行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※個人化網路教學系統之服務控管與排程研究(III)※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：整合型計畫

計畫編號：NSC89-2218-E-002-029

執行期間：89 年 8 月 1 日至 90 年 7 月 31 日

計畫主持人：顏嗣鈞

本成果報告包括以下應繳交之附件：

- □赴國外出差或研習心得報告一份
- □赴大陸地區出差或研習心得報告一份
- ■出席國際學術會議心得報告及發表之論文各一份
- □國際合作研究計畫國外研究報告書一份

執行單位：國立台灣大學電機工程系

中　華　民　國89年10月25日

行政院國家科學委員會專題研究計劃成果報告

個人化網路教學系統之服務控管與排程研究(III)

Service Control and Scheduling of a Personalized Networked Education System (3)

計畫編號: NSC89-2218-E-002-029

執行期間:89 年 8 月 1 日至 90 年 7 月 31 日

計畫主持人: 顏嗣鈞 教授 國立台灣大學電機工程系

計劃參與人員: 林義凱、鄧德雋、張雲璿、秦紹祈

一、摘要

我們在這計劃中嘗試找到在圖狀結構下代理人伺服器的擺設演算法。我們的演算法是根據樹狀結構下找到最佳伺服器的擺設演算法加以修改，擴充到圖狀結構下(Graph)並加以實作。我們並針對整體網路延遲(overall delay of the network)與最佳解(用窮舉法得知)比較。

關鍵字：代理人伺服器，網路快取

The objective of this project is to develop an algorithm to place proxies in a given network environment. Our algorithm, designed for networks of arbitrary topologies, is built upon an algorithm that is optimal for tree-topology networks. Simulation has been conducted to evaluate the performance of our algorithm.

Keyword: Proxy, Web Caching

二、計畫緣由與目的

WWW 的網站資料是現行網際網路上最熱門的資料之一，代理人伺服器在增近讀取網站資料的效能上扮演著很重要的角色。在增進網路效能上，代理人伺服器是一個很普及的方法。它利用了類似快取記憶體(Cache)的機制，把遠端的資料儲存在近端的機器上。當近端有其它的機器也要讀取同一份資料的話，則不必再去跟遠端的機器上要資料，只要從進端的代理人伺服器機器讀取就好，如此一來，不但能減少對遠端的流量，更能大幅縮短我們等待資料的時間，增進網路上的效能。但是在現實世界中，對一個網域而言，近端有幾個代理人伺服器通常是網路服務提供者(Internet Service Provider)設的那幾部。現在我們就來試著扮演網路服務提供者的角色，試著決定我們的代理人伺服器該架在那裡。

我們可以用一個無向圖(undirected graph) G (V, E)來表示我們的網際網路。每個節點 v 有一個 $freq(v)$ 表這個節點所產生的流量(Traffic)。每個連結(Edge)有一個 $delay$ 表示這條連結的延遲。還有 m 個代理人伺服器 ($m < |V|$) 待我們架設。為了讓問題不要太複雜，我們讓每個 $freq(v)$ 和各連結的 $delay$ 不隨時間改變。

如此我們就有一個網際網路的架構，並可以表示各個結點的所產生的流量，以及每條連結的延遲。

在說明過代理人伺服器如何在模型中運作之後，現在我們定義這一篇所要探討的問題：我們定義 P 為各個被設為代理人伺服器節點的集合，然後對每一個節點 v 我們可以定義 $delay\ (v, P)$ 為到最近的代理人伺服器的延遲。

我們並定義整體延遲為：

$$\sum_{v \in V} freq(v) \cdot delay(v, P)$$

我們現在要在 G(V，E)下架設 m 個代理人伺服器，使得整體延遲盡量小。也就是說我們要用演算法來決定 P 這個集合，使得整體的延遲盡量小。

三、演算法部份

這裡討論兩種樹狀結構的演算法，並應用在圖狀網路架構上。

(一)三部份 Dynamic Programming 演算法

首先我們先介紹三部份 Dynamic Programming 演算法，它大部份的模型架構和本篇所題出的模型類似，但是有幾個地方稍有出入。首先它的模型是樹狀結構，其次是它假設樹根節點為唯一對外連結。

在一個樹狀結構下，以及樹根節點為唯一對外連結的假設下，有一個很直觀的節點是最適合架代理人伺服器的，那就是根(Root)。所以當題目要解 m=1 的時，我們就直接把代理人伺服器定在這個樹的根。

當 m≧2 時，首先我們先把一個定在根，我們將第二個代理人伺服器架在樹上的時候，我們可以把一個樹分為三個部份。

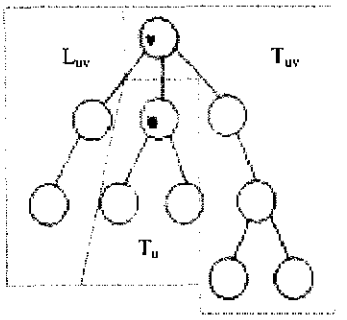

圖 3.1　樹的分割圖

T_u :以 u 節點為根的子樹(Subtree)。

T_{uv}: u 節點之右邊，(包含 v 節點)所形成的樹。

L_{uv}：u 節點之左邊，所有節點的集合。

當我們再架下一個代理人伺服器的時候，可以再把 T_u 或 T_{uv} 以同樣的方式分割下去。

當一個樹 T_v 被分割成三個部份之後，我們要架設 t 個代理人伺服器，希望得到的最小整體延馳 $C(v,t)$ 可以用下面的式子表示：

$$C(v,t)=\min_{u \in T_v} \min_{0<t'<t} (W(u,v)+C(u,t')+C(u,v,t-t'))$$

if t >1

其中 $W(u,v)$ 表示 L_{uv} 裡頭所有節點到根的整體延遲，$C(u,t')$ 表示 T_u 下架設 t' 個代理人伺服器的最佳解，$C(u,v,t-t')$ 則表示 T_{uv} 下架設 t-t' 個代理人伺服器的最佳解，當我們的 T_u 和 T_{uv} 所有可能的子樹的最佳解都已解出來的時候，我們只要找出最佳 u 和 t' 組合，我們就可以求出 T_v 下架設 t 個代理人伺服器的問題了。而我們所分出來的子樹 T_{uv} 我們也有類似的式子表可以求它的最佳解：

$$C(u,v,t)=\min_{x \in T_{uv}} \min_{0<t'<t} (W(u,v,x)+C(x,t')+C(v,x,t-t'))$$

if t >1

當我們要解 m 個代理人伺服器架在 T_v 下的問題時，我們先建立一個代理人伺服器表格(Proxy Table)，其中第一個行列出所有可能的的 u v 組合，最多是 n^2 個，在表格裡每一個欄位只要決定它的(u，t)，所以決定每一個欄位需要的時間複雜度是 O(nm)，這個表格最多有 n^2m 個欄位，所以建好這個表格的複雜度是 $O(n^3m^2)$ 。

(二)二元樹化演算法

P. Krishnan[12]這一篇也有介紹解樹狀結構的方法，而且所用的模型和前一節所介紹很相似。所以它仍有一些和本篇模型相出入的地方，它假設樹的根結點為網頁的資料來源(Data Source)。同樣利用 Dynamic Programming 的方法，但是所分割的方法不一樣。它可以先對一般的樹狀結構以插虛節點(Dummy Nodes)的方法，讓整個樹狀結構形成二元樹(Binary Tree)。

形成二元樹之後，對每個以節點 i，有以 i_L 和 i_R 兩個子節點。

整個樹的高(height)為 h，而我們定由根往下第 l 層設有最接近根節點的代理人伺服器。

然後我們定義我們設節點 i 為代理人伺服器，以為 i 根的子樹下設置 t 個代理人伺服器所得到的最小整體延遲如下式。

$$C(i,t,l) = \min_{0\le t'\le t}(C(i_L,t',l+1)+C(i_R,t-t',l+1) + F(i_L,t',l+1)+F(i_R,t-t',l+1))$$

有了這個關係式之後，以一個二元樹要設 t 個代理人伺服器為例，我們可以由底往上地試出右子樹設 t'個代理人伺服器，右子樹則設 $t-t'$可以得到最小的整體延遲。其中 $F(i_L,t',l+1)$ 和 $F(i_R,t-t',l+1)$ 兩項表示當 i_L 和 i_R 不再設為代理人伺服器時，以 i_L 和 i_R 為根的子樹所增加的整體延遲。

如此我們一樣建立一個表格，其中第一個行列出所有可能的的 il 組合，最多是 nh 個，在表格裡每一個欄位只要決定它的 t'，所以決定每一個欄位需要的時間複雜度是 $O(m)$，這個表格最多有 nhm 個欄位，所以建好這個表格的複雜度是 $O(nhm^2)$之內。而實際上這個方法的複雜度是在 $O(nhm)$。

(三)兩種演算法的比較

以上兩種演算法的比較上，後者所提出的方法所需要的時間複雜度 $O(nhm)$優於前者所提的方法所需的時間複雜度 $O(n^3m^2)$。兩者所需的儲存空間比較上，後者所需要的 $O(nhm)$也比前者所需要的 $O(n^2m)$的空間小一點。

而兩者所求出來的都是樹狀結構的最佳解，除了兩者對根節點的定義不同，前者所假設的是根節點為此子樹的唯一對外連結，所以必定設為代理人伺服器，後者卻設根節點為網頁資料來源，故不必設為代理人伺服器。也就是說，在同一顆樹下，我們套用前者的方法設置 m 個代理人伺服器，和用後者這個方法設置 m-1 個代理人伺服器，這兩種方法除了前者要多擺一個代理人伺服器在根節點之外，其它的代理人伺服器兩者擺的位置應該要一樣，所得到的最小延遲也是一樣的答案。

(四)應用在圖狀結構演算法

這裡所題出的演算法主要是基於上面所題出對樹狀架構的最佳演算法來當作我們的演算法中的一個小區塊(Block)，現在我們要在圖狀結構下解這種問題的話，我們把這個分為三個步驟：

1. 首先我們對每個節點都建立展開樹

(Spanning Tree) T。

2. 以樹狀結構最佳化的演算法解在每個展開樹下代理人伺服器 Server 該架在那些節點上。
3. 重新以圖狀結構估計最小整體延遲，並由 n 個展開樹中挑出最小的整體延遲作為我們的解。

四、模擬結果

我們的模擬程式包含三大部份，第一部份是隨機圖狀結構產生器(Random Graph Generator)，它將產生一個相連的圖狀結構 (Connected Graph)，來表示各種不同的網際網路架構，給我們將要比較的兩個演算法來比較。

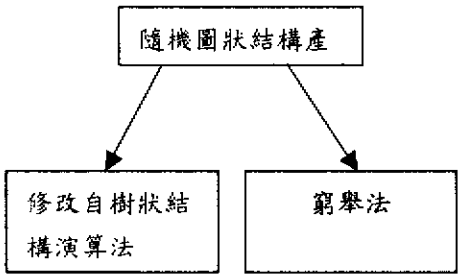

圖 4.1　整體模擬程式大略架構圖

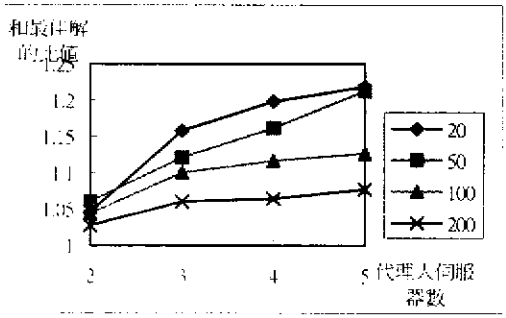

圖 4.2　整體延遲比較圖

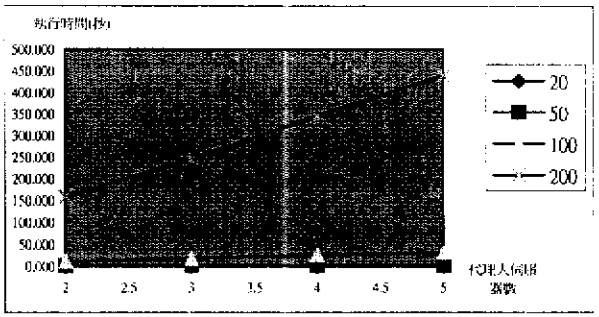

圖 4.3　修改後演算法的計算時間

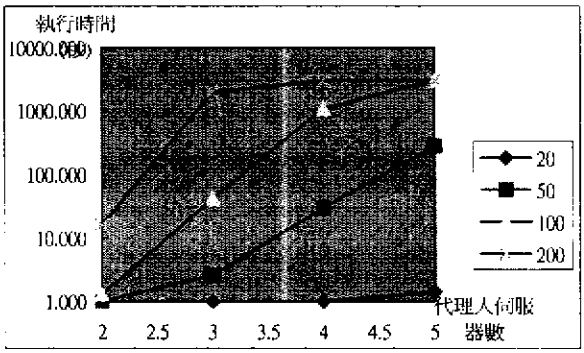

圖 4.4　窮舉法的計算時間

五、未來展望

現行的網路模型多由最短路逕(Shortest Path)來建構出樹，這個方法所建出來的樹和我們的演算法中所用的 Dijkstra 最小延遲展開樹的方法其實很類似，可惜的是這樣子所建構出來的樹，即使套用可以對樹狀結構擺設代理人伺服器達到最佳化的演算法，仍然離最佳的整體延遲有一段差距。

關於如何再進一步加強我們的演算法，我們可以從解樹狀結構的演算法著手，[12]這篇也有提出解出樹狀結構的演算法，而它的時間複雜度為 $O(mhn)$，其中 h 為該樹的高度(height)。但是如同前面所說的，這一個演算法在實作上的難度較高，由於插入虛節點，這個動作需要動態記憶體配置的架構，然後要重建一顆樹。這些動作在實作上的難度遠比用一個陣

列作對映，幾個旗標標示是否該連結為展開樹的連結大得多。但是這仍然是一個可以加速我們演算法的可行方案。

六、參考文獻

[1] M. Abrams, et al., "Caching Proxies: Limitation and Potentials", *Proc. 4th Int'l Conf. WWW*, Calif., pp. 312-319, 1995.

[2] M. F. Arlitt and C. L. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", *IEEE Trans. On Networking*, Vol.5, No.5, Oct. 1997.

[3] J. Bolot and P. Hoschka, "Performance Engineering of the World Wide Web: Application to Dimensioning and Cache Design", *Proc. 5th Int'l Conf. WWW*, Calif., 1996.

[4] H. W. Braun and K. C. Claffy, "Web Traffic Characterization: An assessment of the Impact of Caching Documents from NCSA's Web Server", *Computer Networks and ISDN Systems*, Vol. 28, No. 1-2, Dec. 1995.

[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications", *Proc. of IEEE INFOCOM'99*, New York, Mar. 1999.

[6] P. Cau and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms", *Usenix Symp. Internet Technologies and Systems*, Calif., pp.193-206, Dec. 1997.

[7] J. Dilley and M. Arilitt, "Improving Proxy Cache Performance: Analysis of Three Replacement Policies", *IEEE Internet Computing*, pp. 44-50, Nov.-Dec. 1999.

[8] L. Fan, P. Cao, J. Almeida, and A.Z. Broder, "Summary cache: A scalable wide-area Web cache sharing protocol", in *ACM SIGCOMM*, Sept. 1998, pp. 254-265.

[9] S. Gadde, M. Rabinovich, and J. Chase, "Reduce, reuse, recycle: An approach to build large internet caches", in *Proc. 1997 Conf. Hot Topics in Operating Systems*, 1997.

[10] M. R. Garey and D. S. Johnson, *Computerand Intractability : A Guild to the Theory of NP-Completeness*. San Francisco, CA:Freeman, Nov. 1979.

[11] X. H. Hia, D.Y. Li, X.D. Hu,H. J. Huang, and D. Z. Du, "Optimal Placement of Proxies of Replicated Web Servers in the Internet" , *WISE 2000*, pp. 55-61, 2000.

[12] P. Krishnan, D. Raz, and Y. Shavitt, "The Cache Location Problem", *IEEE/ACM Trans. on Networking*, Vol. 8, No. 5, Oct.

2000

[13] S. Jun and A. Bestavros, "Popularity-Aware Greedy Dual-Size Web Proxy Caching Algorithms", *Proc. of ICDCS'2000: The IEEE International Conference on Distributed Computing Systems*, Taiwan, May. 2000.

[14] P. Krishnan and B. Sugla, "Utility of co-operatiing web proxy caches", in *Proc. 7th Int. World Wide Web Conf.*, Brisbane, Australia, Apr. 1997.

[15] B. Li, M. J. Golin, G. F. Itlianoo, and X. Deng, "On the Optimal Placement of Web Proxies in the Internet", *Proc. of IEEE InfoCom'99*, pp. 1282-1290, Mar. 1999.

[16] R. Malpani, J. Lorch, and D. Berger, "Making world wide web caching servers cooperate", *in Proc. 4th Int. World Wide Web Conf.*, Dec. 1995.

[17] A. Tamir, "An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs", *Oper. Res. Lett.*, Vol. 19, pp. 59-64, 1996

出席會議報告

顏嗣鈞
國立台灣大學電機系

一、 參加會議經過

主持人於計畫執行期間，出國參與以下兩項國際會議，發表論文:

1. 「第五屆國際離散事件系統會議」(5th International Workshop on Discrete Event Systems)
2. 「第七屆 IEEE 國際複雜電腦系統工程會議」 (7th IEEE International Conference on Complex Computer Systems)

「第五屆國際離散事件系統會議」(5th International Workshop on Discrete Event Systems, WODES 2000)於 2000 年八月二十一至二十三日於比利時的根特市中著名的 University of Ghent 校園內舉辦。WODES 為離散事件系統研究領域最重要的國際會議。會議論文集並由 Kluwer Academic Publishers 出版為專書「Discrete Event Systems: Analysis and Control」。

「第七屆 IEEE 國際複雜電腦系統工程會議」 (7th IEEE International Conference on Complex Computer Systems) 於六月十一日至十三日於瑞典的 Skovde 市舉行。此會議為 IEEE 主辦的年會，此次由瑞典的 University of Skovde 協辦。Skovde 市位於首府斯德哥爾摸(Stockholm)西南方約三小時的車程。本會議的主題在於探討複雜軟、硬體系統中之 Specification, verification 以及 Analysis 問題。隨著實際系統之複雜度與日俱增，這些議題越來越重要。

二、 與會心得

「國際離散事件系統會議」 與 「IEEE 國際複雜電腦系統工程會議」此二會

議的特色之一在於其專精，以及對於論文審查之嚴謹。「國際離散事件系統會議」為離散事件系統研究領域最重要的國際會議。該領域的國際著名專家學者都會出席。筆者為國內唯一參與該會議發表論文的。發表的論文題目為「Fair Control of Omega Automata」，探討如何將「公平性」 (fairness) 的觀念，引進離散事件的控制上。筆者講演後，離散事件系統研究上國際權威學者 University of Toronto 的 Prof. W. M. Wonham 也親自來向筆者表達他的興趣，並提供許多寶貴意見。

根據「IEEE 國際複雜電腦系統工程會議」大會資料，此次論文錄取率約 40%。每份論文均經兩位以上學者專家評審，並附詳細審查資料。筆者為國內唯一參與該會議發表論文的。發表的論文題目為「Analysis of Self-Stabilization for Infinite-State Systems」，由演算法的角度來探討無限狀態系統(infinite-state systems)之自穩定性研究。「自穩定性」與「容錯」有密不可分的關係。

筆者參加的兩項會議理論與實際兼顧，筆者的論文偏重於理論方面的探討。為了結合工業界的應用，大會也安排了數場軟體工具(software tool)系統的展示，並應用於分析設計實際複雜系統上。會議的傳統是同時間只有一 session，所以與會專家學者均能充分討論，交換研究心得，氣氛融洽。筆者能與他們交換研究心得，收穫良多。

三、 建議

近幾年筆者深深感覺到，電腦理論(Theoretical Computer Science)相關的國際會議少有國內學者參加。這是一項質得我們重視的問題。相關單位應積極鼓勵國內年輕優秀的研究人員，積極重事理論方面的基礎研究，參與國際活動，才能迎頭趕上國際水準。

四、 攜回資料名稱及內容

1. 「Discrete Event Systems: Analysis and Control」(edited by R. Boel and G.

Stremersch) ，Kluwer Academic Publishers 2000 年出版的專書

2. Proceedings of 7th IEEE International Conference on Complex Computer Systems 會議論文集

五、 其它

十分感謝國科會補助「第五屆國際離散事件系統會議」的旅費、註冊費以及生活費。以及「第七屆 IEEE 國際複雜電腦系統工程會議」的註冊費以及生活費。

FAIR CONTROL OF ω-AUTOMATA

Hsu-Chun Yen
Dept. of Electrical Engineering, National Taiwan University
Taipei, Taiwan 106, R.O.C.
yen@ee.ntu.edu.tw

Keywords: Controllability, fairness, ω-automata.

Abstract Given a Rabin automaton, we consider the problem of computing the *fair controllability subset* of states, which consists of those states from which the automaton can be guided by a 'fair' *controller* to meet the acceptance condition of the automaton. The problem is investigated with respect to three notions of fairness, namely, *impartiality*, *justice* and *fairness*. By relating fair controllability to the existence of certain types of *strongly connected components*, we propose a graph-theoretic approach, which runs in polynomial time, for solving the problem with respect to all three fairness notions. A matching PTIME-hardness result is also shown.

1. INTRODUCTION

The study of control theory in discrete event dynamic systems [3] has benefited a great deal from the knowledge of *automata theory* and *formal languages*. Among problems of interest regarding the infinite behaviors of discrete event dynamic systems, the so-called *controllability subset* of an ω-automaton captures the set of states from which the operations of the automaton can be 'controlled' in such a way that its computation meets the predefined specification. In [6], a fixed-point characterization has been established to capture the controllability subset of a deterministic Rabin automaton, yielding an exponential time algorithm. (The decision version of the problem was shown to be NP-complete in [6].) In a subsequent article by Thistle [4], the work of [6] has been extended to incorporate a *liveness* constraint into the acceptance condition. More recently, Thistle and Malhamé [5] have examined the controllability subset problem with the so-called *state fairness* assumption imposed on the acceptance condition. What *state fairness* implies is that if a transition is enabled *infinitely often* under the control mechanism, then the transition

must be taken infinitely many times in the accepting computation. Like [6], in [5] the controllability subset under the state fairness assumption for deterministic Rabin automata was computed through a fixed-point characterization. With the additional state fairness constraint imposed on the acceptance condition, the problem becomes solvable in polynomial time, as reported in [5].

The control mechanism employed in the setting of [4, 5, 6] is the following. The set of transitions of a Rabin automaton is partitioned into a group of (not necessarily disjoint) sets $\{\Gamma_1, ..., \Gamma_n\}$, where each Γ_i is a subset of transitions, and the control policy is to choose, at any point in the course of the computation, a Γ_i which governs those transitions that are entitled to proceed. In addition to the state fairness condition assumed in [5], in this paper we investigate the controllability subset problem with respect to three types of *fair control policies*, namely, *impartial*, *just* and *fair* policies. *Impartiality*, *justice* and *fairness* are three notions of fairness introduced in [1]. We give a graph-theoretic approach to computing the controllability subset of a Rabin automaton with respect to the above three notions of fairness. Our approach is based on the idea of relating controllability to the existence of a certain type of *strongly connected components* (*SCC*s, for short) in the so-called *execution graph* of the automaton. As it turns out, our graph-theoretic treatment of the fair controllability issue yields a polynomial time algorithm for computing the controllability subset of a Rabin automaton (with respect to all three fairness notions). As for the lower bound, we are able to show the decision version of the problem to be PTIME-hard.

2. PRELIMINARIES

Given a finite *alphabet* Σ, we write Σ^* (resp., Σ^ω) to denote the set of all strings of finite length (resp., infinite length) over Σ. Let λ denote the empty string. For a finite (resp., infinite) string s, the *prefix set* of s, denoted by $pref(s)$, is the set $\{w \in \Sigma^* \mid$ there exists a finite (resp., infinite) string $s', s = ws'\}$.

A *deterministic Rabin automaton* (or simply *Rabin automaton*) M is a 5-tuple $(\Sigma, X, \delta, x_0, F)$, where Σ is a finite set of *event symbols*, X is a finite set of *states*, $\delta : \Sigma \times X \rightarrow X$ (a partial function) defines the *transition function*, $x_0 (\in X)$ is the *initial state*, and $F = \{(R_1, P_1), ..., (R_n, P_n) \mid R_i, P_i \subseteq X, 1 \leq i \leq n\}$, for some n, is a set of pairs of state subsets specifying the *acceptance condition*. As assumed in [5], distinct transitions of M carry distinct event symbols. The intuitive meaning of transition $\delta(\sigma, x) = x'$ is that in state x, M can go to the next state x' by generating the event symbol σ. For

case of expression, we extend δ to $\Sigma^* \times X \rightarrow X$ recursively as follows: $\delta(\lambda, x) = x; \delta(w\sigma, x) = \delta(\sigma, \delta(w, x))$ (where $w \in \Sigma^*$ and $\sigma \in \Sigma$), provided that $\delta(w, x)$ is defined. We write $x \xrightarrow{w} x'$ if $\delta(w, x) = x'$, where $x, x' \in X$ and $w \in \Sigma^*$. A finite string s is said to be *generated* by M if $\delta(s, x_0)$ is defined. M *generates* an infinite string s if $\forall w \in pref(s), \delta(w, x_0)$ is defined. Given a finite or infinite sequence $s = \sigma_1\sigma_2\cdots\sigma_i\cdots$ generated by M, it should be noted that the sequence of states $x_1, x_2, \cdots, x_i, \cdots$ witnessing $x_0 \xrightarrow{\sigma_1} x_1 \xrightarrow{\sigma_2} x_2 \cdots x_{i-1} \xrightarrow{\sigma_i} x_i \cdots$ is unique, for the underlying automaton is deterministic. Throughout the rest of this paper, we write $st(s)$ to denote such a sequence. We define $\Omega_s = \{x \mid x\ (\in X)$ appears infinitely often in $st(s)\}$. A string $s \in \Sigma^\omega$ is *accepted* by M iff $\exists(R_i, P_i) \in F, (\Omega_s \cap R_i \neq \emptyset) \wedge (\Omega_s \subseteq P_i)$.

Given a $\mathcal{C} \subseteq 2^\Sigma$, a *controller* is a function $f : \Sigma^* \rightarrow \mathcal{C}$ such that $f(w)$ specifies the set of symbols among which the next symbol comes from, if w is the sequence that has been generated thus far. (That is, with respect to a given a history, f places a restriction on the next eligible symbol.) Under the influence of a controller, the notion of *generation* is refined as follows. A finite string $s \in \Sigma^*$ is *generated* by M *under* f iff $\delta(s, x_0)$ is defined, and for every $w\sigma \in pref(s)$, $\sigma \in f(w)$. Likewise, an infinite string $s = \sigma_1\sigma_2\cdots(\in \Sigma^\omega)$ is said to be *generated* by M *under* controller f if there exists an infinite sequence of states $x_1, x_2\cdots$ such that (1) $x_0 \xrightarrow{\sigma_1} x_1 \xrightarrow{\sigma_2} x_2 \cdots x_{i-1} \xrightarrow{\sigma_i} x_i \cdots$, (2) $\forall i \geq 1, \sigma_i \in f(\sigma_1 \ldots \sigma_{i-1})$, (3) (*state fairness*) for every $\sigma \in \Sigma$, if there exist infinitely many indices $1 \leq i_1 < i_2 < \cdots < i_j < \cdots$ such that $\forall j \geq 1, \sigma \in f(\sigma_1 \cdots \sigma_{i_j})$, and $\delta(\sigma_1 \cdots \sigma_{i_j}\sigma, x_0)$ is defined, then σ must occur infinitely many times in s. A Γ in $\mathcal{C}$ is said to be *enabled* in a state x if there exists a $\sigma \in \Gamma$ such that $\delta(\sigma, x)$ is defined.

Given a Rabin automaton $M = (\Sigma, X, \delta, x_0, F)$ and a state x, we write M_x to denote a Rabin automaton which is identical to M except that the initial state of M_x is x (i.e., $M_x = (\Sigma, X, \delta, x, F)$). With a given $\mathcal{C} \subseteq 2^\Sigma$, the *controllability subset* contains those states x such that for M_x, there exists a controller f such that

(i) (*liveness condition*) if $w(\in \Sigma^*)$ is generated by M_x under f, then there exists an $s \in \Sigma^\omega$, $w \in pref(s)$ and s is generated by M_x under f (i.e., every generated finite string can be extended to an infinite one (under the same controller)),

(ii) (*safety condition*) if $s(\in \Sigma^\omega)$ is generated by M_x under f, then s is accepted by M_x (i.e., the set of generated infinite strings complies with that of the accepted strings under the Rabin acceptance condition).

Intuitively, a state x is *controllable* if all the computations emanating from x are *live* (i.e., non-blocking) and *safe* (i.e., in compliance with the Rabin acceptance condition). The reader is referred to [5, 6] for more about the above definitions.

In this paper, we investigate the problem of computing the controllability subset with respect to 'fair' controllers. The following three types of fairness notions (stated in [1]) are considered. Given a Rabin automaton $M = (\Sigma, X, \delta. x_0, F)$ and a $\mathcal{C} \subseteq 2^{\Sigma}$, a controller $f : \Sigma^* \to \mathcal{C}$ is said to be

- *impartial* iff for every infinite computation $x_0 \xrightarrow{\sigma_1} x_1 \xrightarrow{\sigma_2} x_2 \cdots x_{i-1} \xrightarrow{\sigma_i} x_i \cdots$ under f, the set $\{\Gamma \in \mathcal{C} \mid$ there exist infinitely many i, $f(\sigma_1 \cdots \sigma_i) = \Gamma\}$ equals $\mathcal{C}$.
- *just* iff for every infinite computation $x_0 \xrightarrow{\sigma_1} x_1 \xrightarrow{\sigma_2} x_2 \cdots x_{i-1} \xrightarrow{\sigma_i} x_i \cdots$ under f, and for every Γ in $\mathcal{C}$, if there exists a $j \geq 1$ such that $\forall i \geq j, \Gamma$ is enabled in x_i, then there must be infinitely many indices $1 \leq i_1 < \cdots < i_k \cdots$, $f(\sigma_1 \cdots \sigma_{i_k}) = \Gamma, \forall k \geq 1$.
- *fair* iff for every infinite computation $x_0 \xrightarrow{\sigma_1} x_1 \xrightarrow{\sigma_2} x_2 \cdots x_{i-1} \xrightarrow{\sigma_i} x_i \cdots$ under f, and for every Γ in $\mathcal{C}$, if there exist infinitely many indices $1 \leq j_1 < \cdots < j_m < \cdots$ such that $\forall m \geq 1, \Gamma$ is enabled in x_{j_m}, then there must be infinitely many $1 \leq i_1 < \cdots < i_k \cdots$, $f(\sigma_1 \cdots \sigma_{i_k}) = \Gamma, \forall k \geq 1$.

The *fair controllability subset problem* is that of, given a Rabin automaton $M = (\Sigma, X, \delta, x_0, F)$ and a set $\mathcal{C} \subseteq 2^{\Sigma}$, finding the controllability subset (with respect to a controller which is *impartial*, *just* or *fair*) of maximum size.

Consider a Rabin automaton $M = (\Sigma, X, \delta, x_0, F)$ and a $\mathcal{C} = \{\Gamma_1, ..., \Gamma_n\}$ $(\subseteq 2^{\Sigma})$. We define the corresponding *execution graph*, denoted by $G_{M,\mathcal{C}}$, as a directed labeled graph (V, E, L), where $V = X$, the edge labeling function $L : E \to 2^{\{1,...,n\}}$ is defined as $L((x, x')) = \{j \mid 1 \leq j \leq n, \exists \sigma \in \Gamma_j, \delta(\sigma, x) = x'\}$. Edge (x, x') is absent if $L((x, x')) = \emptyset$. For convenience, we write $(x, x')_r, r \subseteq \{1, ..., n\}$, to denote that the label of edge (x, x') is r. Given a set of vertices $V' \subseteq V$, a *restricted graph* over V' is a subgraph (V', E', L') such that $\emptyset \neq E' \subseteq E$, $L'(e) \subseteq L(e), \forall e \in E'$, and if an edge $(x, x')_r \in E'$, then there is no $x'' \in V - V'$ such that $(x, x'')_{r'} \in E$ and $r \cap r' \neq \emptyset$. (The last condition will be referred to as the *state fairness* condition.) Consider a restricted graph $G' = (V', E', L')$ over V'. Suppose G' is strongly connected. G' is said to be of type SCC_{imp} iff $\forall 1 \leq i \leq n$, $(\exists\ e \in E', i \in L'(e))$; SCC_{just} iff $\forall 1 \leq i \leq n$, $(\forall x \in V', \exists x' \in V, i \in L((x, x')) \Longrightarrow (\exists\ e \in E', i \in L'(e))$; SCC_{fair} iff $\forall 1 \leq i \leq n$, $(\exists x \in V', \exists x' \in V, i \in L((x, x'))) \Longrightarrow (\exists\ e \in E',$

$i \in L'(e)$). SCC_{imp} (SCC_{just} and SCC_{fair}) G' is said to be *accepting* iff $\exists (R, P) \in F, (V' \cap R \neq \emptyset) \wedge (V' \subseteq P)$. (Intuitively, an SCC is to capture the set of states and transitions that appear infinitely often in an accepting computation.)

Given a vertex x, a subgraph (V', E', L') is said to be a *prefix subgraph rooted at* x if *(a)* $\forall y \in V'$, y is reachable from x, *(b)* $|L'(e)| = 1, \forall e \in E'$, and *(c)* if $(y, y')_{\{i\}} \in E'$, then $\forall z \in V$ with $i \in L(y, z)$, we have $(y, z)_{\{i\}} \in E'$. Intuitively speaking, a prefix subgraph is to capture the control policy prior to entering the set of states that occur infinitely often in the computation.

3. COMPUTING THE FAIR CONTROLLABILITY SUBSET

We are in a position to present a graph-theoretic approach to the fair controllability subset problem. Before doing so, a couple of lemmas are needed.

Lemma 3.1 *Consider a Rabin automaton* $M = (\Sigma, X, \delta, x_0, F)$ *and a* C $(= \{\Gamma_1, ..., \Gamma_n\}) \subseteq 2^{\Sigma}$, *let* $G_{M,C} = (V, E, L)$. *If a state* y *resides in some accepting* SCC_{imp} *(resp.,* SCC_{just} *and* SCC_{fair}*)* (V', E', L'), *then* y *is controllable under impartiality (resp., justice and fairness).*

Proof: Without loss of generality, let $V' = \{q_1, ..., q_k\}$. For each state $q_j \in V'$, we partition V' into a finite set $\{V_0^j, ..., V_{m_j}^j\}$, for some m_j, where $V_0^j = \{q_j\}$, and $V_t^j = \{q_i \mid q_i \in V' - (\cup_{s=0}^{t-1} V_s^j)$ such that $(q_i, q_l)_r \in E'$, for some r and $q_l \in V_{t-1}^j\}$, $1 \leq t \leq m_j$. Since V' is finite, such an m_j must exist. For a state q_j, we design a control policy $S(q_j)$ as follows: if $q_i \in V_t^j$ $(t \geq 1)$ is the current state, choose Γ_h such that for some state $q_l \in V_{t-1}^j$, $(q_i, q_l)_r \in E'$ and $h \in r$. Intuitively, the policy has the tendency of moving the computation toward V_0^j (and hence, state q_j). Now we claim that from an arbitrary state q_i, all the infinite computations under S_j must eventually reach q_j.

Let $A = \{[q, \Gamma_h] \mid 1 \leq h \leq n, q \in V'$, such that $\exists\ (q, q')_r \in E', h \in r\}$. Since A is clearly finite, let $A = \{[q_{i_1}, \Gamma_{h_1}], ..., [q_{i_l}, \Gamma_{h_l}]\}$. Starting from state y, the control policy is as follows:

(1) Use $S(q_{i_1})$ until q_{i_1} is reached; in state q_{i_1}, select Γ_{h_1}; then proceed according to (2).

(2) Use $S(q_{i_2})$ until q_{i_2} is reached; in state q_{i_2}, select Γ_{h_2}; then proceed according to (3). ...

(l) Use $S(q_{i_l})$ until q_{i_l} is reached; in state q_{i_l}, select Γ_{h_l}; then proceed according to (1).

Based on our earlier claim, in step $(j), 1 \leq j \leq l$, the computation must eventually reach state q_{i_j} in any generated infinite computation. Hence, our round-robin control policy is *impartial* (resp., *just* and *fair*), since (V', E', L') is an SCC_{imp} (resp., SCC_{just} and SCC_{fair}). ∎

Lemma 3.2 *Given a Rabin automaton $M = (\Sigma, X, \delta, x_0, F)$ and a $\mathcal{C} \subseteq 2^{\Sigma}$, a state x is controllable with respect to impartiality (resp., justice and fairness) iff for $G_{M,\mathcal{C}} = (V, E, L)$, there exists a prefix subgraph (V', E', L') rooted at x such that for each $y \in V'$, if the out-degree of y is zero, then y is in some accepting SCC_{imp} (resp., SCC_{just} and SCC_{fair}).*

Proof: The *if part* (i.e., $\Longleftarrow$) follows immediately from the definition of 'fair controllable states' and Lemma 3.1. Now we prove the *only if part* (i.e., $\Longrightarrow$). Suppose $\pi : x \xrightarrow{\sigma_1} x_1 \xrightarrow{\sigma_2} x_2 \cdots x_{i-1} \xrightarrow{\sigma_i} x_i \cdots$ constitutes an accepting computation under some *impartial* (resp., *just* and *fair*) controller f, where $x_i \in X$ and $\sigma_i \in \Sigma$, $\forall i \geq 1$. Let $j \geq 1$ be the smallest index such that for all $i \geq j$, x_i appears infinitely many times in π. We define the following subgraph $G' = (V', E', L')$ of (V, E, L): $V' = \{x_i \mid i \geq j\}$, $E' = \{(x_i, x_{i+1}) \mid i \geq j\}$, and $l \in L'((x', x''))$ if $f(\sigma_1 \cdots \sigma_m) = l$ for some $m \geq j$ and $x_m = x', x_{m+1} = x''$. Clearly, G' is strongly connected and is a restricted graph. Now we show that G' is an accepting SCC_{imp} (resp., SCC_{just} and SCC_{fair}).

1 (*impartial*): As f is impartial, for each $\Gamma_d, 1 \leq d \leq n$, there exists a transition (x_m, x_{m+1}) in π such that $f(\sigma_1 \cdots \sigma_m) = d$. Hence, d occurs in the label of an edge in G' (definition of G').

2 (*just*): The notion of *justice* requires that for each Γ_d, if some symbol in Γ_d is enabled continuously after some point in time on π, then d must be selected infinitely many times by f. Hence, the condition $\forall 1 \leq i \leq n$, $(\forall x \in V', \exists x' \in V, i \in L((x, x')) \Longrightarrow (\exists e \in E', i \in L'(e))$ in the definition of SCC_{just} holds for G'.

3 (*fair*): Since f is *fair* (i.e., for each Γ_d, if some symbol in Γ_d is enabled infinitely often on π, then d must be selected infinitely many times by f), the condition $\forall 1 \leq i \leq n$, $(\exists x \in V', \exists x' \in V, i \in L((x, x'))) \Longrightarrow (\exists\ e \in E', i \in L'(e))$ in the definition of SCC_{fair} holds for G'.

Since π is accepting, so is G'. ∎

Theorem 1 *The fair controllability subset problem with respect to 'impartiality' ('justice' and 'fairness') is solvable in polynomial time.*

Proof: Consider a Rabin automaton $M = (\Sigma, X, \delta, x_0, F)$ and a $\mathcal{C} \subseteq 2^{\Sigma}$. Suppose F=$\{(R_1, P_1), ..., (R_h, P_h)\}$, for some h. We use the following algorithms to find **all** the maximal accepting SCC_{imp} (resp. SCC_{just} and SCC_{fair}).

Algorithm Find-max-Rabin-SCC
input: $G = (\bar{V}, \bar{E}, \bar{L})$
output: $S = \{Q_1, ..., Q_u\}$, containing all the maximal SCCs of G such that each Q_i satisfies the Rabin acceptance condition.
(1) $S = \emptyset$.
(2) **for** each $1 \leq j \leq h$ **do**
Let $G_j = (P_j, E_j, L_j)$ be the induced subgraph w.r.t. P_j.
Find the maximal SCCs, say, $Q_1, ..., Q_p$ (for some p) of G_j, each containing some vertex in R_j. $S \leftarrow S \cup \{Q_1, ..., Q_p\}$
end algorithm

Algorithm *Find-max-Rabin-SCC* is to partition an input graph into a set of maximal SCCs, each of which complies with the Rabin acceptance condition.

Algorithm Find-max-fair-SCC
input: $G_{M,\mathcal{C}} = (V, E, L)$
output: $S = \{G_1, ..., G_m\}$, containing all the max. accepting SCC_{imp} (resp. SCC_{just} and SCC_{fair}) of $G_{M,\mathcal{C}}$
(1) $S = \emptyset$.
(2) Let $H =$ *Find-max-Rabin-SCC*$(G_{M,\mathcal{C}})$.
(3) **If** $H = \emptyset$ then *return*.
(4) Choose some $Q \in H$; $H \leftarrow H - \{Q\}$
(5) **if** $Q = (V', E', L')$ is an SCC_{imp} (resp. SCC_{just} and SCC_{fair}) **then** $S = S \cup \{Q\}$; **goto** (3) **else**
begin
(i) **if** Q violates the *state fairness* condition **then** find nodes x, x' $\in V'$, $x'' \notin V'$ and a Γ_i with $i \in L'((x, x'))$ and $i \in L((x, x''))$. Then let Q' be the subgraph of Q resulting from changing edges $(x, y)_r, i \in r$, to $(x, y)_{r-\{i\}}$. If Q'' is strongly connected, then let $H \leftarrow H \cup \{Q'\}$ and goto (3); otherwise,
$H \leftarrow H \cup$ *Find-max-Rabin-SCC*(Q') and goto (3).
(ii) **if** Q is not of type
(ii-1) SCC_{imp}: then goto (3)
(ii-2) SCC_{just}: then goto (3)

(ii-3) SCC_{fair}: then find an $x \in V'$, $x' \notin V'$, and Γ_i such that $i \in L((x,x'))$ but no edge in E' has i in its label set.
Let Q' be Q's subgraph by removing x and its adjacent edges from Q. Let $H \leftarrow H \cup$ *Find-max-Rabin-SCC*(Q') and goto (3).
end

end algorithm

Let $\mathcal{S} = \{x \mid x$ is a node in some $G_i, 1 \leq i \leq m\}$, where $\{G_1, ..., G_m\}$ is the output of algorithm *Find-max-fair-SCC*. In what follows, we design an $O(log\ n)$ space bounded *alternating Turing machine* (*ATM*, for short) U to decide whether a state x is in the fair controllability subset. We let $\mathcal{S}$ be the set of accepting states of U. A *state* of the ATM is of the form $(y, \exists)$ or $(y, i, \forall)$, where $y \in V$ and i is the index of some Γ_i. Starting from the initial state $(x, \exists)$, a computation of the ATM U consists of two phases: ($\exists$ phase) In an $\exists$-state $(y, \exists)$, U nondeterministically chooses a Γ_i and enter a $\forall$-state $(y, i, \forall)$; ($\forall$ phase) In a $\forall$-state $(y, i, \forall)$, ATM U branches to all the $\exists$-states $(z, \exists)$, provided that $i \in L((y,z))$. As $O(log\ n)$ space bounded ATMs capture exactly the complexity class PTIME, the theorem follows. ∎

As for the lower bound, we have:

Theorem 2 *The fair controllability problem is PTIME-hard.*

References

[1] Hart, S., Sharir, M. and Pnueli, A., (1983). Termination of probabilistic concurrent programs, *ACM Trans. on Programming Languages and Systems*, 5: 356-380.

[2] Jones, N. and Laaser, W., (1977). Complete problems for deterministic polynomial time, *Theoret. Comput. Sci.*, 3: 105-117.

[3] Ramadge, P. and Wonham, W., (1989). The control of discrete event systems, *Proc. IEEE*, 77(1): 81-98.

[4] Thistle, J., (1995). On control of systems modeled as deterministic Rabin automata, *Discrete Event Dynamic Systems: Theory and Applications*, 5(4): 357-381.

[5] Thistle, J. and Malhamé, R., (1998). Control of ω-automata under state fairness assumptions, *Systems and Control Letters*, 33: 265-274.

[6] Thistle, J. and Wonham, W., (1994). Control of infinite behavior of finite automata, *SIAM J. Control and Optimization*, 32(4): 1075-1097.

Analysis of Self-Stabilization for Infinite-State Systems

Hsu-Chun Yen
Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan 106, R.O.C.
yen@cc.ee.ntu.edu.tw

Abstract

The problem of deciding whether an infinite-state system is self-stabilizing *or not is investigated from the decidability viewpoint in this paper. We develop a unified strategy through which checking self-stabilization is shown to be decidable for* one-counter machines *and* conflict-free Petri nets. *Our strategy relies on the reachability sets being semilinear, as well as on the capability of extracting periodic behaviors of infinite computations, which, in turn, facilitates the expression of self-stabilization by Presburger Arithmetic.*

As fairness *is frequently used as a qualitative measure to capture the notion of a quantitative measure of 'something happens with probability one,' it is of interest to examine the* fair *version of the self-stabilization problem, i.e., the problem of asking whether all 'fair' infinite computations eventually become self-stabilizing. We propose a* potential method *through which the problem is shown to be decidable for conflict-free Petri nets.*

Keywords: Decidability, infinite-state system, Petri net, self-stabilization.

1 Introduction

The notion of *self-stabilization* was introduced by Dijkstra [3] to describe a system having the behavior that regardless of its starting configuration, the system would return to a 'legitimate' configuration eventually. (By a legitimate configuration, we mean a configuration which is reachable from the initial configuration of the system.) The motivation behind self-stabilization is that a self-stabilizing system has the ability to 'correct' itself even in the presence of certain unpredictable errors that force the system to reach an 'illegitimate' configuration during the course of its operations. In this sense, self-stabilizing systems exhibit fault-tolerant behaviors to a certain degree. With the increased interest in designing fault-tolerant systems, the study of self-stabilization has been gaining increasing popularity in the computer science community lately. In what follows, we briefly review some of the previous work done in the area of self-stabilization.

Following the seminal work of Dijkstra [3], the majority of research along the line of self-stabilization has been focusing on providing solutions (and their proofs) for self-stabilizing systems with a variety of properties and topologies. (See [6] for a bibliography of self-stabilization.) In a somewhat different direction of research, Gouda, Howell and Rosier ([5]) have showed that the property of self-stabilization is 'unstable' across a wide variety of system classes, ranging from cellular automata, Turing machines, communicating finite state machines, to Petri nets. They basically demonstrated that the *simulation paradigm*, which is a useful tool for designing and analyzing systems, may not be 'robust' with respect to the property of self-stabilization. (The simulation paradigm, simply speaking, is a methodology routinely used for designing or analyzing one class of systems with the help of another (hopefully, a well-studied one) through the simulation of the former by the latter. In this manner, properties of the former can be deduced from the respective properties of the latter.) Unlike traditional properties such as liveness, boundedness and fairness which are almost always preserved under standard simulations, self-stabilization could easily be lost through the process of simulation from one class of systems to another. (For more details, see [5].) To our knowledge, [5] also pioneers the introduction of the notion of self-stabilization to the model of Petri nets. In a subsequent paper [1], specific efforts have been devoted to reasoning about self-stabilization aspects of Petri nets from the computational complexity point of view. Among those complexity results derived in [1], detecting whether a Petri net is self-stabilizing is complete for polynomial time for bounded ordinary Petri nets, whereas it is PSPACE-complete for bounded general Petri nets.

As far as we know, very little is known regarding the decidability/complexity issue of the self-stabilization problem (i.e., the problem of deciding whether a system is self-

stabilizing or not) for infinite-state systems. At this moment, the best bounds of the problem for general Petri nets are a lower bound of exponential space and an upper bound of Π_2 (the second level of the arithmetic hierarchy), whereas it is Π_2-complete for Turing machines [1]. Therefore, it is of interest and importance to take a closer look at the problem for other infinite-state systems, in the hope of recognizing the key characteristics which govern the decidability/undecidability nature of the problem. An equally important goal is to, perhaps, come up with a unified framework through which decidability/undecidability of self-stabilization can be obtained. In this paper, we extend the work of [1] by investigating, from the decidability viewpoint, the problem of deciding whether a given system is self-stabilizing for *one-counter machines* and *conflict-free Petri nets*. As it turns out, the decidability of self-stabilization for these two computational models can be established in a unified setting, taking advantage of the following key properties:

1. The reachability set as well as the set of 'dead' configurations are effectively semilinear,

2. The existence of a *periodic* witness for non-self-stabilizing infinite computations.

The above properties allow us to formulate self-stabilization in terms of formulas in Presburger Arithmetic, giving rise to the decidability result.

As 'fairness' is frequently used as a qualitative measure to capture the notion of a quantitative measure of 'something happens with probability one,' it is of interest to re-examine the self-stabilization problem by asking whether all 'fair' infinite computations eventually enter legitimate states. We propose a 'potential method' through which the problem is shown to be decidable for a restricted class of Petri nets. The idea is to associate a value (called 'potential') with each marking in the Petri net in such a way that the potential along any Petri net computation is non-increasing, and in some cases, has the tendency to move towards the ground level (i.e., potential zero). To a certain degree, the basic idea of our potential method is analogous to the analysis of *neural nets* (in particular, *Hopfield nets*), in which an *energy function* is associated with a Hopfield net, and the energy of the net decreases as the computation proceeds (provided that the net meets certain conditions) which, in turn, guarantees the convergence of the net.

The rest of this paper is organized as follows. In Section 2, we give the definitions of transition systems and self-stabilization. Section 3 deals with the issue of deciding whether a given system is self-stabilized or not from the decidability viewpoint for infinite-state systems. A fair version of self-stabilization is investigated in Section 4. A conclusion is given in Section 5.

2 Preliminaries

Let $\mathbf{Z}$ ($\mathbf{N}$) denote the set of (nonnegative) integers, and $\mathbf{Z}^k$ ($\mathbf{N}^k$) the set of vectors of k (nonnegative) integers.

A *transition system* (or *system*, for short) S is a four-tuple (X, Σ, δ, c_0), where X consists of a (possibly infinite) set of *configurations*, Σ is a (possibly infinite) set of *transition symbols*, $\delta : X \times \Sigma \rightarrow 2^X$ defines the *transition function*, and $c_0 (\in X)$ is the *initial configuration*. We write $c \xrightarrow{t} c'$ if $c' \in \delta(c, t)$. Also let $\xrightarrow{*}$ (resp., $\xrightarrow{+}$) be the reflexive and transitive closure (resp., transitive closure) of $\rightarrow$. In words, $c \xrightarrow{*} c'$ (resp., $c \xrightarrow{+} c'$) iff c' can be reached from c through zero (resp., one) or more transitions. A *computation* σ is a (finite or infinite) sequence $c_1 \xrightarrow{t_1} c_2 \xrightarrow{t_2} \cdots c_i \xrightarrow{t_i} c_{i+1} \cdots$. A configuration c is said to be a *dead configuration* if $\forall t \in \Sigma, \delta(c, t) = \emptyset$, i.e., c has no immediate successor in S. In this paper, we consider only those systems for which X can be encoded in such a way that $X \subseteq Q \times \mathbf{N}^k$, for some finite set Q and integer $k \geq 1$. (Q is referred to as the set of *states* of system S.)

Intuitively speaking, a system is said to be *self-stabilizing* (*ss*, for short) if, regardless of its starting configuration, the system would eventually return to a 'legitimate' configuration which is reachable from the *initial configuration* of the system. That is, a self-stabilizing system has the ability to 'correct' itself even in the presence of certain unpredictable errors that force the system to reach an 'illegitimate' configuration during the course of its operations. Let S be a (finite or infinite) *system* with c_0 as its *initial configuration*. Also let $R(S, c_0)$ ($=\{c \mid c_0 \xrightarrow{*} c\}$) denote the set of *reachable configurations* from c_0. A computation σ from configuration c_1 is said to be *non-self-stabilizing* (*non-ss*) iff one of the following holds:

1. σ is finite ($\sigma : c_1 \xrightarrow{t_1} c_2 \xrightarrow{t_2} \cdots c_{m-1} \xrightarrow{t_{m-1}} c_m$, for some m) such that c_m is a dead configuration and $c_m \notin R(S, c_0)$, or

2. σ is infinite ($\sigma : c_1 \xrightarrow{t_1} c_2 \xrightarrow{t_2} \cdots c_i \xrightarrow{t_i} c_{i+1} \cdots$) such that $\forall i \geq 1, c_i \notin R(S, c_0)$.

See Figure 1. A system is said to be *self-stabilizing* if for each configuration c, none of the computations emanating from c is *non-ss*. The *self-stabilization problem* is to determine, for a given (finite or infinite) system, whether the system is self-stabilizing. In this paper, our main concern is to investigate the decidability issue of the self-stabilization problem for a variety of infinite-state systems. The formal definitions of *configurations* and *computations* of each individual system will be clarified as our discussion progresses.

For a vector $v \in \mathbf{N}^k$ and a finite set $\rho (= \{v_1, ..., v_n\}$, for some $n) \subseteq \mathbf{N}^k$, the set $\mathcal{L}(v; \rho) = \{v \mid \exists a_1, ..., a_n \in \mathbf{N}, v = v + \sum_{i=1}^{n} a_i * v_i\}$ is called the *linear set* with *base* v over the

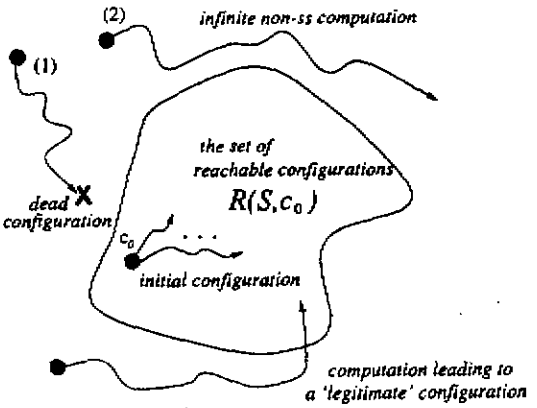

Figure 1. Non-self-stabilizing computations.

set of *periods* ρ. A *semilinear set* (SLS) (cf. [4]) is a finite union of linear sets. It is known that semilinear sets are exactly those that can be expressed in *Presburger Arithmetic* [10], which is a decidable theory.

3 Decidability analysis

In this section, the *ss* problem for *one-counter machines* and *conflict-free Petri nets* will be shown to be decidable. Before going into the details, we begin by giving an overview of the general strategy using which the decidability results are obtained.

Recall from the definition of *ss* that a system $\mathcal{S}$ is not self-stabilizing iff either (i) a dead configuration is not reachable from the initial configuration, or (ii) an infinite non-*ss* computation exists. As we shall see later, (i) is relatively easy to check as long as certain properties of $\mathcal{S}$ are decidable. The idea behind our approach of checking (ii) is built upon showing that to demonstrate the non-*ss* nature of a system, it is sufficient to search among only non-*ss* computations of *periodic* behaviors, and hopefully, the confinement to such computations admits a decidable checking of (ii). By non-*ss* periodic computations we mean those non-*ss* computations of the form $s \xrightarrow{\tau\tau\cdots}$, i.e., repeating τ from s infinitely many times witnesses non-*ss*. In our subsequent investigation, we characterize the following two types of non-*ss* periodic behaviors, through which a unified strategy is developed serving as a foundation for our decidability results of the *ss* problem.

- (*Strongly periodic:*) *Every* finite computation $c \xrightarrow{\tau} c'$ with $c' \geq c$ and $c \notin R(\mathcal{S}, c_0)$ ensures non-*ss* of its infinite repetition (i.e., $c \xrightarrow{\tau\tau\cdots}$ is non-*ss*).
- (*Weakly periodic:*) If an infinite non-*ss* computation exists, there must exist a witness τ which is in a computable finite set, and $c \xrightarrow{\tau\tau\cdots}$ constitutes an infinite non-*ss* computation.

The disparity between strongly and weakly periodic non-*ss* computations is that in the former, a finite computation of the form '$c \xrightarrow{\tau} c'$ with $c' \geq c$ and $c \notin R(\mathcal{S}, c_0)$' is guaranteed to conclude the system's non-*ss* behavior, whereas in the latter, however, not *every* finite computation of the above form is extendible to render *ss*; nevertheless, there does *exist* such a witness which is computable. (The meaning of '$\geq$' will become clear as our discussion progresses.)

Before going into the details, we require the following notations. Let $\mathcal{S}=(X, \Sigma, \delta, c_0)$ be a transition system, where $X \subseteq Q \times \mathbf{N}^k$, $c \in X$, and $q, q' \in Q$.

- $SUCC(\mathcal{S}, q, q') = \{(s, s') \mid (q, s) \xrightarrow{t} (q', s'),$ for some transition $t \in \Sigma\}$.
- $REACH(\mathcal{S}, q, q') = \{(s, s') \mid (q, s) \xrightarrow{*} (q', s')\}$;
  $REACH^+(\mathcal{S}, q, q') = \{(s, s') \mid (q, s) \xrightarrow{+} (q', s')\}$;
  $Reach(\mathcal{S}, c, q') = \{s' \mid c \xrightarrow{*} (q', s')\}$,
  $R(\mathcal{S}, c) = \{c' \mid c \xrightarrow{*} c'\}$.
- $DEAD(\mathcal{S}, q) = \{s \mid (q, s)$ is a dead configuration$\}$.

As we shall see later, decidability of the *ss* problem for each of the aforementioned computational models is based upon the following unified strategy:

Theorem 3.1 *Let $\mathcal{X}$ be a computational model for which the ss problem is considered. If for each system $\mathcal{S} = (X, \Sigma, \delta, c_0)$ in $\mathcal{X}$, where $X \subseteq Q \times \mathbf{N}^k$, we have*

(1) (Semilinear condition) *$\forall q, q' \in Q$, the sets $SUCC(\mathcal{S}, q, q')$, $REACH(\mathcal{S}, q, q')$ and $DEAD(\mathcal{S}, q)$ are effectively semilinear, and*

(2) (Periodic condition) *$\mathcal{S}$ has an infinite non-ss computation iff there exists an infinite computation $(q, s) \xrightarrow{+} (q, s + d) \xrightarrow{+} (q, s + 2d) \cdots$, for some $(q, s) \in X$ and $d \in \mathbf{N}^k$, such that either (a) or (b) below holds:*

- *(a)*
  - *$s \notin Reach(\mathcal{S}, c_0, q)$,*
  - *$(s \notin Reach(\mathcal{S}, c_0, q))$ implies $(\forall n \geq 0, s + n * d \notin Reach(\mathcal{S}, c_0, q))$, and*
  - *$((q, s) \xrightarrow{+} (q, s+d))$ implies $(\forall n \geq 0, (q, s+ n * d) \xrightarrow{+} (q, s + (n + 1)d))$.*
- *(b)*
  - *$\forall n \geq 0, s + n * d \notin Reach(\mathcal{S}, c_0, q)$, and*
  - *d belongs to a computable finite set $D \subseteq \mathbf{N}^k$.*

then the ss problem is decidable for $\mathcal{X}$. (Notice that types (a) and (b) characterize strongly and weakly periodic non-ss computations, respectively.)

Proof: First notice that if $\forall q, q' \in Q$, $REACH(S, q, q')$ is semilinear, so is $Reach(S, c, q')$, $\forall c \in X$ and $q' \in Q$. According to the definition of ss, system S is not ss iff either (i) a dead configuration is not reachable, or (ii) an infinite non-ss computation exists. The (i) case can be checked by testing $(\exists q \in Q)\ (DEAD(S, q) \cap \overline{Reach(S, c_0, q)} \neq \emptyset)$, which is decidable because of the semilinearity assumption of (1). To check (ii), assumption (2) suggests that it is sufficient to look for an infinite non-ss computation of type (a) or (b). Deciding the existence of such a path can be formulated as follows.

- Type (a) (i.e., *strongly periodic case*):
$\forall_{q \in Q}\ (\exists\ s \in \mathbf{N}^k)\ (\exists\ d \in \mathbf{N}^k)$ $((s, s+d) \in REACH^+(S, q, q) \wedge (s \notin Reach(S, c_0, q)))$, which can be expressed in Presburger Arithmetic (due to assumption (1)).
- Type (b) (i.e., *weakly periodic case*):
$\forall_{q \in Q}\ \forall_{d \in D}\ (\exists\ s \in \mathbf{N}^k)\ (\forall\ n \geq 0)$ $((s + n * d, s + (n+1) * d) \in REACH^+(S, q, q))$ $\wedge\ (s + n * d \notin Reach(S, c_0, q))$, which can be expressed in Presburger Arithmetic (due to assumption (1) and the fact that $(s, s') \in REACH^+(S, q, q')$ iff $(\exists s'' \in \mathbf{N}^k)\ (\exists q'' \in Q)\ ((s, s'') \in SUCC(S, q, q'')) \wedge ((s'', s') \in REACH(S, q'', q'))$. ∎

We are in a position to show why each of the models of *one-counter machines* and *conflict-free Petri nets* falls into a special case of Theorem 3.1.

3.1 One-counter machines

A *one-counter machine* M consists of a finite-state control equipped with a *counter* on which three types of operations, namely, *addition*, *subtraction* and *test-for-zero* can be performed. For convenience, we use the following to denote the above three basic operations:

(1) (q, q', n), $n \geq 0$ (resp., $n < 0$), meaning that M can go from state q to state q' by adding n to the counter (resp., by subtracting $|n|$ from the counter, provided that the value of the counter is greater than or equal to $|n|$)

(2) $(q, q', = 0)$, meaning that M can go from state q to state q' provided that the value of the counter is zero.

Given a one-counter machine M, we denote by M_{state} and M_{tr} the sets of states and transitions of M, respectively. A *configuration* of one-counter machine M is a pair (q, n), where $q \in M_{state}$, and $n \in \mathbf{N}$ (indicating the value of the counter). Suppose q_0 is the *initial state* of M, the configuration $(q_0, 0)$ is called the *initial configuration* of M. Given two configurations (q, m), (q', m'), and a transition t, we write $(q, m) \xrightarrow{t} (q', m')$ iff one of the following holds:

1. $t = (q, q', n)$, $n \in Z$ and $m' = m + n$,
2. $t = (q, q', = 0)$ and $m' = m = 0$.

A *short positive loop* in M is a sequence of transitions $(q_1, q_2, n_1)(q_2, q_3, n_2) \cdots (q_m, q_1, n_m)$, for some $m \geq 1$ such that for all $1 \leq i \neq j \leq m$, $q_i \neq q_j$, and $(\sum_{i=1}^{m} n_i) > 0$. (In words, a short positive loop is a cycle (without test-for-zero transitions) in M's transition diagram along which no state repeats itself except the first and the last, and the counter has a positive gain in value along the cycle.)

Before proceeding, we require the following result which is a direct consequence of a similar one in [11].

Lemma 3.2 *Let M be a one-counter machine of size z (when a binary encoding scheme is used). If $(q, m) \xrightarrow{*} (q', m')$, then there exists a computation $\tau : (q, m) \xrightarrow{s} (q', m')$, for some transition sequence s, along which every configuration (p, h) in τ has $h \leq 3 * (z2^z)^3 + max\{m, m'\}$.*

Proof: In Lemma 4.3 of [11], it was shown that for one-counter machines in which the counter can only be incremented or decremented by 1, the reachability of (q', m') from (q, m) is witnessed by a short computation along which the counter never exceeds $3 * (n)^3 + max\{m, m'\}$, where n is the number of transitions of the machine. In our one-counter machine model, up to 2^z can be added to or subtracted from the counter in a single transition. Hence, our one-counter machines of size z can easily be simulated by those defined in [11] using at most $z2^z$ transitions. The lemma follows immediately. ∎

Lemma 3.3 *Given a one-counter machine M and two states q and q', $REACH(M, q, q')$ is effectively semilinear.*

Proof: $REACH(M, q, q')$ is the union of the following two sets:

- $R^1(M, q, q')$=$\{(c, d) | (q, c) \xrightarrow{r} (q', d)$ and sequence r does not use any test-for-zero moves$\}$. As one-counter machines without test-for-zero moves are computationally equivalent to 1-dimensional VASSs, $R^1(M, q, q')$ can be characterized as the reachability set of a 2-dimensional VASS in the following way. We construct a 2-dimensional VASS $\mathcal{P} = (v_0, A, p_0, Q, \delta)$ with $v_0 = (0, 0)$ to simulate the computation of M in such a way that $(c, d) \in R^1(M, q, q')$ is witnessed by the following computation of $\mathcal{P}$: $(p_0, (0, 0)) \xrightarrow{\overbrace{t \cdots t}^{c}} (p_0, (c, c)) \rightarrow (q, (c, c)) \xrightarrow{\sigma} (q', (c, d))$, where repeating transition $t : p_0 \rightarrow (p_0, (1, 1))$ c times is to initialize the counter value, and in the course of σ, $\mathcal{P}$ simulates

t is enabled at μ. We then write $\mu \xrightarrow{t} \mu'$, where $\mu'(p) = \mu(p) - \varphi(p,t) + \varphi(t,p)$ for all $p \in P$. A sequence of transitions $\sigma = t_1 \ldots t_n$ is a *firing sequence* from μ iff $\mu \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \mu_n$ for some sequence of markings $\mu_1, \ldots, \mu_n$.

For ease of expression, the following notations will be used throughout the rest of this paper. Let σ, σ' be transition sequences, and t be a transition.

- $\#_\sigma(t)$ represents the number of occurrences of t in σ.
- $Tr(\sigma) = \{t | t \in T, \#_\sigma(t) > 0\}$, denoting the set of transitions used in σ.
- $\sigma \div \sigma'$ is defined inductively as follows. Suppose $\sigma' = t_1 \ldots t_n$. Let σ_0 be σ. If t_i is in σ_{i-1}, let σ_i be σ_{i-1} with the leftmost occurrence of t_i deleted; otherwise, let $\sigma_i = \sigma_{i-1}$. Finally, let $\sigma \div \sigma' = \sigma_n$. For example, if $\sigma = t_1 t_2 t_3 t_4 t_5$ and $\sigma' = t_4 t_3 t_1$, then $\sigma \div \sigma' = t_2 t_5$.

For firing sequences σ and σ', σ' is said to be a *rearrangement* of σ if $\#_\sigma(t) = \#_{\sigma'}(t), \forall t \in T$.

To simplify our notations, $REACH(\mathcal{P}, q, q')$, $REACH^+(\mathcal{P}, q, q')$, $Rreach(\mathcal{P}, q, q')$, $SUCC(\mathcal{P}, q, q')$ and $DEAD(\mathcal{P}, q)$ are abbreviated as $REACH(\mathcal{P})$, $REACH^+(\mathcal{P})$, $Reach(\mathcal{P})$, $SUCC(\mathcal{P})$, and $DEAD(\mathcal{P})$, respectively, since configurations (i.e., markings) of Petri nets belong to the set $\mathbf{N}^k$.

- *Conflict-free Petri nets*:
  A PN $\mathcal{P}$ =(P, T, φ) is said to be *conflict-free* iff for every place p, either
  1. $|p^\bullet| \leq 1$, or
  2. $\forall t \in p^\bullet$, t and p are on a self-loop (i.e., $t \in (p^\bullet \cap {}^\bullet p)$),

  where $p^\bullet = \{t \mid \varphi(p,t) > 0\}$ (resp., ${}^\bullet p = \{t \mid \varphi(t,p) > 0\}$) represents the set of output (resp., input) transitions of place p. In words, a PN is conflict-free if every place which is an input of more than one transition is also an output of each such transition [9]. In a conflict-free PN, once a transition becomes enabled, the only way to disable the transition is to fire the transition itself. (That is, $\forall t, t' \in T, t \neq t'$, $\mu \xrightarrow{t} \mu'$ and $\mu \xrightarrow{t'}$ implies $\mu' \xrightarrow{t'}$.)

To show the *ss* problem for conflict-free PNs to be decidable, we require the following lemmas.

Lemma 3.6 *Given a conflict-free PN $\mathcal{P}=(P,T,\varphi)$, we can construct in nondeterministic polynomial time a system of linear inequalities $\mathcal{L}(\mathcal{P}, \mu_0, \mu)$ in such a way that $\mu \in R(\mathcal{P}, \mu_0)$ iff $\mathcal{L}(\mathcal{P}, \mu_0, \mu)$ has an integer solution. Furthermore, $\mathcal{L}(\mathcal{P}, \mu_0, \mu)$ remains linear even if μ_0 and μ are replaced by variables. (The reader is referred to Lemma 4.3 in [8] for a detailed description of the system of linear inequalities associated with $\mathcal{L}(\mathcal{P}, \mu_0, \mu)$.)*

Proof: (Sketch) The semilinearity of the reachability set for conflict-free PNs was originally illustrated in [9]. Subsequently, it has been shown in [8] that, given a conflict-free PN $\mathcal{P} = (P, T, \varphi)$, reachability can be characterized as a system of linear inequalities $ILP(\mathcal{P})$ over variables $v_1, \ldots, v_{|P|}, v'_1, \ldots, v'_{|P|}, x_1, \ldots, x_{|T|}$, among others, such that

(1) the size of $ILP(\mathcal{P})$ is bounded by a polynomial in the size of $\mathcal{P}$,

(2) given two markings μ and μ', $\mu \xrightarrow{\sigma} \mu'$, for some sequence σ, *if and only if* $ILP(\mathcal{P})$ has an integer solution while assigning μ to $(v_1, \ldots, v_{|P|})$ and μ' to $(v'_1, \ldots, v'_{|P|})$, and

(3) for each transition $t \in T$, the solutiuon of variable x_t in $ILP(\mathcal{P})$ exactly equals the number of times transition t fires in σ.

∎

As a result, $REACH(\mathcal{P})$ is clearly effectively semilinear.

Lemma 3.7 *Given a PN $\mathcal{P}=(P,T,\varphi)$, the set $DEAD(\mathcal{P})$ is effectively semilinear. (Here $\mathcal{P}$ can be a general Petri net (not necessarily conflict-free) for which the range of φ is* **N**.*)*

Proof: Let $T = \{t_1, t_2, \cdots, t_m\}$. We define $\mathcal{H} = \{\{p_1, p_2, \ldots, p_h\} \subseteq P \mid \forall 1 \leq i \leq m, \exists 1 \leq j \leq h, \varphi(p_j, t_i) > 0\}$ (For each $H \in \mathcal{H}$ and $t_i \in T$, t_i has at least one input place in H.) $\mathcal{H}$ is clearly a finite set. We also let $\mu_H = \{\mu \in \mathbf{N}^k \mid \forall t_i \in T, \exists p_j \in H, \varphi(p_j, t_i) > \mu(p_j)$ and $\forall p' \notin H, \mu(p') = 0\}$, for a given $H \in \mathcal{H}$. Notice that μ_H is finite, which contains dead markings only. If μ' is a marking such that $(\mu'(p_j) = \mu(p_j), \forall 1 \leq j \leq h)$, and $(\mu'(p') \geq \mu(p'), \forall p' \notin H)$, then μ' is a dead marking as well. Hence the set of dead markings equals $\bigcup_{H \in \mathcal{H}} \bigcup_{\mu \in \mu_H} \{\mu' \mid (\mu' \geq \mu) \wedge (\mu'(p) = \mu(p), \forall p \in H)\}$, which is semilinear since $\mathcal{H}$ and μ_H range over finite sets. ∎

Lemma 3.8 *(From [12]) For an arbitrary path $\mu \xrightarrow{\sigma} \overline{\mu}$ in a conflict-free PN $\mathcal{P}=(P,T,\varphi)$, σ can be rearranged into $\overbrace{\sigma_1 \cdots \sigma_1}^{l_1} \overbrace{\sigma_2 \cdots \sigma_2}^{l_2} \cdots \overbrace{\sigma_d \cdots \sigma_d}^{l_d}$, for some sequences $\sigma_1, \sigma_2, \ldots, \sigma_d$ and integers $l_1, l_2, \ldots, l_d$, $1 \leq d \leq |T|$, such that*

(1). $(\forall 1 \leq i \leq d)(\forall t \in T)(\#_{\sigma_i}(t) \leq 1)$, and

(2). $(\forall 1 \leq i \leq d-1)(Tr(\sigma_{i+1}) \overset{\subseteq}{\neq} Tr(\sigma_i))$.

M's transitions using the second vector position to keep track of M's counter value while keeping the first vector position intact. The semilinearity of $R^1(M,q,q')$ follows immediately from [7], which shows the reachability sets of 2-dimensional VASSs to be effectively semilinear.

- $R^2(M,p,q)$={(c,d)|$(p,c) \xrightarrow{r} (q,d)$ and sequence r uses some test-for-zero moves}. Consider a computation $(p,c) \xrightarrow{r_1} (p_1,c_1) \xrightarrow{z} (p_1',c_1') \xrightarrow{r_3} (p_2,c_2) \xrightarrow{z'} (p_2',c_2') \xrightarrow{r_3} (q,d)$ such that z and z' are test-for-zero moves, and sequences r_1 and r_3 do not use any test-for-zero moves. (Here (p_1,c_1) and (p_2,c_2) are the first and last, respectively, configurations along the above computation at which test-for-zero moves are carried out. Also notice that (p_1,c_1) may be equal to (p_2,c_2) and z to z'.) We define a binary relation $\mathcal{R}$ on M_{state} such that $\bar{p}\mathcal{R}\bar{q}$ iff $(\bar{p},0) \xrightarrow{*} (\bar{q},0)$ in M. According to Lemma 3.2, using a depth-first search the relation $\mathcal{R}$ can effectively be computed. Now $R^2(M,p,q) = \{(c,d) \mid \exists \bar{p},\bar{q} \in M_{state}, (\bar{p}\mathcal{R}\bar{q}) \wedge (((p,c),(\bar{p},0)) \in R^1(M,p,\bar{p})) \wedge (((\bar{q},0),(q,d)) \in R^1(M,\bar{q},q))\}$, which is expressible in Presburger Arithmetic.

∎

Lemma 3.4 *A one-counter machine M (with initial state q_0) has an infinite non-ss computation iff one of the following holds:*

(1) there exists a configuration (q,r) such that $(q,r) \xrightarrow{+} (q,r)$ and $(q,r) \notin R(M,(q_0,0))$.

(2) there exist a configuration (q,r) and a short positive loop l starting from state q such that $(q,r) \xrightarrow{l} (q,r')$, $r' > r$, and $\forall\ j \geq 0, (q,r+j*(r'-r)) \notin R(M,(q_0,0))$. (Notice that test-for-zero transitions are absent in l.)

Proof: The if ($\Longleftarrow$) part is rather obvious; now we show the only if ($\Longrightarrow$) part.

Suppose $\sigma : d_1 \rightarrow d_2 \rightarrow \cdots \rightarrow d_i \rightarrow \cdots$ is a non-ss computation of infinite length. Then one of the following must hold:

(A) there exist indices i,j $(i<j)$ such that $d_i = d_j$, or

(B) the counter grows unboundedly and there exists an i such that the counter never becomes zero after d_i(= (q,r)) and there exists a short positive loop $l : q \xrightarrow{l} q$ (in M's transition diagram) appears infinitely often in σ.

If (A) is the case, then condition (1) follows immediately. Consider case (B). Let $(q,r) \xrightarrow{l} (q,r+d)$, for some $d>0$. We claim that $(q,r) \xrightarrow{l} (q,r+d) \xrightarrow{l} (q,r+2d) \cdots \xrightarrow{l} (q,r+3d) \cdots$ constitutes an infinite non-ss computation. If this is not the case, there must exist an integer $j > 0$ such that $(q,r+j*d) \in R(M,(q_0,0))$. Consider Figure 2, where d_h is the configuration such that l appears j times in $d_i \xrightarrow{*} d_h$. Clearly, $(q,r+j*d) \xrightarrow{\sigma_1\sigma_2\cdots\sigma_j} d_h$ is executable (for the test-for-zero operation is never used after d_i), which contradicts the assumption that d_h be on a non-ss computation σ. ∎

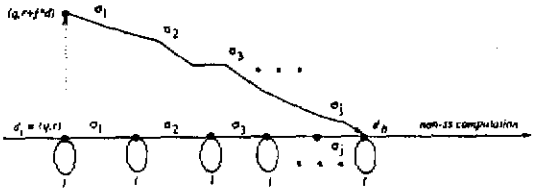

Figure 2. A short positive loop witnessing non-ss.

Theorem 3.5 *The ss problem for one-counter machines is decidable.*

Proof: The proof is carried out through the use of the unified strategy stated in Theorem 3.1. First consider Condition (1) of Theorem 3.1. For a one-counter machine M, Lemma 3.3 establishes the effective semilinearity of $REACH(M,q,q')$, $\forall\ q,q' \in M_{state}$. $SUCC(M,q,q')$ is trivially semilinear, for the finiteness of M_{tr}. $\forall q \in M_{state}$, an $r \in \mathbf{N}$ is in $DEAD(M,q)$ iff the following hold: (i) $(\forall q' \in M_{state})\ (\forall n \geq 0)((q,q',n) \notin M_{tr})$, (ii) $(r=0)$ implies $(\forall q' \in M_{state})((q,q',=0) \notin M_{tr})$, and (iii) $(r \geq 0)$ implies $(\forall q' \in M_{state})(\forall n, r > n \geq 0)((q,q',-n) \notin M_{tr})$; hence, $DEAD(M,q)$ is clearly effectively semilinear. Condition (2) of Theorem 3.1 is established in Lemma 3.4. In view of the above, the *ss* problem is decidable. ∎

3.2 Conflict-free Petri nets

A *Petri Net* (PN, for short) is a 3-tuple (P,T,φ), where P is a finite set of *places*, T is a finite set of *transitions*, and φ is a *flow function* φ : (P × T) ∪ (T × P) → {0, 1}. A *marking* (or *configuration*) is a mapping μ : P → N. A *marked* PN is a 2-tuple ($\mathcal{P}$, μ_0) where $\mathcal{P}$ (=(P,T,φ)) is a PN and μ_0 is a marking (μ_0 : P → N) called the *initial marking*. A transition t ∈ T is *enabled* at a marking μ iff for every p ∈ P, φ(p,t) ≤ μ(p). A transition t may *fire* at a marking μ if

t is enabled at μ. We then write $\mu \xrightarrow{t} \mu'$, where $\mu'(p) = \mu(p) - \varphi(p,t) + \varphi(t,p)$ for all $p \in P$. A sequence of transitions $\sigma = t_1 \ldots t_n$ is a *firing sequence* from μ iff $\mu \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \mu_n$ for some sequence of markings $\mu_1, \ldots, \mu_n$.

For ease of expression, the following notations will be used throughout the rest of this paper. Let σ, σ' be transition sequences, and t be a transition.

- $\#_\sigma(t)$ represents the number of occurrences of t in σ.
- $Tr(\sigma) = \{t | t \in T, \#_\sigma(t) > 0\}$, denoting the set of transitions used in σ.
- $\sigma \dot{-} \sigma'$ is defined inductively as follows. Suppose $\sigma' = t_1 \ldots t_n$. Let σ_0 be σ. If t_i is in σ_{i-1}, let σ_i be σ_{i-1} with the leftmost occurrence of t_i deleted; otherwise, let $\sigma_i = \sigma_{i-1}$. Finally, let $\sigma \dot{-} \sigma' = \sigma_n$. For example, if $\sigma = t_1 t_2 t_3 t_4 t_5$ and $\sigma' = t_4 t_3 t_1$, then $\sigma \dot{-} \sigma' = t_2 t_5$.

For firing sequences σ and σ', σ' is said to be a *rearrangement* of σ if $\#_\sigma(t) = \#_{\sigma'}(t), \forall t \in T$.

To simplify our notations, $REACH(\mathcal{P}, q, q')$, $REACH^+(\mathcal{P}, q, q')$, $Rreach(\mathcal{P}, q, q')$, $SUCC(\mathcal{P}, q, q')$ and $DEAD(\mathcal{P}, q)$ are abbreviated as $REACH(\mathcal{P})$, $REACH^+(\mathcal{P})$, $Reach(\mathcal{P})$, $SUCC(\mathcal{P})$, and $DEAD(\mathcal{P})$, respectively, since configurations (i.e., markings) of Petri nets belong to the set $\mathbf{N}^k$.

- *Conflict-free Petri nets*:
  A PN $\mathcal{P}$ =(P, T, φ) is said to be *conflict-free* iff for every place p, either
  1. $|p^\bullet| \leq 1$, or
  2. $\forall t \in p^\bullet$, t and p are on a self-loop (i.e., $t \in (p^\bullet \cap {}^\bullet p)$),

  where $p^\bullet = \{t \mid \varphi(p,t) > 0\}$ (resp., ${}^\bullet p = \{t \mid \varphi(t,p) > 0\}$) represents the set of output (resp., input) transitions of place p. In words, a PN is conflict-free if every place which is an input of more than one transition is also an output of each such transition [9]. In a conflict-free PN, once a transition becomes enabled, the only way to disable the transition is to fire the transition itself. (That is, $\forall t, t' \in T, t \neq t'$, $\mu \xrightarrow{t} \mu'$ and $\mu \xrightarrow{t'}$ implies $\mu' \xrightarrow{t'}$.)

To show the *ss* problem for conflict-free PNs to be decidable, we require the following lemmas.

Lemma 3.6 *Given a conflict-free PN $\mathcal{P}$=(P, T, φ), we can construct in nondeterministic polynomial time a system of linear inequalities $\mathcal{L}(\mathcal{P}, \mu_0, \mu)$ in such a way that $\mu \in R(\mathcal{P}, \mu_0)$ iff $\mathcal{L}(\mathcal{P}, \mu_0, \mu)$ has an integer solution. Furthermore, $\mathcal{L}(\mathcal{P}, \mu_0, \mu)$ remains linear even if μ_0 and μ are replaced by variables. (The reader is referred to Lemma 4.3 in [8] for a detailed description of the system of linear inequalities associated with $\mathcal{L}(\mathcal{P}, \mu_0, \mu)$.)*

Proof: (Sketch) The semilinearity of the reachability set for conflict-free PNs was originally illustrated in [9]. Subsequently, it has been shown in [8] that, given a conflict-free PN $\mathcal{P} = (P, T, \varphi)$, reachability can be characterized as a system of linear inequalities $ILP(\mathcal{P})$ over variables $v_1, \ldots, v_{|P|}, v'_1, \ldots, v'_{|P|}, x_1, \ldots, x_{|T|}$, among others, such that

(1) the size of $ILP(\mathcal{P})$ is bounded by a polynomial in the size of $\mathcal{P}$,

(2) given two markings μ and μ', $\mu \xrightarrow{\sigma} \mu'$, for some sequence σ, *if and only if* $ILP(\mathcal{P})$ has an integer solution while assigning μ to $(v_1, \ldots, v_{|P|})$ and μ' to $(v'_1, \ldots, v'_{|P|})$, and

(3) for each transition $t \in T$, the solution of variable x_t in $ILP(\mathcal{P})$ exactly equals the number of times transition t fires in σ.

∎

As a result, $REACH(\mathcal{P})$ is clearly effectively semilinear.

Lemma 3.7 *Given a PN $\mathcal{P}$=(P, T, φ), the set $DEAD(\mathcal{P})$ is effectively semilinear. (Here $\mathcal{P}$ can be a general Petri net (not necessarily conflict-free) for which the range of φ is* $\mathbf{N}$.)

Proof: Let $T = \{t_1, t_2, \cdots, t_m\}$. We define $\mathcal{H} = \{\{p_1, p_2, \ldots, p_h\} \subseteq P \mid \forall 1 \leq i \leq m, \exists 1 \leq j \leq h, \varphi(p_j, t_i) > 0\}$ (For each $H \in \mathcal{H}$ and $t_i \in T$, t_i has at least one input place in H.) $\mathcal{H}$ is clearly a finite set. We also let $\mu_H = \{\mu \in \mathbf{N}^k \mid \forall t_i \in T, \exists p_j \in H, \varphi(p_j, t_i) > \mu(p_j)$ and $\forall p' \notin H, \mu(p') = 0\}$, for a given $H \in \mathcal{H}$. Notice that μ_H is finite, which contains dead markings only. If μ' is a marking such that $(\mu'(p_j) = \mu(p_j), \forall 1 \leq j \leq h)$, and $(\mu'(p') \geq \mu(p'), \forall p' \notin H)$, then μ' is a dead marking as well. Hence the set of dead markings equals $\bigcup_{H \in \mathcal{H}} \bigcup_{\mu \in \mu_H} \{\mu' \mid (\mu' \geq \mu) \wedge (\mu'(p) = \mu(p), \forall p \in H)\}$, which is semilinear since $\mathcal{H}$ and μ_H range over finite sets. ∎

Lemma 3.8 *(From [12]) For an arbitrary path $\mu \xrightarrow{\sigma} \overline{\mu}$ in a conflict-free PN $\mathcal{P}$=(P, T, φ), σ can be rearranged into $\overbrace{\sigma_1 \cdots \sigma_1}^{l_1} \overbrace{\sigma_2 \cdots \sigma_2}^{l_2} \cdots \overbrace{\sigma_d \cdots \sigma_d}^{l_d}$ for some sequences $\sigma_1, \sigma_2, \ldots, \sigma_d$ and integers $l_1, l_2, \ldots, l_d$, $1 \leq d \leq |T|$, such that*

(1). $(\forall 1 \leq i \leq d)(\forall t \in T)(\#_{\sigma_i}(t) \leq 1)$, and

(2). $(\forall 1 \leq i \leq d-1)(Tr(\sigma_{i+1}) \overset{\subseteq}{\neq} Tr(\sigma_i))$.

The above lemma allows us to rearrange an arbitrary firing sequence in a conflict-free PN into a 'canonical form' satisfying conditions (1) and (2), in which (1) ensures that for each σ_i, no transition will appear more than once (hence, $|\sigma_i| \leq |T|$, for all i), and (2) says that $Tr(\sigma_1)Tr(\sigma_2)\cdots Tr(\sigma_d)$ forms a 'shrinking' sequence of sets in the sense that if a transition, say t, occurs in σ_i, for some i, then t is guaranteed to appear in $\sigma_j, \forall 1 \leq j \leq i$. Symmetrically, if t does not appear in σ_i, then it will never be found in $\sigma_j, \forall i \leq j \leq d$.

Lemma 3.9 *If an infinite non-ss computation exists in a conflict-free PN $\mathcal{P}=(P,T,\varphi)$ (with initial marking μ_0), then there exist markings μ, μ' and a 'short' sequence τ (i.e., $\#_\tau(t) \leq 1, \forall t \in T$) such that $\mu \xrightarrow{\tau} \mu'$, $\mu' \geq \mu$, and $\mu \xrightarrow{\tau\tau\tau\cdots}$ is an infinite non-ss computation.*

Proof: Consider an arbitrary infinite non-ss computation δ from μ_1. Suppose we consider the suffix computation δ' along which all the transitions appear infinitely many times. By applying Dickson's Lemma [2] to δ', δ can be written as $\delta_1\sigma\delta_2$ such that $\mu_1 \xrightarrow{\delta_1} \mu \xrightarrow{\sigma} \mu'' \xrightarrow{\delta_2}$ with $\mu'' \geq \mu$, and all the transitions in σ occur infinitely often in δ. Let $\overbrace{\sigma_1\cdots\sigma_1}^{l_1}\overbrace{\sigma_2\cdots\sigma_2}^{l_2}\cdots\overbrace{\sigma_d\cdots\sigma_d}^{l_d}$ be the canonical rearrangement of σ guaranteed by Lemma 3.8. Clearly σ_1 must be that if $\mu \xrightarrow{\sigma_1} \mu'$, then $\mu' \geq \mu$; otherwise, $\exists p, \mu''(p) < \mu(p)$ due to Condition (2) of Lemma 3.8. (To see this, the conflict-freedom property and (2) of Lemma 3.8, together with the fact that $\forall p \in P, t \in T, \varphi(p,t) \leq 1$, imply that no transitions will deposit tokens to a place p in any of the segments $\sigma_1, \ldots, \sigma_d$ if $\mu'(p) < \mu(p)$.) Finally, since all the transitions in σ_1 occur infinitely often in the infinite non-ss computation, we claim that $\mu \xrightarrow{\sigma_1\sigma_1\cdots}$ constitutes an infinite non-ss computation. Suppose, in contrast, that for some n, $\mu \xrightarrow{\overbrace{\sigma_1\cdots\sigma_1}^{n}} \mu_n$ and $\mu_n \in R(\mathcal{P},\mu_0)$. Let $\pi = \overbrace{\sigma_1\cdots\sigma_1}^{n}$. Let δ_3 be a prefix of $\sigma\delta_2$ such that $\#_\pi(t) \leq \#_{\delta_3}(t), \forall t \in T$ (i.e., all the transitions of π occur in δ_3). Assume that $\mu \xrightarrow{\delta_3} \bar{\mu}$, for some $\bar{\mu} \notin R(\mathcal{P},\mu_0)$. Due to the conflict-freedom property of $\mathcal{P}$, $\mu \xrightarrow{\pi}$, $\mu \xrightarrow{\delta_3}$, and $(\#_\pi(t) \leq \#_{\delta_3}(t), \forall t \in T)$ imply the existence of a δ_4 such that $\pi\delta_4$ is a permutation of δ_3, and $\mu \xrightarrow{\pi} \mu_n \xrightarrow{\delta_4} \bar{\mu}$ (see [9]), contradicting the assumption that $\bar{\mu} \notin R(\mathcal{P},\mu_0)$. By letting $\tau = \sigma_1$, our result follows. ∎

We are now in a position to show the *ss* problem to be decidable.

Theorem 3.10 *The ss problem for conflict-free PNs is decidable.*

Proof: Again, the proof is done through the use of Theorem 3.1. The semilinearity of $REACH(\mathcal{P})$ and $DEAD(\mathcal{P})$ follows from Lemmas 3.6 and 3.7, respectively. T being finite clearly implies the semilinearity of $SUCC(\mathcal{P})$. As a consequence, Condition (1) of Theorem 3.1 holds. Finally, Condition (2) of Theorem 3.1 follows immediately from Lemma 3.9. This completes the proof of the theorem. ∎

4 Fair self-stabilization

Given a PN $\mathcal{P}$ with initial marking μ_0, an infinite computation $\mu_1 \xrightarrow{t_1} \mu_2 \xrightarrow{t_2} \cdots \mu_i \xrightarrow{t_i} \mu_{i+1} \cdots$ is said to be *fair* if for every transition t, if t is enabled at infinitely many μ_{i_l} $(l \geq 1)$, then there exist infinitely many j_l $(l \geq 1)$ such that $t_{j_l} = t$. (In words, if a transition is enabled infinitely many times, then the transition must occur infinitely often as well.) A computation σ from marking μ_1 is said to be *fair non-self-stabilizing* iff one of the following holds:

(1) σ $(\mu_1 \rightarrow \mu_2 \rightarrow \cdots \rightarrow \mu_m$, for some $m)$ is finite such that μ_m is a 'dead' marking (i.e., μ_m has no immediate successor in $\mathcal{P}$) and $\mu_m \notin R(\mathcal{P},\mu_0)$, or

(2) σ $(\mu_1 \rightarrow \mu_2 \rightarrow \cdots \rightarrow \mu_i \rightarrow \cdots)$ is infinite and fair such that $\forall i \geq 1, \mu_i \notin R(\mathcal{P},\mu_0)$.

We let $FSS(\mathcal{P},\mu_0) = \{\mu \mid$ none of the computations emanating from μ is *fair non-self-stabilizing*$\}$. The *fair self-stabilization problem* is that of deciding whether starting from an arbitrary marking μ, none of its computations is *fair non-self-stabilizing*, i.e., $FSS(\mathcal{P},\mu_0) = N^k$.

Given a set of markings S, the *successor* (resp., *predecessor*) of S, written as $succ(S)$ (resp., $pred(S)$), is the set $\{\mu \mid \exists t \in T, \mu' \in S, \mu' \xrightarrow{t} \mu\}$ (resp., $\{\mu \mid \exists t \in T, \mu' \in S, \mu \xrightarrow{t} \mu'\}$). Let $pred^*$ (resp., $succ^*$) be the reflexive and transitive closure of $pred$ (resp., $succ$). (That is, $succ^*(S) = \{\mu \mid \exists \mu' \in S, \mu' \xrightarrow{*} \mu\}$, and $pred^*(S) = \{\mu \mid \exists \mu' \in S, \mu \xrightarrow{*} \mu'\}$.) The sets $succ^*(S)$ and $pred^*(S)$ will be referred to as the *forward reachability set* and the *backward reachability set* of S, respectively. Notice that $R(\mathcal{P},\mu_0) = succ^*(\{\mu_0\})$. A set of markings S is said to be *forward-closed* if $\forall \mu \in S, \forall t \in T, \mu \xrightarrow{t} \mu'$ implies $\mu' \in S$.

Given a PN $\mathcal{P} = (P, T, \varphi)$, a *potential function* f is a mapping $N^k \rightarrow N \cup \{\infty\}$, i.e., f maps each marking μ to a number in $N \cup \{\infty\}$. (The value of $f(\mu)$ is called the *potential* of marking μ.) A potential function f is said to be *monotone* if for every marking μ, if $\mu \xrightarrow{t} \mu'$ (for some marking μ' and transition t), then $f(\mu) \geq f(\mu')$.

Given a conflict-free PN $\mathcal{P}=(P,T,\varphi)$ and a set of markings S, the following potential function f will be used throughout the rest of this paper: $f(\mu)$ is defined to be the

length of the shortest path from μ to a marking in S; if μ cannot reach S, $f(\mu)$ is ∞. Notice that $\forall \mu \in S, f(\mu) = 0$ (i.e., S defines the set of markings of zero potential). What follows is another way to view such a potential function. We partition N^k into a sequence of disjoint sets of markings $U_0, U_1, ..., U_\infty$ such that

$$U_0 = S$$

$$U_1 = (pred(U_0)) - U_0$$

...

$$U_i = (pred(U_{i-1})) - (\cup_{j=0,...,i-1} U_j), i \geq 1$$

$$U_\infty = N^k - (\cup_{j \geq 0} U_j)$$

It is not hard to see that $f(\mu) = i$ iff $\mu \in U_i$.

Lemma 4.1 *Given a conflict-free PN $\mathcal{P}$ and a forward-closed set S, let f be the potential function based upon the shortest path criterion defined above. The following hold:*

(1) f is always monotone.

(2) For an arbitrary μ and a path $\mu \xrightarrow{\delta} \mu_1$, if $\mu \xrightarrow{\sigma} \mu'$ ($\mu' \in S$) is one of the shortest paths reaching S and $Tr(\delta) \cap Tr(\sigma) \neq \emptyset$, then $f(\mu_1) < f(\mu)$.

Proof: We first consider (1). It suffices to show that $\mu \xrightarrow{t} \mu_1$ implies $f(\mu) \geq f(\mu_1)$. Consider the following cases:

(a) ($f(\mu) = \infty$:) In this case, $f(\mu_1) = \infty$; otherwise, $\mu \xrightarrow{t} \mu_1 \xrightarrow{*} \mu'', \mu'' \in S$, violating the assumption that $f(\mu) = \infty$.

(b) ($0 < f(\mu) < \infty$:) Let $\mu \xrightarrow{\sigma} \mu' (\mu' \in S)$ be one of the shortest paths reaching S. Consider two cases:

(i) $t \notin Tr(\sigma)$. Since the PN is conflict-free, t is enabled at μ'. Suppose $\mu' \xrightarrow{t} \mu''$, for some $\mu'' \in S$ (since S is forward-closed). Clearly, $\mu_1 \xrightarrow{\sigma} \mu''$, yielding $f(\mu_1) \leq f(\mu)$.

(ii) $t \in Tr(\sigma)$. Suppose $\mu \xrightarrow{\sigma_1} \mu_2 \xrightarrow{t} \mu_3 \xrightarrow{\sigma_3} \mu'$ (for some μ_2, μ_3), where $\sigma = \sigma_1 t \sigma_2$ and t is not in σ_1. Again due to the conflict-freedom property of the PN, $\mu_1 \xrightarrow{\sigma_1} \mu_3 \xrightarrow{\sigma_3} \mu'$; hence, $f(\mu_1) \leq f(\mu) - 1$.

Figure 3 illustrates the monotone property of the potential function for paths in a conflict-free PN.

Now consider (2). Let $\sigma = \sigma_1 t \sigma_2$, where t is the first occurrence of a transition that is also in δ. Suppose $\delta = \delta_1 t \delta_2$ ($t \notin \delta_1$), and $\mu \xrightarrow{\delta_1} \mu_2 \xrightarrow{t} \mu_3 \xrightarrow{\delta_2} \mu_1$. (Notice that the selection of t ensures that $Tr(\delta_1) \cap Tr(\sigma) = \emptyset$.) It is not hard to see that σ_1 is enabled at μ_3, since $Tr(\sigma_1) \cap Tr(\delta_1 t) = \emptyset$ and PN $\mathcal{P}$ is conflict-free. We immediately have $\mu_3 \xrightarrow{\sigma_1 \sigma_2} \mu_4$, for some $\mu_4 \in S$ (since $\mu' \xrightarrow{\delta_2} \mu_4$ and S is forward-closed). Hence, $f(\mu_3) \leq |\sigma_1 \sigma_2| = f(\mu) - 1$. See Figures 3 and 4. Using the result of (1), (2) follows ∎

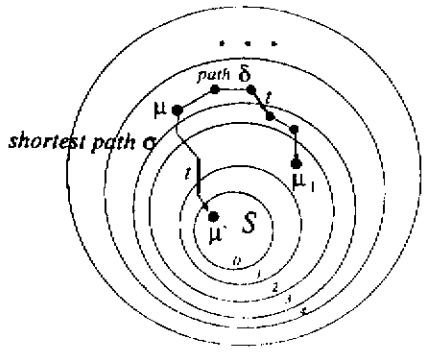

Figure 3. Paths and the associated potentials.

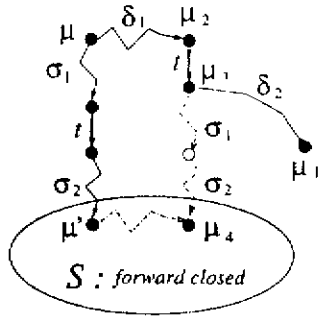

Figure 4. Proof of Lemma 4.1.

Theorem 4.2 *Given a conflict-free PN $\mathcal{P} = (P, T, \varphi)$ and an initial marking μ_0, $\mu \in FSS(\mathcal{P}, \mu_0)$ iff $\mu \in pred^*(R(\mathcal{P}, \mu_0))$.*

Proof: Let S be $R(\mathcal{P}, \mu_0)$, which is clearly forward-closed. We let the potentials of those markings in S be zero. If $\mu \notin pred^*(S)$, none of the (finite or infinite) computations can reach $R(S)$; hence $\mu \notin FSS(S)$.

Suppose $\mu \in pred^*(S)$. If μ has a fair (infinite) path which never reaches S, then there must exist a marking μ_1 and sequences σ_1 and σ_2 such that $\mu \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2}$, $f(\mu) \geq f(\mu_1) > 0$ and $\mu_1 \xrightarrow{\sigma_2}$ is fair along which the potential never drops. Let $\mu_1 \xrightarrow{t_1 t_2 ... t_i} \mu' (\mu' \in S)$ be one of the shortest paths reaching some marking in S. The fairness assumption

ensures that $t_1 \in Tr(\sigma_2)$; otherwise, t_1 would have been enabled infinitely often without being fired. By Lemma 4.1, the potential along σ_2 eventually drops – a contradiction. As a consequence, from μ all infinite computations avoiding S are unfair. Using a similar argument, we can show that if $\mu \stackrel{*}{\rightarrow} \mu'$ and μ' is a dead marking, then $\mu' \in S$. ∎

According to Lemma 3.6, checking reachability for conflict-free PNs can be equated with solving the integer linear programming problem, which is known to be in NP. It is important to point out that, as μ_0 and μ in Lemma 3.6 can be regarded as variables, the backward reachability set $pred^*(R(\mathcal{P}, \mu_0)$ is readily expressible in terms of integer linear programming. Hence, the following is obvious.

Corollary 4.3 *The fair self-stabilization problem for conflict-free Petri nets is decidable.*

5 Conclusions and future research

We have studied, from the decidability viewpoint, the problem of determining whether a given system is self-stabilizing or not for infinite-state systems including *one-counter machines* and *conflict-free Petri nets*. The key behind our strategy is that for each of the three classes of systems, the reachability set from an arbitrary configuration is effectively semilinear, and it suffices to consider infinite non-self-stabilizing computations with periodic behaviors. It is interesting to see whether our strategy can be applied to a wider class of infinite-state systems, or a similar strategy (without relying on the reachability set being semilinear) can be extended to systems not enjoying the semilinearity property. An equally important direction of future research is to find out the exact complexity of the self-stabilization problem for the above classes of systems. We have also studied the fair version of self-stabilization, and a decidability result was shown for the class of conflict-free Petri nets.

Another issue that deserves further investigation is to consider the problem with respect to the notion of 'self-stabilization with probability one.' In spite of having some non-self-stabilizing computations, in practice a system might be considered 'fault-tolerant' if the probability of self-stabilization equals one. Finally, self-stabilization for real-time systems deserves further investigation, as many real-world systems are of real-time nature. For the model of *timed automata*, the *region graph* technique can easily be applied to showing the decidability (in fact, in PSPACE) of self-stabilization (in the sense defined in this paper). However, to cope with the nature of real-time systems, one might have to tailor the notion of self-stabilization to better capture the intuitive concept of 'self-stabilizing in a certain amount of time,' as opposed to 'reaching a legitimate configuration eventually' as defined in the conventional sense.

References

[1] Cherkasova, L., Howell, R. and Rosier, L. Bounded self-stabilizing Petri nets, *Acta Informat.*, 32:189-207, 1995.

[2] Dickson, L., Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors, *Amer. J. Math.*, 35:413-422, 1913.

[3] Dijkstra, E. Self-stabilizing systems in spite of distributed control, *C. ACM*, 17, pp. 643-644, 1974.

[4] Ginsburg, S., *The mathematical theory of context-free languages*, McGraw-Hill, 1966.

[5] Gouda, M., Howell, R. and Rosier, L. The instability of self-stabilization, *Acta Informat.*, 27, pp. 697-724, 1990.

[6] Herman, T., A comprehensive bibliography on self-stabilization, *Chicago Journal of Theoretical Computer Science*, (http://www.cs.uiowa.edu/ftp/selfstab/bibliography), 1998.

[7] Hopcroft, J. and Pansiot, J., On the reachability problem for 5-dimensional vector addition systems, *Theoret. Comput. Sci.*, 8(2):135-159, 1979.

[8] Howell, R. and Rosier, L. and H. Yen, Normal and sinkless Petri nets, *Journal of Computer and System Sciences*, 46:1–26, 1993.

[9] Landweber, L. and Robertson, E., Properties of conflict-free and persistent Petri nets, *J. Assoc. Comput. Mach.*, 25:352–364, 1978.

[10] Presburger, M., Über die vollständigkeit eines gewissen systems der arithmetik ..., Comptes rendues du premier Congres des Math. des Pays Slaves, Warsaw, 92-101, 395, 1929.

[11] Rosier, L. and Yen, H., Logspace hierarchies, polynomial time and the complexity of fairness problems concerning ω-machines. *SIAM J. Computing*, 16(5):779-807, 1987.

[12] Yen, H., Wang, B. and Yang, M., Deciding a class of path formulas for conflict-free Petri nets, *Theory of Computing Systems*, 30(5):475-494, 1997.