A Limitation of Self-Stabilization

The problem of constructing (if possible) a self-stablizing net from a net which is not self-stablizing is an interesting problem. A transaction t is said to be *dead* at $S = (M, N)$ iff there does not exists $M' \in [M]$ such that $M'[t >$. Similarly, an arc is dead if removing it makes no difference (assuming there are no failures). Let us denote by $\mathcal{F}(S)$ the marked

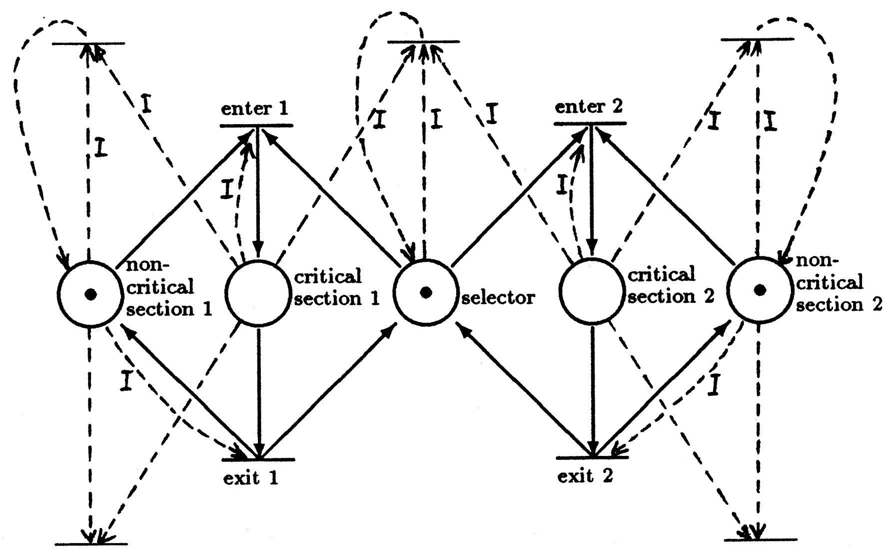

Figure 1. Weak self-stablizing arbiter.

net which result from S by first omitting all dead transactions and all dead arcs and than omitting all isolated places. Clearly $L(S) = L(\mathcal{F}(S))$, and also $\pi(S) = \pi(\mathcal{F}(S))$. The problem of constructing a self-stablizing net from a net which is not self-stablizing can now be formulated as follows: Given a net S construct a self-stablizing net S' where $S = \mathcal{F}(S')$. Next we prove that it is not always possible to do so.

Definition: A marked net $S = (M, N)$ is *potentially self-stablizing* iff there exists a self-stablizing marked net S' such that $S = \mathcal{F}(S')$.

Theorem: Not all marked nets are potentially self-stablizing.

Proof: The marked nets S_1 and S_2 in Figure 2, are examples for nets that are not potentially self-stablizing. To see that, we observe that each one of these marked nets has only one possible reachable marking, and the reachable marking of S_1 is different from that of S_2. Also, in both marked nets the transition t can always be fired. Now, assume to the contrary that there exists a self-stablizing net S'_1 such that $S_1 = \mathcal{F}(S'_1)$. From the definition of S'_1 it follows that, if there is a single token in the place p in the net S'_1 then the transition t will forever be the only transition that can be fired and there will always be exactly one token in p (assuming no failure occurs). However, from the definition of Petri nets, it follows that if (after a failure) there are two tokens in the place p in the net S'_1 then the transition t will forever be the only transition that can be fired and there will always be two tokens in p. This contradicts the assumption that S'_1 is self-stablizing. $\square$

We notice that the theorem holds also if we assume stronger type of nets which have also emptying arcs, logical arcs and absorbing places and restricted places [EY]. Also, the theorem holds assuming weak self-stablization.

We say that a language L is self-stablizing iff there exists a self-stablizing net S such that $L = L(S)$. It is interesting to investigate the properties of self-stablizing languages. For example, since self-stablizing nets are close under union, it follows that self-stablizing

Figure 2. Examples for marked nets which are not potentially self-stablized.

languages are close under the shuffle operator. Another question is how to determine for a given net wither it is self-stablizing net or not.

Acknowledgements: I am grateful to Michael J. Fischer and Michael Yoeli for helpful discussions concerning this work.

References

[BGW] Brown, G.M., Gouda M.G., and Wu, C.L. Token systems that self-stabilize, *IEEE Transaction on Computers*, 1989, to appear.

[Dij1] Dijkstra E.W. Self-stablizing systems in spite of distributed control, *CACM*, Vol. 17, 1974, 643-644.

[Dij2] Dijkstra E.W. A belated proof of self-stabilization, *Distributed Computing*, Vol. 1, 1986, 5-6.

[Er] Chang J.H. Ernest. On the cost of self-stabilization, *IPL* Vol. 24, 1987, 311-316.

[EY] Etzion, T., and Yoeli, M. Super-nets and their hierarchy, *Theoretical computer science*, Vol. 23, 1983, 243-272.

[Go] Gouda M.G. The stabilizing philosopher: asymmetry by memory and by action, *Science of computer programming*, 1989, to appear.

[Kr] Kruijer, H.S.M. Self-stabilization (in spite of distributed control) in tree-structured systems, *IPL* Vol. 2, 1979, 91-95.

[La] The mutual exclusion problem: statement and solutions, *JACM*, Vol. 33, 1986, 327-348.

[YE] Yoeli, M., and Etzion, T. Behavioral equivalence of concurrent systems, *Application and theory of petri nets*, Springer Verlag, 1983, 292-305.